Comparing Similarity of Patent Textual Data Through the Application of Machine Learning

BY

SALVATORE C. IMMORDINO
BS, Northern Illinois University, 1992
MS, Marquette University, 2004

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Industrial Engineering and Operations Research
in the Graduate College of the
University of Illinois at Chicago, 2019

Chicago, Illinois

Defense Committee:

Michael J. Scott, Chair and Advisor
Houshang Darabi
Sybil Derrible, Civil and Materials Engineering
Mengqi Hu
Jelena Spanjol, Ludwig-Maximilians-Universität München

Copyright by

SALVATORE C. IMMORDINO

2019

To Tracy,

My amazing wife, who has believed in me since the day we met.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Dr. Michael Scott for his continuous support of my PhD study, for his patience, motivation, and consistent encouragement. His guidance helped me in my research and writing of this thesis. I could not have asked for a better advisor and mentor for my PhD study.

My sincere thanks to USG and my industry colleagues and friends Dr. Dominic Dannessa, Dr. Srinivas Veeramasuneni, and Dr. Kumar Natesaiyer who provided unwavering support and encouragement throughout this journey.

Last but not the least, I am grateful to Dr. Sybil Derrible who opened my eyes to the potential of data science.

SCI

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF ABBREVIATIONS

USPTO	United States Patent and Trademark Office
USPC	United States Patent Classification
NBER	The National Bureau of Economic Research
CPC	Cooperative Patent Classification
WIPO	World Intellectual Property Organization
NLP	Natural Language Processing
PCA	Principle Component Analysis
IP	Intellectual Property
VSM	Vector Space Model
LOE	Loss of Exclusivity
ECLA	European Classification System
IPC	International Patent Classification
NaN	Not a Number
ANN	Artificial Neural Network
PAP	Patent Examiner Performance Appraisal Plan
CSS	Cosine Similarity Score
DTM	Document Term Matrix

LIST OF ABBREVIATIONS (Continued)

VSM	Vector Space Model
HITL	Human-in-the-loop
SVD	Single Value Decomposition
CIP	Continuation-in-part
RCE	Request for continued examination
VOC	Voice of Customer
NLTK	Natural Language Toolkit
POS	Part of Speech

SUMMARY

The objective of this research is to understand if machine learning (ML) techniques, including natural language processing (NLP), can be applied to patent data to help *domain experts* efficiently answer important questions about how to manage newly created intellectual property (IP) by revealing latent relationships between IP documents. Domain experts are authorities in a specific area of technical knowledge based upon years of experience, education, and training. They are typically the first people consulted when it comes to assessing the value of a new invention and how to protect the inventive knowledge associated with it. The first step in making this assessment is usually a determination of whether the invention has any commercial value, and if so, how best to protect the innovation. There are a number of methodologies and techniques used to make this initial assessment including value engineering, value stream mapping, voice of customer research (VOC), and price sensitivity analysis.

Once a determination is made that an invention has commercial value a number of additional questions are raised. For example: Should we keep the inventive knowledge associated with the innovation secret or should we seek patent protection? What is the likelihood that our inventive knowledge will be discovered independently or reverse engineered? How likely is it that a competitor would develop a comparable invention and how long would it take them to do so? and if they were to develop a comparable invention would we be able to know if they were infringing on our intellectual property? The latter set of questions go to the heart of the issue at hand. It basically comes down to the question "how much time do we have before the knowledge of this invention is disseminated?"

The answers to these questions are typically subjective and while domain expertise continues to be a crucial component of intellectual property analysis, reliance on intuition and subjective opinion to make important innovation related decisions fails to exploit a wealth of information in the patent record. Furthermore, patterns extracted objectively from large data-sets, such as the patent record, hold vastly more data than even the most knowledgeable domain experts can process. Unfortunately, most of the textual data is unstructured, making it difficult and time consuming for humans, even domain experts, to process quickly, let alone decipher complex relationships between invention knowledge flow timing. This research attempts to understand invention patterns by studying the textual data used by inventors to describe their inventions. It uses NLP techniques to distill inventors' words down to core descriptive term combinations that are representative of one or more inventive steps. These latent combinations of words represent discrete inventive units referred to here as *inventive descriptors.* By measuring patents based upon the similarity of their respective inventive descriptors it is possible to measure latent relationships between inventive documents that can be used to assess invention impact as well as understand inventive knowledge dissemination patterns. These insights can in turn be used to answer questions with regard to likely competitive responses by using historical patterns from the patent record as a guide.

This approach is distinct from a wide body of research used to assess inventive impact that is reliant upon either patent surveys or statistical methods applied to numerical data derived from patents. While patent surveys can be used to gain insights into the independent variables that drive innovation it is difficult to compare results over time when using qualitative and potentially subjective measures. Statistical methods rely heavily upon constructed metrics derived from patents, such as measuring patent counts. However, counting alone is not a suitable proxy for inventiveness as the individual technological impact and quality of each patent may differ. To overcome this, many analysis techniques have been developed, including patent indicators, patent landscape maps, and patent citation analysis. The most prevalent of these is citation analysis, which is based upon the optimistic assumption that patent seekers in good faith cite any body of work that they relied upon at the time of patent application filing. Insights gained by studying statistical methods applied to patent numerical data raised doubts as to whether patent citation data and patent counts were the best modes of assessing overall inventiveness or measuring inventive knowledge flow. This is due to a number of changes made to the patent process over time that can directly impact citation behavior. The fundamental problem is that most citations (unlike the academic world) are done after an invention has been conceived, either by the patent attorney handling the filing or the examiner themselves.

Independent of my research, Jaffe and de Rassenfosse [1] also highlighted many concerns in their survey review of patent citation data use over the last two decades. The rationale behind current citation methods no longer holds and there is good reason to believe that citation counts cannot completely capture knowledge flow. This research presents the application of ML and NLP techniques to a subset of United States Patent and Trademark Office (USPTO) patent documents to assess whether it can be used to discover latent relationships between patent documents and in turn be used to measure invention impact and study diffusion of inventive knowledge over time. Patents are uniquely suited for this analysis because of the rules placed upon patent seekers to describe their inventions in such detail that they can be used to define legal boundaries of exclusivity.

This approach is twofold. First, develop an improved computational process to enhance how domain experts determine the core inventive aspects of any patented invention. Ideally, this process could be used to reduce the time-consuming aspects of manual patent searches by culling out core inventive knowledge quickly, so that valuable expert time can be used more efficiently. Second, develop new techniques to measure *inventiveness* and *knowledge flow* through the application of ML using newly constructed textual-based cosine similarity measures as dependent variables for intellectual property decision modeling.

This approach begins by first extracting unstructured abstract, title, and claim textual data from a subset of patent documents from a competitive group of companies and then utilizing NLP techniques to convert that data into a vector space model (VSM) using a numerical statistical method called term frequency-inverse document frequency (tf-idf). This technique converted the sets of inventive descriptors associated with each patent into a representative numerical vector within the vector space model. The next step measures the cosine angle between patent vectors within the higher-dimension vector space and establishes a minimum cosine similarity threshold to select for forward cosine related patents. Since each vector space is different the selection of an appropriate threshold is usually dependent upon a human to provide subject relevance for an appropriate threshold. This research utilizes the existing patent record to establish a minimum threshold using statistical analysis applied to known forward cited patent pairs where a human (e.g., patent attorney or patent examiner) has already determined relatedness by citing patents. Thus it exploits the patent record by using the cosine angle of cited patent pairs as a basis to establish a CSS threshold. The result is an ability to count patents based upon cosine relatedness which can be directly substituted for citation counts. For example, measurements like total citation count (inventiveness) and mean forward citation lag (knowledge flow) can use the CSS relatedness count value. To visualize patent cosine relatedness the higher-dimension vector space was reduced to a two-dimensional plot first by using principal component analysis and then t-distributed stochastic neighbor embedding. The resulting visualization was then color coded using patent office technical classifications to reveal some discrete technology class clusters. Lastly, a comparison of invention impact and inventive knowledge flow is demonstrated by plotting patent citation data alongside cosine relatedness outputs to reveal both similar and dissimilar inventive patterns.

This research shows that new computational techniques can be used to convert unstructured patent textual data into actionable knowledge by revealing latent relationships between patents. It accomplishes this by relying upon the language used by inventors to describe their inventions and in doing so lays the ground work to study inventiveness and knowledge flow with less dependence on domain expertise and metrics derived from patent counts and citation analysis. The increased understanding can be used to study inventive knowledge flow which has both scientific and practical applications. The conversion of unstructured patent textual data into structured data that can be used for intellectual property decision modeling, and the methodological enhancements provided to researchers in the field provides increased understanding of the nature of inventiveness and inventive knowledge flow. There is also significant potential to reduce the most time-consuming and tedious aspects of patent searches, to not only free up valuable domain expert time, but also provide them with more actionable knowledge to make informed intellectual property decisions. Going forward these techniques could be used in a number of applications including novelty assessment of patent applications prior to filing, competitive surveillance to monitor published patent applications, an automated patentability assessment tool, or as a means to suggest new inventive steps by combining inventive descriptors between CSS related patents.

CHAPTER 1

INTRODUCTION

How do business leaders effectively manage new inventive knowledge when it is created at the corporate level? The first step in answering this question is usually a determination of whether the new inventive knowledge has any commercial value, and if so, how best to protect it. This in turn leads to a series of additional questions. For example: Should we keep the inventive knowledge secret or seek patent protection? What is the likelihood that our inventive knowledge will be discovered independently or reverse engineered and how long would it take to do so? How likely is it that a competitor would develop a comparable invention without infringing on our patent and would we be able to know if they were infringing? In answering these questions, most business leaders rely heavily on subjective opinion, historical market behaviors, and *domain experts*. Domain experts, also known as subject matter experts, are authorities in a specific area of technical knowledge. They often have years, even decades, of experience, or specific skills in a particular area from significant training and educational investments. While domain expertise continues to be a crucial component of intellectual property analysis, reliance on intuition and subjective opinion to make important intellectual property decisions fails to exploit a wealth of information in the patent record. Furthermore, patterns extracted objectively from large data-sets, such as the patent record, hold vastly more data than even the most knowledgeable domain experts can process. Unfortunately most of the information contained within the patent system is in the form of unstructured textual data making it difficult and time consuming for humans, even domain experts, to make informed decisions. The purpose of this research is to understand if ML techniques can be applied to patent data to help domain experts answer the above questions in an efficient manner that is less dependent on subjective opinion. This research attempts to understand invention patterns within the construction industry by studying the textual data used by inventors to describe their inventions. The hypothesis is that patent texts can be mined for combinations of words and unique expressions that represent discrete inventive units and by measuring patents based upon the similarity of their inventive descriptors it will be possible for domain experts to answer many of the invention impact and knowledge dissemination related questions more objectively.

1.1 Motivation

Patent information can be used to study the flow of inventive knowledge through space, time, and technology domains, making it a valuable tool for the measurement of scientific progress and to study what factors influence technological change. The World Intellectual Property Organization (WIPO), a specialized agency of the United Nations (UN) created in 1967, lists [2] that there are over 200 patent offices worldwide representing 191 member states [3]. The overwhelming amount of unstructured textual data associated with this inventive knowledge is beyond the capacity of humans to study holistically; however, with advances in ML techniques and ever-increasing computational power it is possible that this data can be turned into comprehensible, structured, and actionable knowledge. A key area where such actionable knowledge would be helpful is at the corporate level where the decision of whether to apply for a patent, and as a result make inventive knowledge public, carries inherent financial risks. This research attempts to understand invention patterns within the construction industry by studying the textual data used by inventors to describe their inventions. The goal is to efficiently reveal core inventive concepts, assess overall invention impact, and understand how those inventive concepts diffuse over time. These insights can be used to make more informed decisions when it comes to protecting *intellectual property.* Intellectual property includes creations of the mind, such as inventions, literary and artistic works, designs, and symbols, names, and images used in commerce [4]. This research methodology builds upon existing ML and NLP techniques to convert unstructured patent textual data into structured data such as those outlined by [5] who proposed a text mining methodology for full-text patent analysis based upon methods used by patent analysts. Their methodology includes various text mining techniques including segmentation, summarization, feature selection, term association, and topic clustering. Their goal was to improve the process of patent analysis by automating many of the specialized, time consuming, and tedious steps performed by analysts. Tseng *et al.* articulate well the motivation for automated patent analysis when they state:

> *"...these processes require the analysts to have a certain degree of expertise in information retrieval, domain-specific technologies, and business intelligence. This multi-discipline requirement makes such analysts hard to find or costly to train."* (1)

This research leverages conventional methods of NLP, patent office application requirements, and methods used by corporate domain experts to discover patterns, relations, and trends among intellectual property documentation. The premise is that there are high value combinations of words and expressions that represent discrete inventive units within patents. Analyzing patents based upon the similarity of these inventive sets should make it possible to not only measure inventive knowledge dissemination over time but also gain insight into the potential impact of that inventive knowledge. This impact can be measured within a given technology sector or broadly across many sectors. This research focuses on the technology area of building science & construction using patent data from the USPTO. It begins with a historical review on patents and how they have been used to study *inventiveness*, which is defined as a measure of the technological significance that an invention creates. This includes an overview of historical methods used to study invention impact and inventive knowledge diffusion followed by a review of the evolving patent process in the United States including a discussion of patent types, classification methods, requirements, exclusivity periods, and rights. It then presents the methodology used including a direct comparison to patent citation analysis to reveal interesting similarities and dissimilarities. Lastly, it finishes with a conclusion and a discussion of potential areas of future research as access to latent relationships have raised new questions.

1.2 Background

The United States Patent and Trademark Office (USPTO), which was created in 1836, has issued over 6 million patents [6]. The USPTO issues patents to inventors thereby granting them special rights for their intellectual property. This fulfills the USTPO's mandate which is outlined in the United States Constitution under Article I, Section 8, Clause 8:

> *"To promote the Progress of Science and useful Arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries."* (2)

A patent represents a government sanctioned right to exclude others from practicing a specified invention for a *limited time*, in the U.S., 20 years from the earliest filing date, thereby creating an economic incentive for disclosing valuable inventive knowledge to the public. This in turn facilitates societal progress as the inventive knowledge diffuses over time and becomes part of the public domain. Patent holders receive a legal monopoly to exclude others from practicing their inventions, however, a new inventor can be prevented from practicing their own invention if it infringes upon the rights holder of another enforceable patent. Such scenarios occur because the process of invention is an inherently iterative as it builds upon the prior inventive knowledge of others. [1] provide an elegant description of this process:

> *"First, we can think of all possible technologies as mapping onto a high-dimensional technology space, such that a given invention can be located in that space, and a patent represents the right to exclude others from marketing products that impinge upon specific region (or regions) of that space. Second, the invention process is cumulative, that is, inventions build on those that came before and, in turn, facilitate those that come after. In this geometric interpretation, the patent claims delineate the metes and bounds of the region of technology space over which exclusivity is being granted, while the citations indicate previously marked off areas that are in some sense built upon by or connected to the invention being granted."* (3)

This research aims to show that ML & NLP can be used to create this geometric interpretation by converting patent title, abstract, and claim textual data into numerical vectors that can be placed into a higher-dimensional technology space. Furthermore, the metes and bounds can be delineated by measuring the cosine angle between these vectors.

1.3 The Study of Inventiveness

Patents have been used by researchers to study inventiveness on a global, corporate, and individual level for decades. Jacob Schmookler proposed the relationship between patent counts and economic performance among industries [7]. Subsequent research has shown that a country's collective technological advancement is directly correlated to international competitiveness and economic prosperity [8–11]. Patent data analysis has been used to study how inventive knowledge flows between corporate and not-for-profit entities [12], to understand the impact of R&D investment strategy [13], and as a competitive business intelligence tool where the intellectual property filing behavior of rival businesses is analyzed [14–16]. Patents have been used as a measure of corporate innovative performance [17–19] and to strategically assess technical advances among competitors for stock market valuation and merger and acquisition targeting [20]. On an individual basis, patent data has been used as a performance metric for patent office examiners [21] and as a means to assess R&D employee engagement [22]. It has also been used to visualize inventor networks within organizations [23] and to compare innovation culture between business entities [24]. Patent data has even been used to study the movement of inventors over space and time using geographical data collected from patent documents [25–28]. While there is a wide body of research regarding the use of patent data to measure inventive activity, much of the historical analysis focuses on the use of patent surveys or statistical methods applied to the numerical structured data derived from patents. While the former can be used to gain unique insights into the independent variables that drive innovation, it is difficult to compare results over time, especially when using qualitative and potentially subjective measures. The latter relies heavily upon statistics applied to patent counts. However, counting patents alone is not a suitable proxy for inventiveness as patents differ in technological impact and quality. Patent quality reflects the degree to which an invention meets the statutory requirements enforced by the patent office. For example, inventors are required to articulate in written form the best mode of their invention while also showing novelty over prior inventions. High-quality patents are those that clearly articulate the scope and boundaries of a claimed invention over others with little ambiguity. When patent rights are asserted against others the validity of the asserting patent can be challenged if it can be shown that they did not meet the requirements of the patent office. To overcome differences in patent impact and quality, many analysis techniques have been developed, including patent change mining, patent indicators, patent maps, and patent citation analysis. The most prevalent of these is patent citation analysis, which is based upon the assumption that inventors, in good faith, list any evidence that their invention is already known at the time of filing. While the use of citation metrics has been shown to be an effective means of weighting patent counts, the methods used for discovering prior art citations, and the incentives for doing so, can result in missed or incorrect citations by either the patent examiner or applicant. The fundamental problem is that most citations (unlike the academic world) are done after an invention has been conceived, either by the patent attorney handling the filing or the examiner themselves. The premise that the number of citations made is somehow reflective of knowledge flow to the inventor is this flawed. It is clear that many pitfalls exist with the use of citation metrics [1] and improved methods of measuring invention impact and inventive knowledge flow over time are required. The method of this research accomplishes this by measuring invention relatedness using the cosine similarity of patent document vectors as a substitute for direct citation counts. Patent documents are uniquely suited for this technique because of the requirements set forth by the patent office. These rules require inventors to describe their inventions in great detail thus allowing us to select for core and likely rare inventive terms. The following section highlights important aspects of the USPTO patenting process which is relevant for both citation analysis and our methods.

1.4 Patent Requirements

In the following sections we will provide an overview of the various patent types and classification methods. This includes the reasoning behind selection of utility patents and the cooperative patent classification system as a research focus. It also discuss patent examination procedures related to the *general*, *description*, and *citation of prior art* patent application requirements as they are heavily dependent on written communication making them fertile ground for text mining. Lastly, there is a review of changes made to patent office rules related to both the patent exclusivity period and inventorship rights. This changes have impacted intellectual property filing behavior over time.

1.4.1 Patent Types

The USPTO grants four types of patents with varying degrees of protections.

- Reissue Patent - Issued to correct an error in an already issued patent
- Plant Patent - A distinct, invented or discovered asexually reproduced plant
- Design Patent - A new, original, and ornamental design embodied in or applied to an article of manufacture

- Utility Patent - Issued for a new and useful process, machine, manufacture, or composition of matter, or a new and useful improvement thereof
- Provisional Patent - A legal document that establishes a filing date but does not convert to a regular patent

Both reissue and plant patents are uncommon. A reissue patent is intended to correct for unintentional errors that can render a granted patent partially or completely invalid. The patentee is prevented from introducing any new inventive matter and the reissue patent covers only the remaining unexpired term of the original patent. Plant patents are intended to provide protection for asexually produced distinct plant varieties that cannot otherwise be made or manufactured. This includes plants, algae, or fungi that are considered mutants or hybrids. There have been less that 1500 plant patents issued per year in the Unites States from 1997 through 2018 [29]. The majority of patents issued in the United States are *Design Patents* and *Utility Patents*. Design patents are intended to protect artistic works and ornamental designs with a heavy dependence on pictures and drawings to communicate the scope of an invention. Design patents are less frequent than utility patents and are suited for inventions that are dependent on design aesthetics for success. The vast majority, in excess of 90%, of patents issued in the United States are utility patents [30]. The successful granting of a utility patent is dependent upon its functional purpose which must be expressed in written form, with diagrams and renderings used as supporting aids. This research focuses on utility patents because they represent the vast majority patents granted in the United States and because they are heavily dependent on written communication.

1.4.2 Provisional Patent

The provisional patent allows authors to file a patent before they have fully refined their invention. At the end of one year provisional filers are required to submit a full non-provisional patent application which allows them to keep their provisional filing date. They also have the option to abandon their provisional application and file a new application therby losing their provisional filing date. Provisional filers are not allowed to introduce new matter outside of what was described in their original application. The provisional patent is an excellent tool for inventors to ensure an early filing date. This research focuses on textual data contained within granted utility patents; however, provisional patents are intriguing, as they represent an earlier stage of inventive knowledge which could be potentially combined with published patent application textual data to study the evolution of inventive knowledge as it moves through the entire patent application process. This includes comparing how innovation flows out from the first public provisional disclosure to the actual granted patent as lag times between the two can exceed three years. This particular patent type is noted because of the future potential to include it's textual data to study the evolution of inventive knowledge flow with ML from the earliest filing date.

1.4.3 Patent Classification

Although there are numerous patent classification systems employed worldwide we will focus on the following systems they cover the vast majority of patents issued:

1. United States Patent Classification (USPC)
2. National Bureau of Economic Research (NBER)

3. International Patent Classification (IPC)
4. Cooperative Patent Classification (CPC)

1.4.3.1 United States Patent Classification (USPC)

The USPC is a legacy system for classifying all U.S. patents by subject matter that dates back to the mid 1800's. The USPC has historically been an evolving classification system that tries to keep up with new technologies. In some cases entirely new technology classes were created while others have been deleted and replaced with sub-classes when they became excessively large to manage. The fact that the USPC technology classes can change over time makes it difficult to compare the influence of specific technologies both within and outside specific categories. It also makes it difficult to compare intellectual property between countries. In June, 2015 the USPTO transitioned classification of utility patents from the USPC to the Cooperative Patent Classification System (CPC). For patents granted prior to this date both USPC & CPC classifications are listed.

1.4.3.2 National Bureau of Economic Research (NBER)

The NBER technological category system was developed by [17]as an overlay to the USPC system to make it more manageable for academic research. They consolidated the USPC classes into 37 economically meaningful technology subcategories and placed those under 6 main technology areas being:

1. Chemical(excluding Drugs)
2. Computers and Communications (C&C)

3. Drugs and Medical (D&M)
4. Electrical and Electronics (E&E)
5. Mechanical
6. Others

They then used these main categories to show interesting patterns and trends in U.S. patenting behavior over three decades. The work was intriguing as the authors proposed a number of constructed measures for assessing a patents influence over time. These measures built upon work conducted by [31], which showed that the number of citations a patent receives were directly related to the technological significance of the particular claimed invention. The NBER research showed a claimed invention varies in technological and economic importance both within and outside the main NBER assigned technology class and that simple patent counts were inherently limited.

1.4.3.3 International Patent Classification (IPC)

The IPC was established by the Strasbourg Agreement of 1971 to create an internationally uniform classification of patent documents. Its primary purpose was the establishment of an effective search tool for the retrieval of patent documents by intellectual property offices and other users worldwide. It has standard definitions to help clarify and overcome discrepancies due to differences in language and terminology among the numerous international patent offices [32], [33].

1.4.3.4 Cooperative Patent Classification (CPC)

The Cooperative Patent Classification (CPC) system was introduced in January 2013. It is a harmonized approach towards a global classification system that was jointly developed by the European Patent Office (EPO) and the United States Patent and Trademark Office (USPTO). It eliminates document reclassification between the two offices and allows for a single classification search result [34]. The CPC is a more specific and detailed version of the International Patent Classification (IPC) system as it is based upon its structure. It includes over 250,00 classes and represents the most detailed smallest denomination of all the English based patent classification systems [35].

1.4.4 Patent Rules

Patent examiners have requirements that they use when deciding whether an invention is patentable, the rules of which can be referenced in the Manual of Patent Examining Procedure, or MPEP [36]. There are a number of ways these rules are enforced. First, a determination is made by the patent examiner upon review of the patent application. If the patent application does not meet the requirements it will be rejected by the examiner via an *Office Action*, which is an official written document mailed to the applicant conveying the determination of patent application allow-ability. Applicants are then given an opportunity to respond to the examiners listed objections. This includes clarifications, adjustments, and proposals to address the examiners concerns. This correspondence is another opportunity space ripe for the application of NLP techniques because it goes to the heart of defining the boundaries of the invention in multidimensional space. Second, there are additional avenues by which third parties can challenge a patent: *pre-issuance submissions*, *post grant review*, and an *inter parte review* [37]. From a wider perspective these patent process requirements have created the core data records that enabled patents to serve as proxies for inventiveness, inventive knowledge flow, and technological change over time [7, 11, 38–42]. It is important to note that much of this historical research was based upon statistics applied to numerical data derived from patents. This historical research required significant resources in terms of human capital and time to acquire and transcribe the data to convert it into a structured format for research. Technology capable of processing large amounts of unstructured textual data was not available when much of this research was conducted.

1.4.4.1 General Requirements

The research of thesis focuses on the *general*, *description*, and *citation of prior art* requirements as they are heavily dependent on written communication making them fertile ground for text mining. Section 101 of the U.S. Patent Act establishes the general requirements for patent protection in the following sentence:

> *"Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefore, subject to the conditions and requirements of this title."* (4)

An invention is patentable if it meets the following four general requirements: (1) it must be *Statutory*, (2) *Novel*, (3) *Useful*, and (4) *Non-obvious.*

1.4.4.1.1 Statutory

The statutory requirement states that processes, machines, articles of manufacture, and compositions of matter are considered patentable. Examples of things that are not patentable are abstract ideas, laws of nature, and natural phenomena.

1.4.4.1.2 Novelty

To meet the novelty requirement an invention must be new and can not have been publicly disclosed more than a year prior to the application date. Public disclosure constitutes items such as a printed publication or previously published patent documents that encompass all features of the invention. Applicants, as part of the patent application process, will submit an information disclosure statement (IDS) that includes citations to any relevant work that their invention may have relied upon. Patent applicants will describe the differences between their subject invention and the cited work. In addition, examiners look for and cite previous patents that counter applicant claims of novelty. Prior public disclosures of inventive work are often referred to as *prior art*, which is defined as all the public information relevant to a patent's claim of originality at the time of application.

1.4.4.1.3 Useful

To meet the useful requirement an invention must have a beneficial purpose. A written statement explaining a market, societal, or human need along with an explanation of how the proposed invention addresses that need is usually enough to meet this requirement.

1.4.4.1.4 non-obvious

Lastly, the non-obvious requirement is based upon a determination that the invention would not be obvious to "...a person having ordinary skill in the art to which the claimed invention pertains". A patent examiner who is familiar with the technology of the claimed invention will typically make this determination. In doing so an examiner will attempt to find all the claimed features of a patent application from the prior art. If this can be done the examiner will reject the claimed invention as obvious. The best counter argument to a patent office rejection based on obviousness is to provide written evidence of an unanticipated or unexpected result related to a particular invention.

1.4.4.2 Description Requirements

Inventors who seek patent protection must meet the following description requirements set forth in Title 35 U.S. Code Section 112:

> *"The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same, and shall set forth the best mode contemplated by the inventor or joint inventor of carrying out the invention."* (5)

Thus inventors are required to describe in *writing* the *best mode* of their invention to such an extent that other practitioners of ordinary skill can *reproduce* it. These specifications are referred to as the written, best mode, and enablement requirements, respectively.

1.4.4.2.1 Written

The first requirement of the specification states that applicants must describe their inventions in writing. This requirement makes patents well suited for NLP techniques.

1.4.4.2.2 Best Mode

The second requirement states that applicants must describe the most preferred mode of their invention upon filing. This requirement is intended to prevent applicants from withholding optimal versions of their inventions from the public.

1.4.4.2.3 Reproducible

The third requirement is intended to ensure that applicants describe their inventions in full reproducible detail. Applicants must provide enough written detail for others of ordinary skill to reproduce their invention without undue experimentation. The description requirements provide fertile ground to mine the written text for inventive descriptors.

1.4.4.3 Citations of Prior Art

Patent applicants have a duty to cite any work they relied upon in the development of their invention. This includes any historical knowledge (e.g., previous patent documents or published scientific work, domestic or foreign) that they may have relied upon. This historical knowledge is known as prior art and its inclusion in the application helps the inventor meet the novelty requirement of their invention by describing how they improved upon or distinguished their work from others. Prior art citations are typically referred to as *backward citations*. Backward citations can also be made by the patent examiner when challenging the novelty of a newly submitted invention. When an examiner cites prior art, the inventor is required to explain how their claimed invention is a novel improvement over the subject cited work. Applicants and examiners can also reference non patent related prior art, for example academic papers, known as *non-patent references (NPRs)*. Citations that a patent receives over time are refereed to as *forward citations*. [43] showed that forward citations can be related to the value of the patented invention. Just as with academic papers, it is presumed that the number of forward citations an original patent receives over time is a proxy for knowledge flow and that a higher number of citations reflects a greater impact to society [19,26,31,44]. Unlike the academic world, where papers are published to share knowledge and the number of citations are considered an affirmation of the body of work's impact, patent applicants are not incentivized to cite prior knowledge as each citation adds an additional burden of proving novelty. The risk of not properly citing prior art can come after granting as it can be used to contest or invalidate the patent. The duty to disclose is limited to those substantively involved in the preparation or prosecution of the application and even then only the individual inventor is held to the standard as opposed to a corporate assignee [36]. Furthermore, the duty is described as a "duty of candor and good faith" by the inventor to share known prior art. Thus there is a high bar to prove that an inventor deliberately chose to not cite prior art out and the risk is directly linked to the value of the inventive knowledge at stack. A lack of proper citing among related patents is one example of the pitfalls of using citation statistics to measure technological significance and knowledge flow over time. Another pitfall of using citation statistics is truncation of forward citations as recent patents have inherently fewer forward citations than older patents [17,45–47]. Failure to cite and truncation are two examples of issues with relying solely upon citation data as the sole means to reveal relationships, impact, and ultimately inventive knowledge flow. Additional pitfalls exist as a result of significant changes made to USPTO rules. For example structural changes were made to the patent process to reduce abuse [48] and bring the U.S. into alignment with the World Intellectual Property Organization (WIPO). Two noteworthy changes are the adjustment of the patent exclusivity period in 1995 and a change to patent granting rights in 2013. These changes affected patent filing behavior and in doing so reduced our ability to compare innovation performance metrics over time [49–55].

[17] highlighted the impacts of Patent Office changes when they stated

"Indeed, the mode of operation of the Patent Office underwent significant changes in the past decades, thereby introducing a great deal of randomness (that have nothing to do with the actual timing of the inventions) into any patent time series dated by grant year." (6)

To understand how changes impacted patent filing behavior it is helpful to understand why changes to patent granting rights and patent exclusivity times were made.

1.4.5 Patent Exclusivity Period

Prior to June 8, 1995 the "limited time" exclusive right to utility inventions was 17 years from the patent grant date or 20 years from the original filing date, whichever was longer. In such a scenario inventors had a strong incentive to file patent applications early in the inventive process even though the review might take several years to complete [17]. This is because there was no penalty related to patent office processing time, additionally, applicants could list patent pending on their inventions as long as it was under review. The time difference between the original application filing date and the date at which the patent is granted is called *grant* *lag.* Prior to June 8, 1995 inventors would effectively maximize their patent exclusivity period whenever the grant lag exceeded three years. Thus filing behavior that would lead to longer process times would be incentivized. For example, larger, more complex patent applications were likely to take longer for the USPTO to process. Since patent application were also kept secret, those that took extended periods to process could be enforced upon granted against others who developed and launched products in the interim. This changed in 1995 when the exclusivity period of 17 years from patent grant date was dropped, leaving 20 years from the earliest patent filing date either in the U.S. or internationally. The change was made to align U.S. patent law with the World Trade Organization (WTO) and the Agreement on Trade-Related Aspects of Intellectual Property Rights or TRIPS [56]. As a result, the incentives around patent filing changed as the exclusivity period was now based upon the earliest filing date of the patent and any claimed *priority dates.* The priority date, sometimes called the "effective filing date", is the date used to establish the novelty and/or obviousness of a particular invention relative to other prior art [57]. Essentially patent seekers would now be penalized for slow office process times which would directly reduce their exclusivity period. To mitigate the impact of this change the USPTO offered term extensions for a limited time to compensate for administrative delays.

1.4.6 Patent Granting Rights

Historically U.S. patents have been granted on a first to invent (FTI) basis, meaning that as long as the original creator of the intellectual property can prove they conceived of the idea first and reduced it to practice they will be granted patent rights as the original inventor. *Reduced to practice* refers to the actual making of the invention in physical form, such as a working model or constructive form via patent application so that one with ordinary skill in the art could make the invention without undue experimentation. This effectively meant that original inventors were granted intellectual property rights over the comparable inventions of others even if they filed their patent application at a later date. This could also be abused if inventors elected to wait to submit their inventions until others successfully commercialized them. First-to-invent required inventors to prove they conceived of the idea first, usually by keeping a detailed record of inventive progress that would hold up to the legal standards required of evidentiary demonstrations. These invention records are typically dated, signed, and witnessed and could be used to prove a historical time-line of inventive progress without *invention abandonment* meaning that the inventor took care to keep their invention secret by not publishing or publicly disclosing their invention. This also means that a wealth of untapped creative textual data likely exists within corporate R&D institutions that is not part of the public domain. A noted benefit of the first-to-invent system was the level playing field it created between individual inventors and those represented by large corporations as inventorship was not dependent on available resources. The rest of the world uses a first-to-file concept which grants patent protection to the applicant with the earliest filing date. As a partial step to bring alignment with global intellectual property rules the USPTO moved to a first-inventor-to-file concept (a.k.a first-to-disclose) which provides a small grace period for applicants to get a filing date on record via a provisional patent application. The provisional patent application allows authors to file before they have fully refined their invention. At the end of one year they are required to submit a full non-provisional patent application while still keeping their provisional filing date. Provisional filers are not allowed to introduce new matter outside of what was described in their original application. Changes to granting rights from first-to-invent to first-to-file fundamentally impact filing behavior as inventors are now incentivized to submit their applications early in the invention process while experimentation is ongoing before the best mode is determined. The objective of the prior review was to establish a baseline understanding of some of the intricacies of the patent process in the hopes that 1) it becomes apparent that significant inventive knowledge is held withing the patent record and 2) the evolutionary nature of the process is marked with significant changes that influences metrics based solely on numerical data. The next section focuses on the application of ML techniques to the NBER citation data file. This initial research revealed that historical methods have been focused on numerical data and that an opportunity exists to exploit inventive language to reveal latent relationships between inventive documents without a formal need for a human to be make the connection via citation.

CHAPTER 2

RESEARCH: ML APPLIED TO NBER CITATION DATA

Preliminary research focused on the use of ML techniques applied to numerical structured data within patents. The premise was that ML could be used to predict inventive impact and inventive knowledge flow using citation data. It began by studying the National Bureau of Economic Research (NBER) Patent Citation Data File developed by [17]. The NBER data file includes structured data on over 3 million U.S. patents granted between 1963 and 1999 matched to Compustat [58] data of all firms traded in the U.S. stock market. The NBER data files include two types of data, *original* and *constructed*. A list of the original and constructed data fields can be referenced in Table I.

2.1 Original Fields

Many of the listed original data fields are self-explanatory, for example patent number, patent grant date and year, and geographic data related to inventor's country and state at the time of filing.

2.1.1 Assignee Identifier

The assignee identifier places patents into categories based upon whether the original inventor holds the legal rights of the patent or has assigned them to a corporate or governmental entity. The classes for assignee can be referenced in Table II.

TABLE I: NBER U.S. Patent Citations Data File Fields

	Field	**Name**	**Definition**
Original	PATENT	Patent number	The assigned patent number
	GYEAR	Grant year	The year in which the patent was granted
	GDATE	Grant date	The date the patent was granted
	APPYEAR	Application year	The year the application was filed
	COUNTRY	Country of first inventor	First inventor's filing country
	POSTATE	State of first inventor	First inventor's filing state
	ASSIGNEE	Assignee identifier	Assignee number to match to full assignee name
	ASSCODE	Assignee type	One of seven USPTO assignee categories
	CLAIMS	Number of claims (starting in 1975)	Total number of patent claims
	NCLASS	Main U.S. patent class	USPTO assigned technology class
Constructed	CAT	Technological category	NBER assigned technology area
	SUBCAT	Technological sub-category	NBER assigned sub-technology area
	CMADE	Number of citations made	Direct count of citations listed in the patent
	CRECEIVE	Number of citations received	Direct count of citations received by the patent
	RATIOCIT	Percent of citations	Percent of citations to patents granted since 1963
	GENERAL	Measure of generality	Technology spread of a patent's forward citations
	ORGINAL	Measure of originality	Technology spread of a patent's backward citations
	FWDAPLAG	Mean forward citation lag	Mean time between patent and forward citations
	BCKGTLAG	Mean backwards citation lag	Mean time between patent and backward citations
	SELFCTUB	Percentage of self-citation made	Percentage of citations made to own work

TABLE II: Raw Assignee Category Type

Code	Name
1	Unassigned (has not assigned rights beyond inventor)
2	US Company or Corporation
3	Foreign Company or Corporation
4	US Individual
5	Foreign Individual
6	US Government
7	Foreign Government
8	Country Government
9	State Government (US)

2.1.2 Number of Claims

Every patent requires a set of claims, located at the end of patent, that are a numbered list that clearly identify what exactly the inventor wants exclusive rights to practice. It is generally understood that the less claims a patent has the broader it's potential impact. As the number of claims are increased the more nuanced the invention becomes as it tries to differentiate itself from the prior art. Claims are usually designated as dependent and independent, which are not distinguished in this data set.

2.1.3 Main U.S. Patent Class

The main U.S. patent classification is based on the USPC system which consists of two components or classes which are then combined into a class/subclass symbolic pair. The beginning part of the pair represents the major technology area that a particular patent falls within. The second half of the pair further delineates the unique features of the technology within the scope of the main class.

Example 1. *Current U.S. Class: 106/778; 106/270*

In Example 2.1.3 the class symbol 106 represents *(Compositions: coating or plastic)* while the subclass 778 means *(with organic material)* and subclass 270 is *(Wax, bituminous material or tarry residue containing)*. A given patent can have numerous classifications assigned to it. When more than one classification is given they are listed by the order in which each technology is presented in the patent application. There is no limit on the number of main classes that can be assigned. For each patent only one symbol is selected as the primary classification and it is emphasized in bold font. An example of the primary main class is listed under section 52 of U.S. Patent 6,673,144B2 shown in Figure 7.

2.2 Constructed Fields

The authors of the NBER data also created a number of constructed data fields. These include the *technological category* as well as a set of citation-based calculated metrics, including *number of citations made & received*, *backward & forward citation lag*, *percent citation total and self-citation*, and *measure of generality and originality.*

2.2.1 Technological Category

The *technological category* matches the invention to one of six higher level NBER-defined technology areas (Chemical, Computer & Communications, Drugs & Medical, Electrical & Electronic, Mechanical, and Others). These higher level NBER technology categories are based upon 36 technological sub-categories used under the *technology sub-category* name. In turn, these 36 technological sub-categories were derived by aggregation of 400 subject matter classes from the original *main U.S. patent class.*

2.2.2 Citations Made, Received, and Lag

Patents require that inventors in good faith list any historical knowledge that they are aware of at the time of filing. This historical knowledge is known as prior art and it helps the inventor distinguish the novelty of their inventions over past inventions. Just as in academic papers it is presumed that the number of citations received is a proxy for knowledge flow and that a higher number of citations reflects the overall inventions impact to society. *Number of citations made* is a direct count of the backward citations made by a patent, while *number of citations received* reflects the total count of forward citations that a patent receives over time. *Percent of citations* is calculated by taking the number of citations made to patents granted since 1963 and dividing by the total number of citations made. *Mean backwards citation lag* is the average time difference between the application year or grant year of the citing patent and year of the patent cited. *Mean forward citation lag* is the average time between the application or grant date of the originating patent and those that cite it. Note that mean forward citation lag suffers from truncation and for the data set at hand the upper forward time limit was 24 years. The mean citation lag measures are of particular interest because they attempt to answer questions related to the diffusion of inventive knowledge.

2.2.3 Measures of Generality and Originality

Measure of generality and originality is based upon the work conducted by [31]. The generality index attempts to gauge the general technological impact of a patent based upon the spread of technical categories associated with its forward citations using the following equation:

$$G_i = 1 - \sum_{j=1}^{J} (\frac{N_{ij}}{N_i})^2 \tag{2.1}$$

where N_i represents the total number of forward citations to the focal patent i and N_{ij} is the number of forward citations received in patent technology class j. Dividing N_{ij} by N_i gives the percentage of citations received by patent i that belong to patent class j, out of J total patent classes. The sum of the squared percentages is based upon the Herfindahl-Hirschman concentration index (HHI) which is used to measure market concentration within a given industry with respect to the potential for monopolistic behavior [59–61]. The result of squaring assures that the sum is not negative while at the same time emphasizing larger differences in concentration. The generality index takes values between zero and one. Patents that receive a high number of citations across differing technology categories will also have a higher generality score compared to citations concentrated in fewer technological categories. The same technique is applied to study patent originality by substitution of backward citation counts for forward citations. If a patent cites a narrower set of patent technology categories its originality score will be lower. Lastly, the *percentage of self-citations made* is a count of backward citations made by a patent to its current assignee divided by the total number of backward citations made.

Self-citations is an important measure when considering the overall impact of a given technology. It is not uncommon for patentee's to cite previous work as part of an overall patent strategy to create a moat around a firm's intellectual property, resulting in an inflated forward citation count.

2.3 Visual Inspection of NBER Data

Statistical, pattern, and association rule analysis was conducted on the NBER data file. On average the USPTO issues over 79,000 patents annually with a median of 71,000 patents between the years 1963 and 1999. A scatter plot based upon the NBER data file of the number of patents issued from 1963 through 1999 is shown in Figure 1. Visual inspection of the plot

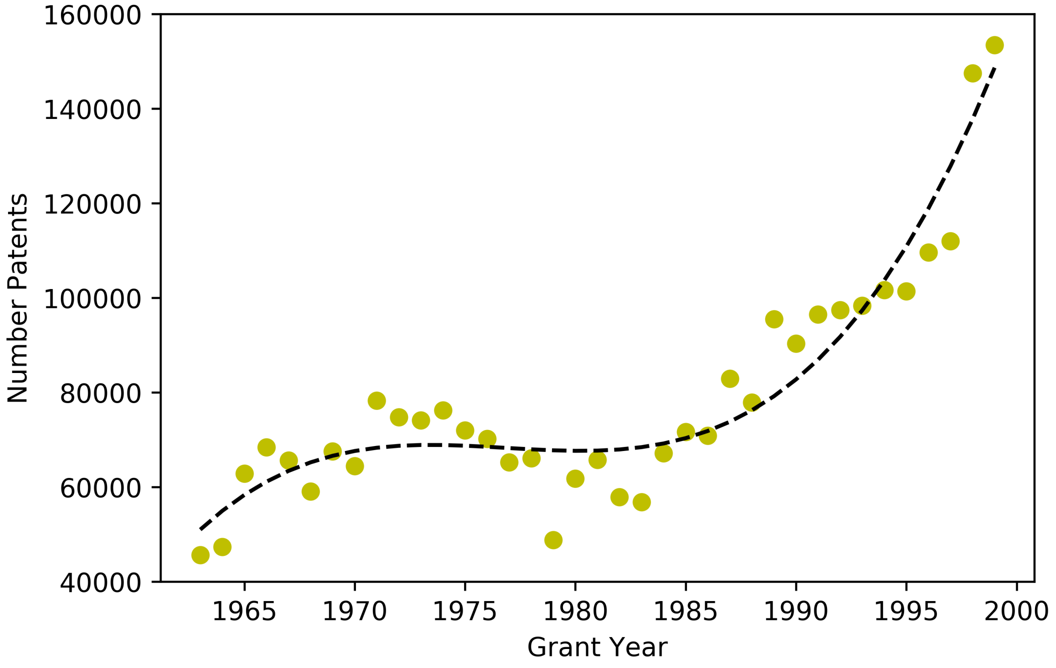


Figure 1: USPTO Issued Patents (1963 - 1999)

reveals several interesting patterns: (i) there is an overall steady growth trend of patents issued annually; (ii) there was a slight growth trend in patents granted from 1962 to 1970; (iii) there is a distinct leveling off period between 1970 & 1985; (iv) there is a significant spike after 1995 when the average number of patents dramatically increased to approximately 150,000 annually; (v) there appear to be outliers with high data points in 1999 and a low value in 1978..

2.4 Statistical Analysis

Statistical analysis on the entire NBER patent data set was conducted for the purpose of discovering potentially interesting patterns that might lead to insights about patent impact and inventive knowledge flow. The code for this analysis can be referenced in Appendix L. Some notable insights include: (i) a significant proportion (47%) of patents are filed by U.S. Corporations, with the second largest category being non-U.S. Corporations at 32%, thus 79% of all patents filed in the U.S. are by corporations; (ii) the average lag between the time a patent is filed and granted is two years.. A histogram of the number of patents granted each year can be referenced in Figure Figure 2. The histogram ranges from a low of 45,679 in 1963 to a high of 153,486 in 1999. The blue vertical line represents the mean of 79,024 patents while the orange vertical line represents the median of 71,661.

2.4.1 Proportion Analysis

Table III represents a simple proportion study based upon the relative percentage of each patent assignee type. The proportion of U. S. non-government organizations (mostly corporations) who filed patents from 1963 through 1999 was 47.2%. This was derived by setting the assignee identifier field code to the value of 2 (U. S. non-government organizations) and dividing by the

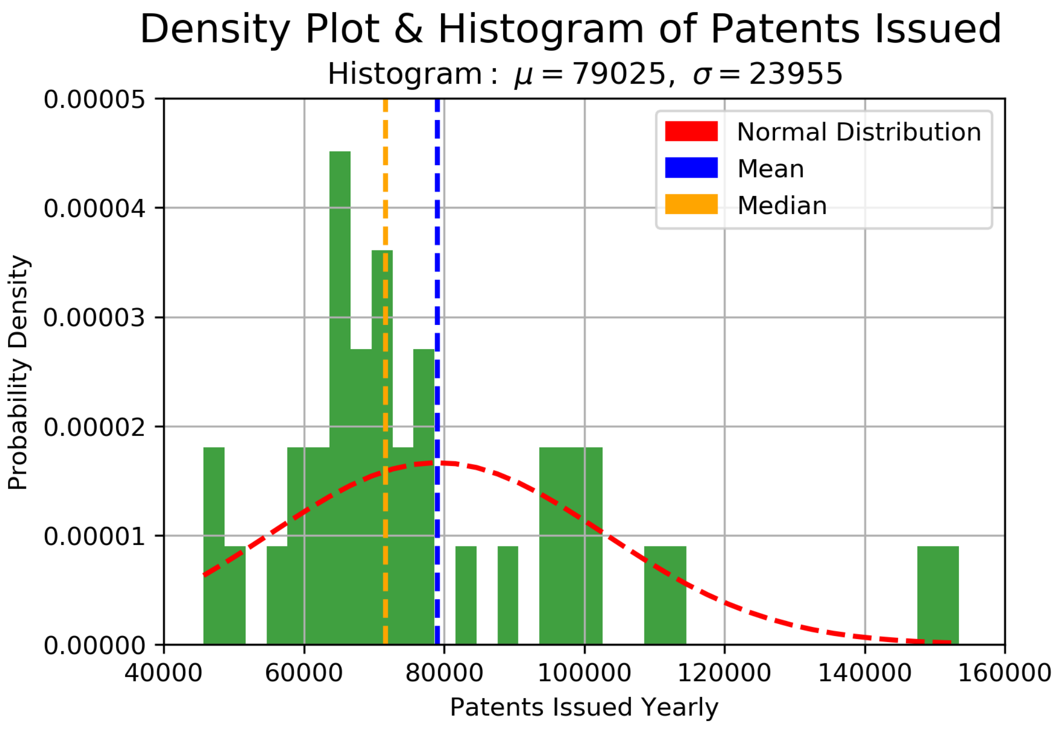


Figure 2: Density plot and histogram of USPTO issued patents (1963 - 1999)

total number of patents. Another interesting fact from the data was the relatively high percentage (31.3%) of corporations outside the U.S. who received patents. Combined corporations represent almost 79% of all patents award between the years of 1963 and 1999. Table IV represents a proportion study based upon the relative percentage of each patent technology category. What is interesting from this data is the fact that Chemical and Mechanical categories represent a significant portion of the number of granted patents. The Others category includes agriculture, amusement, apparel, furniture, and miscellaneous.

2.5 Association and Pattern Analysis

Association and pattern analysis techniques were applied to the data set to investigate whether a particular technology industry took longer than the 2 year average lag to be granted

TABLE III: NBER proportion of patents based on assignee

Type	**Assignee**	**Percent**
1	Unassigned (has not assigned rights beyond inventor)	18.4%
2	US Company or Corporation	47.2%
3	Foreign Company or Corporation	31.2%
4	US Individual	0.8 %
5	Foreign Individual	0.3%
6	US Government	1.7%
7	Foreign Government	0.4%

TABLE IV: NBER proportion of patents based on assignee

Type	**Technical Category**	**Percent**
1	Chemical	20.8%
2	Computers & Communications	9.9%
3	Drugs & Medical	7.0%
4	Electrical & Electronic	17.1%
5	Mechanical	23.3%
6	Others	21.9%

the results of which can be referenced in Table Table V. The objective of this exploration was to understand what influences grant lag, and in turn, knowledge flow dissemination. Analysis

TABLE V: Association & Pattern Measures Applied to the NBER Data

Rule A	Rule B	Support	Confidence	Lift	All Conf	All Conf	Kulczynski	Cosine
Patent lag > 2 years	Chemical	4.9%	23.8%	1.15	0.24	0.24	0.24	0.24
	Computers & Communications	3.2%	15.4%	1.55	0.15	0.32	0.24	0.22
	Drugs & Medical	1.9%	9.1%	1.31	0.09	0.27	0.18	0.16
	Electrical & Electronic	3.3%	16.3%	0.95	0.16	0.20	0.18	0.18
	Mechanical	3.9%	18.9%	0.81	0.17	0.19	0.18	0.18
	Others	3.4%	16.5%	0.75	0.15	0.16	0.16	0.16

started by generating a patent lag field that took the difference between the patent application and grant dates for each patent from the set of 2.9 million. The average patent lag was determined to be approximately 2 years. A single association rule was tested to see what the support, confidence, and lift was of technology categories associated with any particular industry taking longer than the average to have their patents granted. These equations can be referenced in Equation Equation 2.2, Equation Equation 2.3, and Equation Equation 2.4. In this analysis A represents an item-set of patents that exceed 2 years to grant with B being a select technology category. The association rule antecedent A given the consequent B is $A \Rightarrow B$ and the *Support* of A with respect to L, our set of 2.9 million lag records, is defined as the proportion of lags l in the data-set which contain A.

$$supp(A) = \frac{|\{l \in L; A \subseteq l\}|}{|L|} \tag{2.2}$$

In Table V the support column represents the fraction of lag records that contained both the technology category and a patent lag of greater than two years. *Confidence* is an indication of how often the rule is likely to be true:

$$conf(A \Rightarrow B) = supp(A \cup B)/supp(A) \tag{2.3}$$

In Table V the confidence column represents among records containing A the fraction of rows that also contain B or the conditional probability of B given A. *Lift* is the ratio of confidence to support or the ratio of observed support to the support of A and B occurring independently. The lift value is considered an interestingness measure in that it helps us understand the relative strength of a rule. If the lift is > 1 there is a positive correlation while < 1 is negative. A lift value of zero has no correlation.

$$lift(A \Rightarrow B) = \frac{A \cup B}{supp(A) \times supp(B)} \tag{2.4}$$

Table V also includes additional interestingness pattern evaluation measures including *Kulczynski* which is represented by the following equation:

$$Kulczynski = \frac{1}{2}(P(A|B) + P(B|A)) \tag{2.5}$$

A Kulczynski score near 0 or 1 indicates that we have an interesting rule that is either negatively or positively associated respectively while being near 0.5 can be interpreted as uninteresting.

In addition to Kulczynski, other measures of interestingness include all-confidence, maximum-confidence, and cosine (IS) [62–64]. The association analysis indicated that the technology category with the highest support was the Chemistry category. In this category 4.9% of all patents took over 2 years to grant. Furthermore, of the patents that did take over 2 years to issue there is a 23.8% chance that they will also be classified as Chemical. Our lift indicates a slightly positive correlation being 1.15 times more likely to be classified as Chemical. Based upon this analysis it appears that technology category can influence patent lag and that patents associated with certain technology fields, like chemistry, were more likely to have a greater patent lag than other technology fields. Since patent lag directly influences inventive knowledge flow we wanted to understand what independent variables had the most influence on NBER data set.

2.6 Supervised Machine Learning

ML techniques were applied to study what independent variables were driving variability within the NBER data set. Analysis techniques included principal component, decision trees, and artificial neural networks (ANN).

2.6.1 Principal Component Analysis

Imputation was conducted on the data set to address missing data values with substitute values. This is required because some ML based statistical methods are unable to process missing or corrupt values, for example, those that result from encoding issues. There are many different algorithmic methods to perform imputation. The particular technique used for this analysis is known as *mean substitution.* Mean substitution corrects for missing numerical data by subsisting the mean of the variable from the set which has the nice benefit of not changing the overall sample mean. A random sample of 5000 patent records was extracted from the NBER data. Independent variables for patent grant year, grant date, application year, sub patent category, assignment code, main patent class, citations received, and generality were defined. Principal component analysis indicated that patent grant year was responsible for 98% of the data set's overall variability.

2.6.2 Decision Tree

A random sample of 5000 patent records were extracted from the NBER data set. Independent variables for patent grant year, grant date, application year, sub category, assignment code, main patent class, and country of origin were defined. Country of origin data was converted from categorical to numerical using a label-encoder function (e.g. U.S. = 22). The dependent variable was set to technical category as listed in Table IV. Decision tree classification techniques were able to successfully predict the NBER technological category with greater than 96%)accuracy with the main splitting criterion being the application year of the patent (see Figure 3. This made intuitive sense as the year in which an invention was developed is strongly influenced by the technology available at the time. The code for this analysis can be referenced in Appendix N

2.6.3 Artificial Neural Network

A random selection of 5000 records was used for this analysis. Input variables of grant year, grant date, application year, sub category, assignee type, patent class, and patent country were selected. The dependent variable of patent category was selected. A python for loop was used

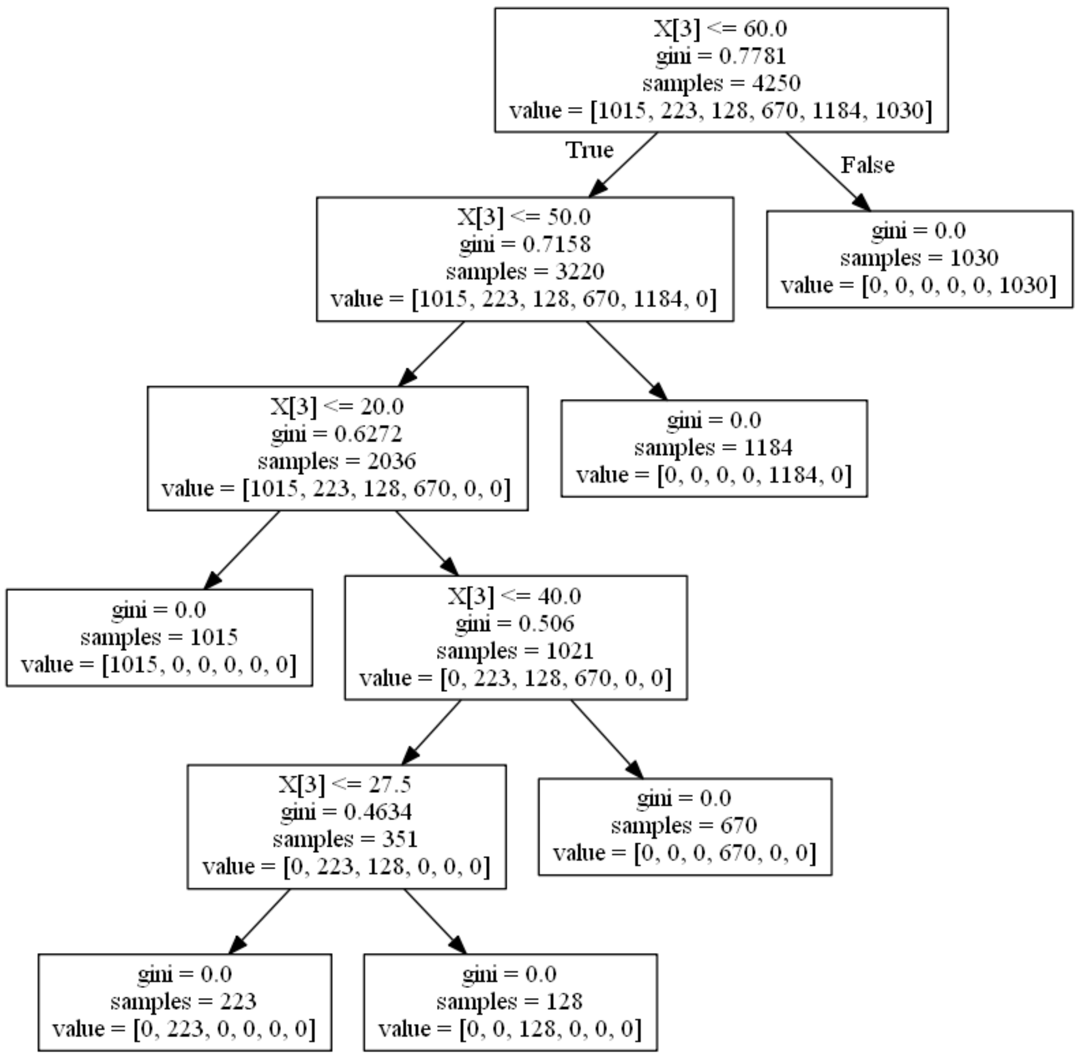

X[3] <= 60.0
gini = 0.7781
samples = 4250
value = [1015, 223, 128, 670, 1184, 1030]
True
False
X[3] <= 50.0
gini = 0.7158
samples = 3220
value = [1015, 223, 128, 670, 1184, 0]
gini = 0.0
samples = 1030
value = [0, 0, 0, 0, 0, 1030]
X[3] <= 20.0
gini = 0.6272
samples = 2036
value = [1015, 223, 128, 670, 0, 0]
gini = 0.0
samples = 1184
value = [0, 0, 0, 0, 1184, 0]
gini = 0.0
samples = 1015
value = [1015, 0, 0, 0, 0, 0]
X[3] <= 40.0
gini = 0.506
samples = 1021
value = [0, 223, 128, 670, 0, 0]
X[3] <= 27.5
gini = 0.4634
samples = 351
value = [0, 223, 128, 0, 0, 0]
gini = 0.0
samples = 670
value = [0, 0, 0, 670, 0, 0]
gini = 0.0
samples = 223
value = [0, 223, 0, 0, 0, 0]
gini = 0.0
samples = 128
value = [0, 0, 128, 0, 0, 0]

Figure 3: Decision Tree prediction of patent technical category using NBER patent data (1963 - 1999)

to test the impact of the neurode number (up to 50) of the first hidden layer on the accuracy of the classification model. For this particular run the accuracy of predicting the assigned patent category was 57% which can be seen in Figure 4. The code for this analysis can be referenced in Appendix M.

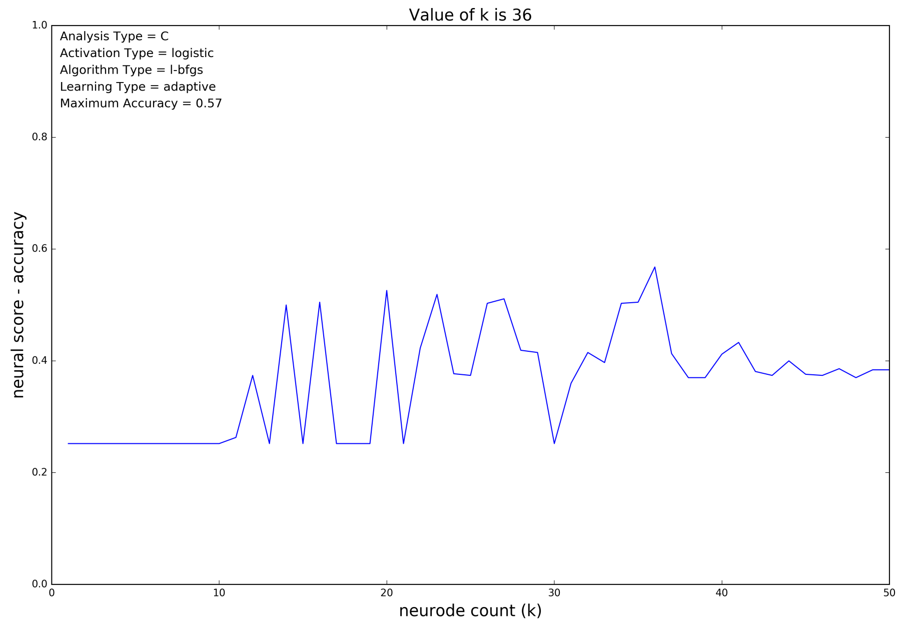


Figure 4: Neural network prediction of patent technical category using NBER patent data (1963 - 1999)

2.7 Preliminary Research Insights

Insights gained by studying the NBER citation data file raised questions as to whether patent numerical data was the best tool to study inventiveness and knowledge flow. Further research led to a survey review by [1] which highlighted many citation analysis drawbacks over the last two decades. They specifically call out four pitfall areas associated with the use of patent citation data. These include: (i) *office effects* caused by different “prior art” requirements between patent offices worldwide; (ii) *time and technology field effects* such as citation truncation, citation inflation over time, and citation count differences between technology fields; (iii) *examiner effects* related to variations in examiner tenure which influence application processing time leading to variation in patent and citation lag as well as potential examiner bias towards foreign applications; and lastly, (iv) *strategic effects* caused by changes in applicants' tendency to cite or not cite prior art, which is a strategic decision influenced by changes to the patent process over time. The research of this thesis raised a number of additional concerns, most of which can be classified under the headings listed by [1] and one additional category of *computational effects*.

2.7.1 Computational Effects

Computational effects are related to increases in computer processing speed, digital storage, and the development of advanced software capable of conducting complex statistical analysis. These effects allow us to process large complex data sets that were not possible before, data sets that are likely to contain interesting patterns and latent knowledge. The unstructured textual data that exists within the USPTO patent corpus is a prime example. Below we discuss additional considerations for office, examiner, and strategic effects.

2.7.2 Office effects

The tendency for applicants to use obfuscating language, whether intentional or not [65], puts an additional burden on the patent examiner to establish credible prior art via keyword searches which are entirely based on the language inventors use to describe their inventions.

It seems reasonable to conclude that a more representative number of citations would exist if applicants held to the spirit of listing *prior art*, while also using clearer language to describe their claimed inventions per the written description requirement. The use of obscure and confusing language also makes it difficult for others to reproduce or improve upon inventions. Later in this thesis Figure Figure 11 we propose methods to mitigate this impact through the use of a specialized *'Legal Jargon'* dictionary which can be used to filter for obfuscating language.

2.7.3 Examiner effects

[21] showed that examiner experience impacts patent outcomes. Human resource policies on patent examiner performance measurement known as the Patent Examiner Performance Appraisal Plan, or PAP, may also influence patent examiner behavior and outcomes. The PAP is based on a combination of quality, productivity, and timeliness measures for examiners. In the United States one specific measure of examiner performance is the production unit time requirement, which is the count of patents processed per unit time for a specific patent technology category. The requirement is based on a two-week cycle, with no allowance for illness or holidays. An unintended consequence of this performance metric is an incentive for examiners to split patents. A patent application can be split when there is a clear lack of unity, i.e., when the patent describes more than one invention. The examiner benefits because the split increases the patent count per unit of time. The division of applications also benefits the applicant because it reduces the filing costs incurred up until the point of division. The outcome is a trend towards longer applications with more claims. The increase in claims is not a function of more inventive steps as much as an increase in the number of inventions per application.

The economic influences have not gone unnoticed by legal firms who specialize in intellectual property filing, and it is common to see fees tied to the number of requested independent and dependent claims. The examiner performance measurement system was revised in 2010 to help remove administrative barriers to patent grants [66] as well as to facilitate identification of patent application issues as early in the examination process as possible [67].

2.7.4 Strategic effects

The evolving legal understanding of the written description requirement has resulted in applicants using more general language with less reliance upon specifically stated scientific based breakthroughs [68]. Changes in patent rights, for example the transition from first-to-invent (FTI) to a first-inventor-to-file (FITF) are noteworthy because they directly effect patent filing behavior and reduce our ability to compare innovation performance over time using citation analysis methods [49–55].

2.8 Results from examination of the NBER data file

Preliminary research using the NBER data revealed that there are many potential issues with the use of historical patent metrics derived from structured numerical data to reveal latent relationships between inventions. These concerns include statistical anomalies in the data set as well as fundamental structural changes in the patent application process over time that impact filing and citation behavior. Examples include, in addition to those outlined by [1], changes to the examiner performance measurement system, changes to patent office filing requirements, and changes to patent data processing capability, including technological advances that have enhanced our ability to transform massive amounts of unstructured patent textual data into structured data for ML. These insights inspired a refocusing of research efforts on the conversion of unstructured patent textual data, via NLP techniques, into structured data for the application of ML. The ultimate objective remains to provide improved methods of discovering latent relationships, overall inventiveness, and how inventive knowledge is disseminated.

CHAPTER 3

RESEARCH: ML APPLIED TO USPTO TEXTUAL DATA

The inadequacy of analysis based solely on structured numerical data such as patent citations to generate actionable knowledge related to the measurement of inventiveness and inventive knowledge flow leads us to consider the unstructured textual data contained within patents. In this research, a methodology is developed to apply text mining to unstructured patent data while adding specific steps to take advantage of the written application requirements and language that are unique to the patenting process. The following sections present research conducted to date using both patent textual and numerical data.

3.1 PatentsView Data Tables

In 2015 the USPTO and partners launched **PatentsView** [69,70]. The PatentsView Data Tables and the PatentsView Visual Analysis Platform were made available specifically for research purposes to increase the "value, utility, and transparency of U.S. patent data". The data set contains over 6 million detailed records representing all U.S. patenting activity since 1976. The records include the NBER technology categories that were discussed earlier in this thesis, along with unstructured textual data located in the patent title, abstract, and claims. PatentsView provides a data table dictionary which includes detailed information on each table, including corresponding fields and relationships [71]. The individual data files are in tab delimited format which allows them to be downloaded and imported into native environments.

Thirty-five database tables, in excess of 26 gigabytes, were downloaded for analysis using the Python programming language [72]. Citation and text mining analysis were conducted using the patent, assignee, citation, claim, CPC, and application data tables which are detailed in the following sections.

3.1.1 Patent Table

The data fields shown in Table VI include patent numbers, dates, abstracts, and titles that were extracted via the code listed in Appendix A. The patent number is the official number assigned to a patent by the USPTO upon granting and is the relational key used to match records from other tables within the database. The date field represents the official patent grant date assigned by the USPTO at the time of issuance. Unstructured textual data exists

TABLE VI: PatentsView Patent Table

Table	**Field Name**	**Definition**	**Type**
patent	id	patent that this record corresponds to	varchar(20)
	type	one of eight patent types (e.g. "utility", "design", etc.)	varchar(100)
	number	patent number	varchar(64)
	country	country in which patent was granted (always US)	varchar(20)
	date	date when patent was granted	date
	abstract	abstract text of patent	text
	title	title of patent	text
	kind	document kind codes[a]	varchar(10)
	num_claims	number of claims	int(11)
	filename	name of the raw data file where patent information is parsed from	varchar(120)

within the title and abstract fields. Patent abstracts are intended to summarize the invention as succinctly as possible, typically in fewer than 150 words, using less complex legal terminology than is normally found in the body of the document. These fields can be seen in Figure 5 which shows the first page of an example United States Patent with the patent number, date, title, and abstract fields outlined in red [73].

3.1.2 Claims Table

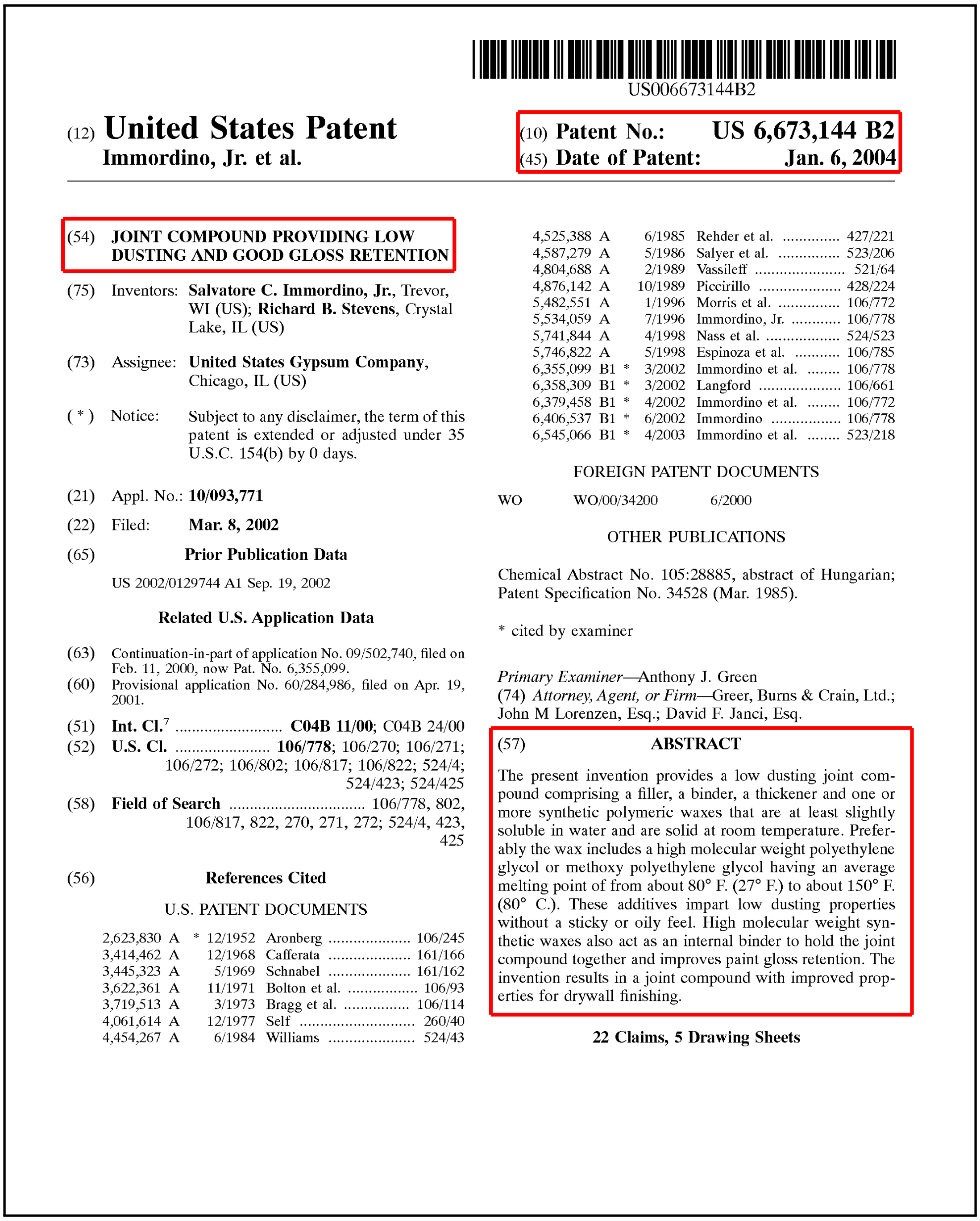
The claims table (Table VII) contains unstructured textual data intended to define the exclusivity zone of an invention by delineating, using an ordered list of detailed text statements, what inventors are seeking to prevent others from making or selling. Patent claims are considered the heart of an invention which is why they are one of the first areas studied by domain experts when trying to understand key inventive elements.

TABLE VII: PatentsView Claims Table

Table	Field Name	Definition	Type
claim	uuid	unique id	varchar(36)
	patent_id	patent number	varchar(20)
	text	claim text	text
	dependent	sequence number of the dependent claim, -1 if independent	int
	sequence	order in which claims appear in patent file	int

3.1.3 Raw Assignee Table

The raw assignee table (Table VIII) provides information on the rights holder of the patent at the time of granting which includes the first and last name of the assignee for an individual and/or the organization name for a corporation. This research uses the organization field from the raw assignee table to select for companies that compete within the building materials technology space via the code listed in Appendix C. There is an alternate disambiguated form of



US006673144B2

(12) **United States Patent**
Immordino, Jr. et al.

(10) **Patent No.: US 6,673,144 B2**
(45) **Date of Patent: Jan. 6, 2004**

(54) **JOINT COMPOUND PROVIDING LOW DUSTING AND GOOD GLOSS RETENTION**

(75) Inventors: **Salvatore C. Immordino, Jr.**, Trevor, WI (US); **Richard B. Stevens**, Crystal Lake, IL (US)

(73) Assignee: **United States Gypsum Company**, Chicago, IL (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **10/093,771**

(22) Filed: **Mar. 8, 2002**

(65) **Prior Publication Data**

US 2002/0129744 A1 Sep. 19, 2002

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/502,740, filed on Feb. 11, 2000, now Pat. No. 6,355,099.
(60) Provisional application No. 60/284,986, filed on Apr. 19, 2001.

(51) **Int. Cl.**^{7} **C04B 11/00**; C04B 24/00
(52) **U.S. Cl.** **106/778**; 106/270; 106/271; 106/272; 106/802; 106/817; 106/822; 524/4; 524/423; 524/425
(58) **Field of Search** 106/778, 802, 106/817, 822, 270, 271, 272; 524/4, 423, 425

(56) **References Cited**

U.S. PATENT DOCUMENTS

2,623,830 A	*	12/1952	Aronberg		106/245
3,414,462 A		12/1968	Cafferata		161/166
3,445,323 A		5/1969	Schnabel		161/162
3,622,361 A		11/1971	Bolton et al.		106/93
3,719,513 A		3/1973	Bragg et al.		106/114
4,061,614 A		12/1977	Self		260/40
4,454,267 A		6/1984	Williams		524/43
4,525,388 A		6/1985	Rehder et al.		427/221
4,587,279 A		5/1986	Salyer et al.		523/206
4,804,688 A		2/1989	Vassileff		521/64
4,876,142 A		10/1989	Piccirillo		428/224
5,482,551 A		1/1996	Morris et al.		106/772
5,534,059 A		7/1996	Immordino, Jr.		106/778
5,741,844 A		4/1998	Nass et al.		524/523
5,746,822 A		5/1998	Espinoza et al.		106/785
6,355,099 B1	*	3/2002	Immordino et al.		106/778
6,358,309 B1	*	3/2002	Langford		106/661
6,379,458 B1	*	4/2002	Immordino et al.		106/772
6,406,537 B1	*	6/2002	Immordino		106/778
6,545,066 B1	*	4/2003	Immordino et al.		523/218

FOREIGN PATENT DOCUMENTS

WO	WO/00/34200	6/2000

OTHER PUBLICATIONS

Chemical Abstract No. 105:28885, abstract of Hungarian; Patent Specification No. 34528 (Mar. 1985).

* cited by examiner

Primary Examiner—Anthony J. Green
(74) *Attorney, Agent, or Firm*—Greer, Burns & Crain, Ltd.; John M Lorenzen, Esq.; David F. Janci, Esq.

(57) **ABSTRACT**

The present invention provides a low dusting joint compound comprising a filler, a binder, a thickener and one or more synthetic polymeric waxes that are at least slightly soluble in water and are solid at room temperature. Preferably the wax includes a high molecular weight polyethylene glycol or methoxy polyethylene glycol having an average melting point of from about 80° F. (27° F.) to about 150° F. (80° C.). These additives impart low dusting properties without a sticky or oily feel. High molecular weight synthetic waxes also act as an internal binder to hold the joint compound together and improves paint gloss retention. The invention results in a joint compound with improved properties for drywall finishing.

22 Claims, 5 Drawing Sheets

Figure 5: U.S. Patent 6,673,144

the assignee data table within the data set that reflects current organization names; however, it was not used because it is not persistent across database updates. Instead the set of companies were disambiguated use the code listed Appendix C.

TABLE VIII: PatentsView Assignee Table

Table	Field Name	Definition	Type
rawassignee	uuid	unique id	varchar(36)
	patent_id	patent number	varchar(20)
	assignee_id	unique assignee ID generated by the disambiguation algorithm	varchar(36)
	rawlocation_id	assignee's location	varchar(128)
	type	assignee type	int(4)
	name_first	first name, if assignee is individual	varchar(64)
	name_last	last name, if assignee is individual	varchar(64)
	organization	organization name if assignee is an organization	varchar(256)
	sequence	order in which assignee appears in patent file	int(11)

3.1.4 Cooperative Patent Classification Table

The CPC table (Table IX) contains four different patent classification systems that were previously reviewed in section 1.4.3. The CPC system, introduced in January 2013, was selected for use in this analysis for two reasons. First, it is a harmonized approach towards global patent classification jointly developed by the European Patent Office (EPO) and the USPTO to eliminate reclassification and simplify patent search results through the use of a single classification system. [34, 74]. Second, it has reached critical mass as many industrialized nations have adopted the CPC with over 50 million patent documents classified [35]. The main CPC section was selected along with the patent_id via the code listed in Appendix F. This research used the highest level main classification of the CPC, which is detailed in Table X.

TABLE IX: PatentsView CPC Table

Table	Field Name	Definition	Type
cpc_current	uuid	unique id	varchar(36)
	patent_id	patent number	varchar(20)
	section_id	cpc section*	varchar(10)
	subsection_id	cpc subsection id*	varchar(20)
	group_id	cpc group id*	varchar(20)
	subgroup_id	cpc subgroup id*	varchar(20)
	category	cpc category (primary or additional)	varchar(36)
	sequence	order in which cpc class appears in patent file	int(11)

* http://www.uspto.gov/web/patents/classification/cpc.html

TABLE X: Cooperative Patent Classification

Code	Name
A	Human Necessities
B	Performing Operations; Transporting
C	Chemistry; Metallurgy
D	Textiles; Paper
E	Fixed Constructions
F	Mechanical Engineering; Lighting; Heating; Weapons; Blasting Engines or Pumps
G	Physics
H	Electricity
Y	General Tagging of New Technological Developments

3.1.5 Citation Table

The citation table (Table XI) matches a patent number to all the corresponding backward citations listed at the time of patent granting. This includes notations as to whether a specific citation was made by the examiner or applicant. The sequence field gives the order in which citations were made. This research used the patent_id and citation_id fields to locate forward citations for each patent within the building material companies subset using the programming code listed in Appendix D.

TABLE XI: PatentsView Citation Tables

Table	Field Name	Definition	Type
uspatentcitation	uuid	unique id	varchar(36)
	patent_id	patent number	varchar(20)
	citation_id	citing patent number	varchar(20)
	date	patent citing date	date
	name	name of cited record	varchar(64)
	kind	document kind codes[a]	varchar(10)
	country	country cited patent was granted (always US)	varchar(10)
	category	who cited the patent (examiner, applicant, other etc)	varchar(20)
	sequence	order in which this reference is cited by select patent	int(11)

[a] http://www.uspto.gov/learning-and-resources/support-centers/electronic-business-center/kind-codes-included-uspto-patent

3.1.6 Application Table

The application table (Table XII) contains the patent filing date information for every patent. This is the official start time for the 20 year term of exclusive rights to practice the claimed invention. It is also a key input when attempting to measure inventiveness and inventive knowledge flow as it represents the earliest official listing by the inventor of prior art via backward citations. The final set of citations listed at the time of patent granting was used for this analysis, however, the patent application date was used in place of grant date as it is a better reflection of when the inventive knowledge was first created.

TABLE XII: PatentsView Application Table

Table	Field Name	Definition	Type
application	id	application id assigned by USPTO	varchar(36)
	patent_id	patent number	varchar(20)
	series_code	application series, ”D” for some designs*	varchar(20)
	number	unique application identifying number	varchar(64)
	country	country the application was filed in	varchar(20)
	date	date of application filing	date

* http://www.uspto.gov/web/offices/ac/ido/oeip/taf/filingyr.htm

3.2 Invention Impact: Intra-Citation Visualization

Figure 6 represents a holistic view of patent intellectual property filing activity for the set of building companies that are shown on the left hand side of the figure for the years 1976 through 2015. The code for this analysis can be referenced in Appendix H.

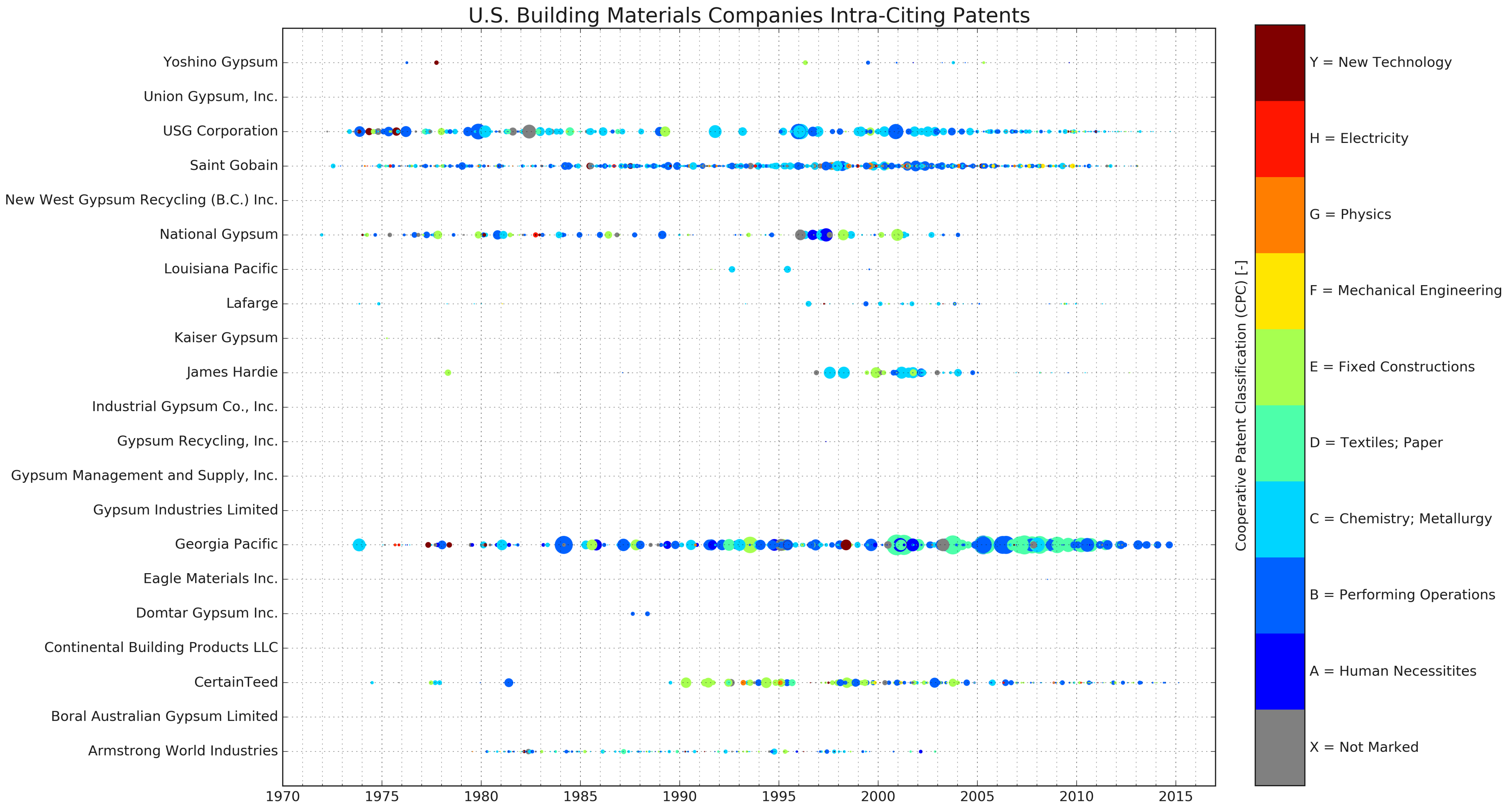


Figure 6: U.S. Building Material Patents: Invention Impact Using Intra-Citations (1970-2015)

Each bubble represents an individual patent out of 5,651 total patents. The bubble size is based upon the number of forward citations that the patent received over the life of the patent. The color of the bubble aligns with the technological classification per the CPC and matches the color bar located on the right hand side of Figure 6 [34, 75]. As expected, the categories of chemistry, performing operations, and textiles dominate this space. Leading companies in terms of patent counts are USG Corporation, Geogia Pacific, National Gypsum Company, Armstrong World Industries, and Saint Gobain. As is typical with citation analysis there is visual evidence of truncation with more recent patents having fewer forward citations than older patents. Larger citation bubbles for Georgia Pacific beginning in 2009 are attributable to a significant increase in patent applications in years 2012 and 2015 which included a large number of self-citations. This latent pattern was discovered as a result of this research and will be discussed in more detail in later sections. The next step in this research was to demonstrate that ML could be used to derive comparable insights by generating a similar visualization using counts of intra-cosine-related patent documents in place of intra-forward-citation counts. The steps for this process are discussed in the next section while the Python code for creation of the visualization can be referenced in Appendix I.

3.3 Text Mining

In this section ML techniques are applied to a subset of USPTO patent data to assess the similarity of patents using modified text mining techniques. It presents both the steps and considerations used to isolate key inventive terms from each patent as detailed in the process flowchart shown in Figure 7. The broad steps of the method are iterative and follow typical data mining methodology for the purpose of knowledge discovery (Table XIII).

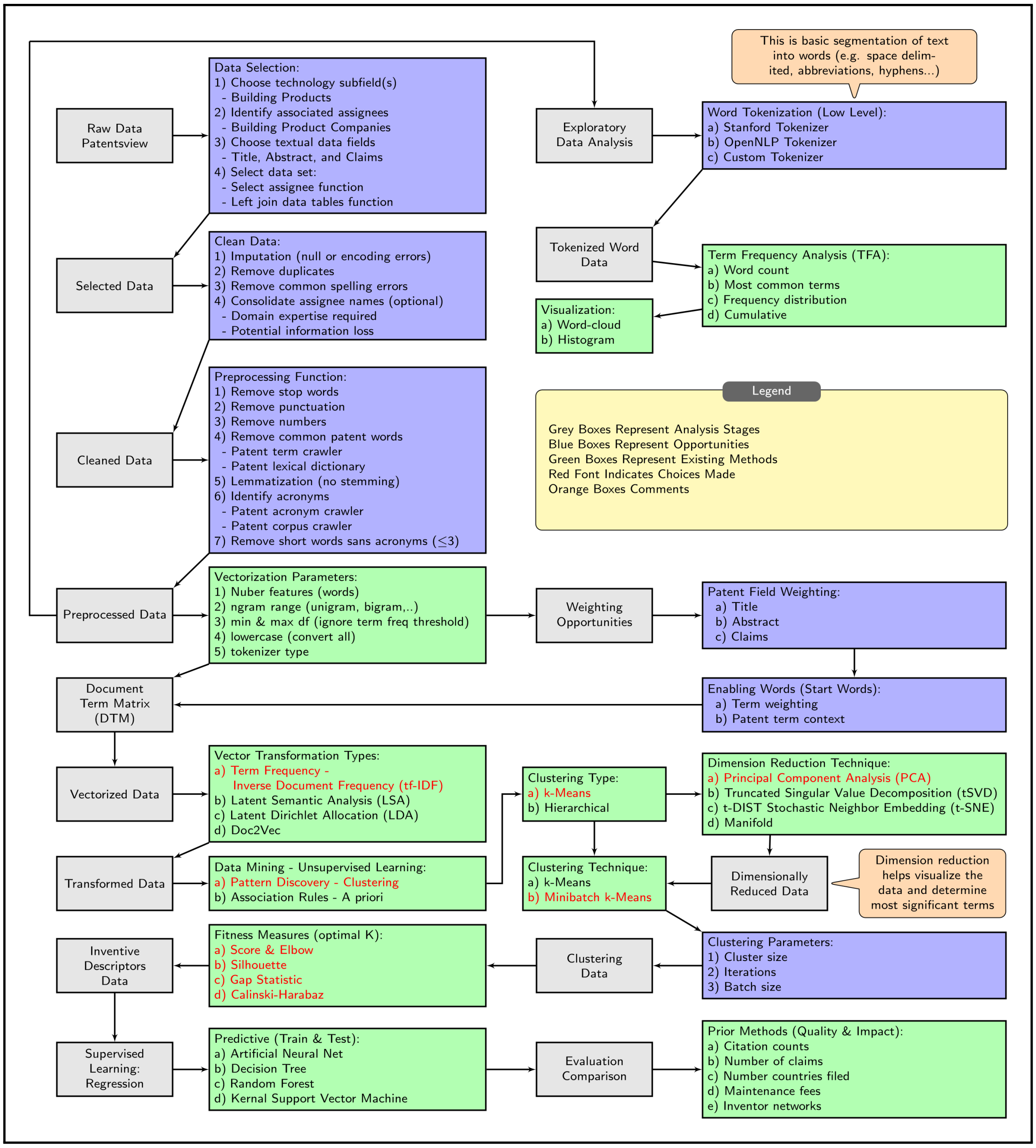
TABLE XIII: Patent NLP Steps

Step	Name	Description
1	Data Selection	Relevant data are retrieved from the database
2	Data Cleaning	Noise and inconsistent data are removed to ensure algorithm compatibility
3	Pre-processing	Data are prepared for use by ML algorithms
4	Vectorization	Textual data are converted into numerical data
5	Transformation	The converted numerical data are transformed and consolidated for mining
6	Dimension Reduction	Intelligent computational methods extract data patterns
7	Visualization	Interesting patterns representing actionable knowledge are shown

3.3.1 Data Selection

The first step of this analysis was to extract the unstructured textual data from the patent title, abstract, and claims and load it into the Python programming language environment [72]. A subset of patents from the building materials industry was selected based on domain expertise in this area, thus allowing human verification that methods were successful in isolating inventive language. To select patents associated with the building materials industry a look-up pattern was run on the organizational field of the assignee data table to select for organization names that contained the words “gypsum” or “cement”, or any companies that were known to compete in the building materials technology space. The patent identification numbers of the subset of

Figure 7: Patent Text Mining Method Flowchart


This is basic segmentation of text into words (e.g. space delimited, abbreviations, hyphens...)
Raw Data Patentsview
Data Selection:
1) Choose technology subfield(s)
- Building Products
2) Identify associated assignees
- Building Product Companies
3) Choose textual data fields
- Title, Abstract, and Claims
4) Select data set:
- Select assignee function
- Left join data tables function
Exploratory Data Analysis
Word Tokenization (Low Level):
a) Stanford Tokenizer
b) OpenNLP Tokenizer
c) Custom Tokenizer
Selected Data
Clean Data:
1) Imputation (null or encoding errors)
2) Remove duplicates
3) Remove common spelling errors
4) Consolidate assignee names (optional)
- Domain expertise required
- Potential information loss
Tokenized Word Data
Term Frequency Analysis (TFA):
a) Word count
b) Most common terms
c) Frequency distribution
d) Cumulative
Visualization:
a) Word-cloud
b) Histogram
Cleaned Data
Preprocessing Function:
1) Remove stop words
2) Remove punctuation
3) Remove numbers
4) Remove common patent words
- Patent term crawler
- Patent lexical dictionary
5) Lemmatization (no stemming)
6) Identify acronyms
- Patent acronym crawler
- Patent corpus crawler
7) Remove short words sans acronyms (≤3)
Legend
Grey Boxes Represent Analysis Stages
Blue Boxes Represent Opportunities
Green Boxes Represent Existing Methods
Red Font Indicates Choices Made
Orange Boxes Comments
Preprocessed Data
Vectorization Parameters:
1) Nuber features (words)
2) ngram range (unigram, bigram,..)
3) min & max df (ignore term freq threshold)
4) lowercase (convert all)
5) tokenizer type
Weighting Opportunities
Patent Field Weighting:
a) Title
b) Abstract
c) Claims
Document Term Matrix (DTM)
Enabling Words (Start Words):
a) Term weighting
b) Patent term context
Vectorized Data
Vector Transformation Types:
a) Term Frequency - Inverse Document Frequency (tf-IDF)
b) Latent Semantic Analysis (LSA)
c) Latent Dirichlet Allocation (LDA)
d) Doc2Vec
Clustering Type:
a) k-Means
b) Hierarchical
Dimension Reduction Technique:
a) Principal Component Analysis (PCA)
b) Truncated Singular Value Decomposition (tSVD)
c) t-DIST Stochastic Neighbor Embedding (t-SNE)
d) Manifold
Transformed Data
Data Mining - Unsupervised Learning:
a) Pattern Discovery - Clustering
b) Association Rules - A priori
Clustering Technique:
a) k-Means
b) Minibatch k-Means
Dimensionally Reduced Data
Dimension reduction helps visualize the data and determine most significant terms
Inventive Descriptors Data
Fitness Measures (optimal K):
a) Score & Elbow
b) Silhouette
c) Gap Statistic
d) Calinski-Harabaz
Clustering Data
Clustering Parameters:
1) Cluster size
2) Iterations
3) Batch size
Supervised Learning: Regression
Predictive (Train & Test):
a) Artificial Neural Net
b) Decision Tree
c) Random Forest
d) Kernal Support Vector Machine
Evaluation Comparison
Prior Methods (Quality & Impact):
a) Citation counts
b) Number of claims
c) Number countries filed
d) Maintenance fees
e) Inventor networks

building material companies were then used to select for corresponding patent data (e.g., patent number, title, abstract, and grant date) within the patent data table. The data was then merged into a single table for further analysis. The code for this analysis can be referenced in Appendix B. The next step was to disambiguate the assignee names to correct for duplicates. An example of disambiguation was the consolidation of all names associated with the building materials company USG (USG Company, USG Interiors, United States Gypsum Company, and USG Corporation) under a single assignee name. Assignee names of all subsidiaries were consolidated under their parent company name to make visualization of the data easier. This included updating assignee names for organizations that no longer existed due to bankruptcy, merger, or sale to another entity. This step was achieved via look-up and replace statements run against the raw assignee fields to pattern match assignee names. The code for the disambiguation step can be referenced in Appendix C.

3.3.2 Data Cleaning

Upon importation of the data into Python several cleaning steps were conducted to remove incomplete and incorrect data. This is required because some ML algorithms do not support null values or non-standard characters. Examples of incomplete values include *"not a number"* (NaN) entries and typographical errors. An NaN is an undefined or unrepresentable value that can not be processed in floating point calculations. Cleaning steps included removal of duplicates, removal of special encoding characters, and imputation. Imputation replaces missing data values with substitute values. Common imputation techniques include mean substitution and zeroing for null values. Mean substitution corrects for missing numerical data by substituting the mean of the variable from the set which has the benefit of not changing the overall sample mean.

3.3.3 Pre-processing

The objective of pre-processing is to remove low-value information terms from documents prior to vectorization [5, 76, 77]. Pre-processing is conducted through a sequence of steps, with each step resulting in the removal of additional low-value information, leaving only high-quality terms for analysis. The Natural Language Toolkit (NLTK) package for Python includes algorithmic tools to conduct many of the standard steps [78]. Table XIV lists the common steps as well as additional steps used in this research. This process improves upon the process steps

TABLE XIV: Patent Pre-processing Function Steps

Step	#	Name	Description
Standard	1	Tokenization	Breaks up strings of text into individual words
	2	Stop Words	Removes high frequency low value words like "the", "is", and "at"
	3	Punctuation	Removes unnecessary punctuation characters like apostrophes and dashes. Note modification to keep hyphenated words for this research.
	4	Lemmatisation	Reduces inflectional word forms while maintaining common word form
Custom	5	Patent Jargon	Designed to remove low value legal terms used in the patent industry
	6	Word Length	Designed to remove acronyms, abbreviations, encoding errors
	7	Numerical	Designed to remove all numbers from the string

by removing noise from the corpus through a reduction of term features that are common to intellectual property documents. The first step is to join the unstructured title, abstract, and claim textual data into a single character string for pre-processing. For efficiency many of the pre-processing steps have been consolidated into a custom function designed to strip text strings of non-value-added terms, as shown in Listing 3.1. The purpose of each line of the custom

```
 1  def clean(doc):
 2      number_free = ''.join([c for c in doc if c not in "1234567890"])
 3      words = [word.strip(string.punctuation) for word in number_free.split(" ")]
 4      filtered = [f for f in words if f and f.lower() not in stop_words]
 5      undo = "".join([" "+i if not i.startswith("'") and i not in string.punctuation else i
    ↪ for i in filtered]).strip()
 6      punc_free = ''.join(ch for ch in undo if ch not in punctuations)
 7      smallword_free = ' '.join([w for w in punc_free.split() if len(w)>word_len])
 8      lemmatized = " ".join(lemma.lemmatize(word) for word in smallword_free.split())
 9      jargon_free = " " .join([j for j in lemmatized.lower().split() if j not in jargon])
10      for i in jargon_free:
11          jargon_free = re.sub((i+i+i), ' ', jargon_free)
12          #jargon_free = jargon_free.replace('the ',' ') # not needed with proper space
    ↪ inserted on merge
13      nonsense = ' '.join([w for w in jargon_free.split() if len(w)>1])
14      return nonsense
15
16  corpus_clean = [clean(doc) for doc in corpus] # list of sentence strings
17  corpus_tokenize = [clean(doc).split() for doc in corpus]  # list of string words
```

Listing 3.1: Patent corpus pre-processing cleaning function

function is as follows:

1. Line 2 removes numbers from the text strings while line 3 is intended to parse words and remove spaces.
2. Lines 4 and 5 remove common low value information words known as *stop words*. Examples of stop words include common filler words like “an”, “the” and “to”. A standard stop word list is included with the NLTK package for Python [78].

3. Line 6 strips punctuation while keeping hyphens which is accomplished through modification of the standard punctuation list provided by NLTK. Hyphens are retained because they are common in patent chemical descriptions and inventive language.

4. Line 7 removes small common words that were not removed by line 4 based on an input parameter for character length. In this case the length parameter value was set to keep all words greater than 2 characters. Examples of small words that are removed include an abbreviation like “wt” for weight. Further improvement of this step would be to expand important abbreviations first. For example words like ”C” for Celsius and “g” for grams could be expanded via inclusion of patent abbreviation dictionary.

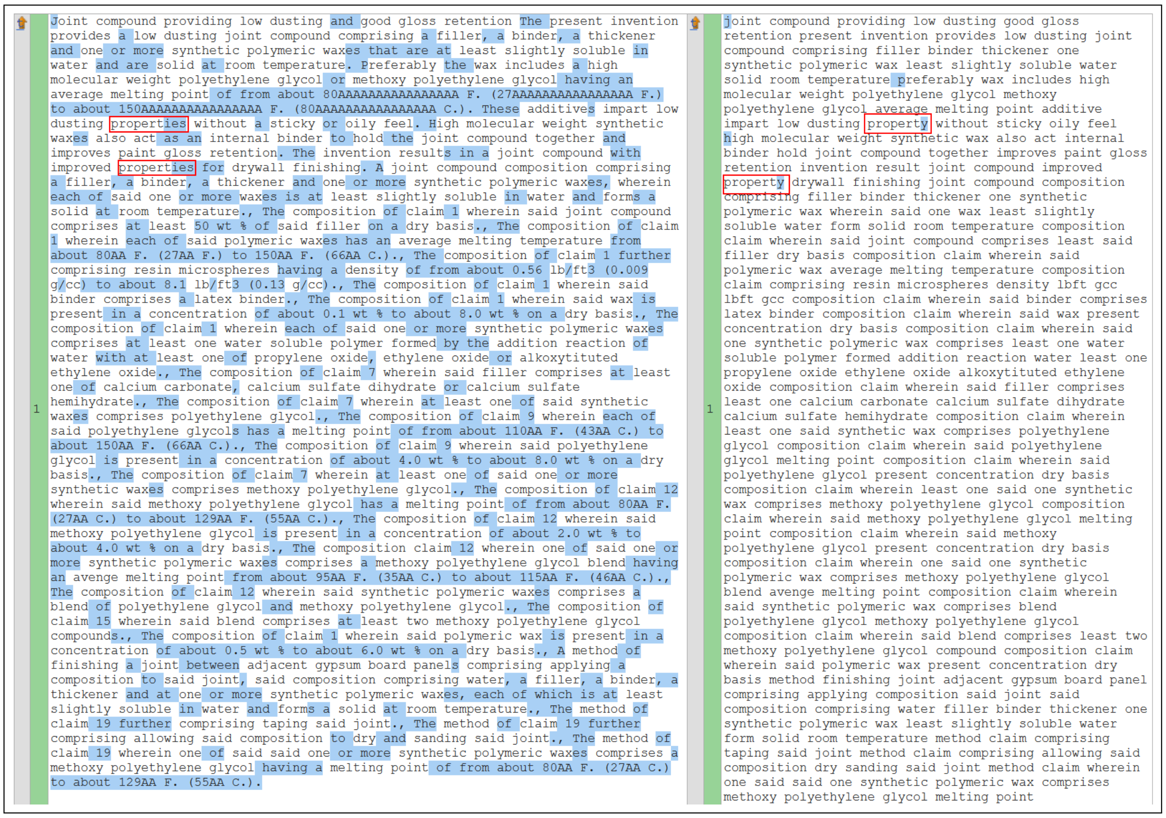
5. Line 8 is a process step referred to as *lemmatisation* which is a method of grouping inflected word forms together and is a standard tool in the NLTK package. Lemmatisation removes inflectional endings while still returning base word forms as it first identifies common parts of speech (POS) and then replaces the word with its base form. An example of lemmatisation can be seen in Figure 8 where the term “properties” has been outlined in red on the left hand side and been converted to the word “property” on the right hand side. The other approach to reducing inflectional word forms is known as *stemming*. Stemming is a faster approach which works by removing word suffixes like “ing” or “ly”. It does not take into account the context of the word based upon POS and can result in non-words. For example the word “sitting” can be reduced to “sitt” instead of “sit”. This method was not integrated into the custom function because it resulted in the generation of non-words which undermines the ability to identify inventive descriptors.

6. Line 9 is intended to remove common legal terms associated with intellectual property preparation and USPTO filing requirements that are referred to in this research as *patent jargon*. A term frequency bar chart comparison of before and after patent jargon removal can be seen in Figure 9 and Figure 10. The text output on the right side of Figure 8 was used as the input on the left side of Figure 11. The result of the jargon removal step is the output on the right hand side of Figure 11. Note that the patent jargon dictionary, which was created to remove common legal words, reduces the size of the string beyond what is accomplished by standard pre-processing. The patent jargon dictionary currently includes the words in Listing 3.2. Improvement of the jargon dictionary is a subject of future research.

7. Lines 10-13 are the final steps of the function which remove nonsensical terms such as those related to encoding errors. This is accomplished by removing terms which have a fixed number of consecutive characters (e.g. "aaa"). The number of repeating characters is set by the domain expert. In this case terms that contain three or more repeating characters were removed.

8. Lines 16 and 17 of 3.1 are commands to clean each patent document string and also parse words for frequency analysis.

Figure 12 is a word-cloud visualization based upon the output from line 17 of Listing 3.1 which shows the highest frequency words from the patent corpus. An illustration of the effectiveness of the patent pre-processing function in removing low value information terms can be seen in Figure 8 where it was applied to U.S. Patent 6,673,144. The left side of the figure shows



```
Joint compound providing low dusting and good gloss retention The present invention
provides a low dusting joint compound comprising a filler, a binder, a thickener
and one or more synthetic polymeric waxes that are at least slightly soluble in
water and are solid at room temperature. Preferably the wax includes a high
molecular weight polyethylene glycol or methoxy polyethylene glycol having an
average melting point of from about 80AAAAAAAAAAAAAAAA F. (27AAAAAAAAAAAAAAAA F.)
to about 150AAAAAAAAAAAAAAAA F. (80AAAAAAAAAAAAAAAA C.). These additives impart low
dusting properties without a sticky or oily feel. High molecular weight synthetic
waxes also act as an internal binder to hold the joint compound together and
improves paint gloss retention. The invention results in a joint compound with
improved properties for drywall finishing. A joint compound composition comprising
a filler, a binder, a thickener and one or more synthetic polymeric waxes, wherein
each of said one or more waxes is at least slightly soluble in water and forms a
solid at room temperature., The composition of claim 1 wherein said joint compound
comprises at least 50 wt % of said filler on a dry basis., The composition of claim
1 wherein each of said polymeric waxes has an average melting temperature from
about 80AA F. (27AA F.) to 150AA F. (66AA C.)., The composition of claim 1 further
comprising resin microspheres having a density of from about 0.56 lb/ft3 (0.009
g/cc) to about 8.1 lb/ft3 (0.13 g/cc)., The composition of claim 1 wherein said
binder comprises a latex binder., The composition of claim 1 wherein said wax is
present in a concentration of about 0.1 wt % to about 8.0 wt % on a dry basis., The
composition of claim 1 wherein each of said one or more synthetic polymeric waxes
comprises at least one water soluble polymer formed by the addition reaction of
water with at least one of propylene oxide, ethylene oxide or alkoxytituted
ethylene oxide., The composition of claim 7 wherein said filler comprises at least
one of calcium carbonate, calcium sulfate dihydrate or calcium sulfate
hemihydrate., The composition of claim 7 wherein at least one of said synthetic
waxes comprises polyethylene glycol., The composition of claim 9 wherein each of
said polyethylene glycols has a melting point of from about 110AA F. (43AA C.) to
about 150AA F. (66AA C.)., The composition of claim 9 wherein said polyethylene
glycol is present in a concentration of about 4.0 wt % to about 8.0 wt % on a dry
basis., The composition of claim 7 wherein at least one of said one or more
synthetic waxes comprises methoxy polyethylene glycol., The composition of claim 12
wherein said methoxy polyethylene glycol has a melting point of from about 80AA F.
(27AA C.) to about 129AA F. (55AA C.)., The composition of claim 12 wherein said
methoxy polyethylene glycol is present in a concentration of about 2.0 wt % to
about 4.0 wt % on a dry basis., The composition claim 12 wherein one of said one or
more synthetic polymeric waxes comprises a methoxy polyethylene glycol blend having
an avenge melting point from about 95AA F. (35AA C.) to about 115AA F. (46AA C.).,
The composition of claim 12 wherein said synthetic polymeric waxes comprises a
blend of polyethylene glycol and methoxy polyethylene glycol., The composition of
claim 15 wherein said blend comprises at least two methoxy polyethylene glycol
compounds., The composition of claim 1 wherein said polymeric wax is present in a
concentration of about 0.5 wt % to about 6.0 wt % on a dry basis., A method of
finishing a joint between adjacent gypsum board panels comprising applying a
composition to said joint, said composition comprising water, a filler, a binder, a
thickener and at one or more synthetic polymeric waxes, each of which is at least
slightly soluble in water and forms a solid at room temperature., The method of
claim 19 further comprising taping said joint., The method of claim 19 further
comprising allowing said composition to dry and sanding said joint., The method of
claim 19 wherein one of said said one or more synthetic polymeric waxes comprises a
methoxy polyethylene glycol having a melting point of from about 80AA F. (27AA C.)
to about 129AA F. (55AA C.).
```

```
joint compound providing low dusting good gloss
retention present invention provides low dusting joint
compound comprising filler binder thickener one
synthetic polymeric wax least slightly soluble water
solid room temperature preferably wax includes high
molecular weight polyethylene glycol methoxy
polyethylene glycol average melting point additive
impart low dusting property without sticky oily feel
high molecular weight synthetic wax also act internal
binder hold joint compound together improves paint gloss
retention invention result joint compound improved
property drywall finishing joint compound composition
comprising filler binder thickener one synthetic
polymeric wax wherein said one wax least slightly
soluble water form solid room temperature composition
claim wherein said joint compound comprises least said
filler dry basis composition claim wherein said
polymeric wax average melting temperature composition
claim comprising resin microspheres density lbft gcc
lbft gcc composition claim wherein said binder comprises
latex binder composition claim wherein said wax present
concentration dry basis composition claim wherein said
one synthetic polymeric wax comprises least one water
soluble polymer formed addition reaction water least one
propylene oxide ethylene oxide alkoxytituted ethylene
oxide composition claim wherein said filler comprises
least one calcium carbonate calcium sulfate dihydrate
calcium sulfate hemihydrate composition claim wherein
least one said synthetic wax comprises polyethylene
glycol composition claim wherein said polyethylene
glycol melting point composition claim wherein said
polyethylene glycol present concentration dry basis
composition claim wherein least one said one synthetic
wax comprises methoxy polyethylene glycol composition
claim wherein said methoxy polyethylene glycol melting
point composition claim wherein said methoxy
polyethylene glycol present concentration dry basis
composition claim wherein one said one synthetic
polymeric wax comprises methoxy polyethylene glycol
blend avenge melting point composition claim wherein
said synthetic polymeric wax comprises blend
polyethylene glycol methoxy polyethylene glycol
composition claim wherein said blend comprises least two
methoxy polyethylene glycol compound composition claim
wherein said polymeric wax present concentration dry
basis method finishing joint adjacent gypsum board panel
comprising applying composition said joint said
composition comprising water filler binder thickener one
synthetic polymeric wax least slightly soluble water
form solid room temperature method claim comprising
taping said joint method claim comprising allowing said
composition dry sanding said joint method claim wherein
one said said one synthetic polymeric wax comprises
methoxy polyethylene glycol melting point
```

Figure 8: Patent 6,673,144: Pre-processing Output of Unstructured Textual Data with Examples of Lemmatisation Boxed in Red

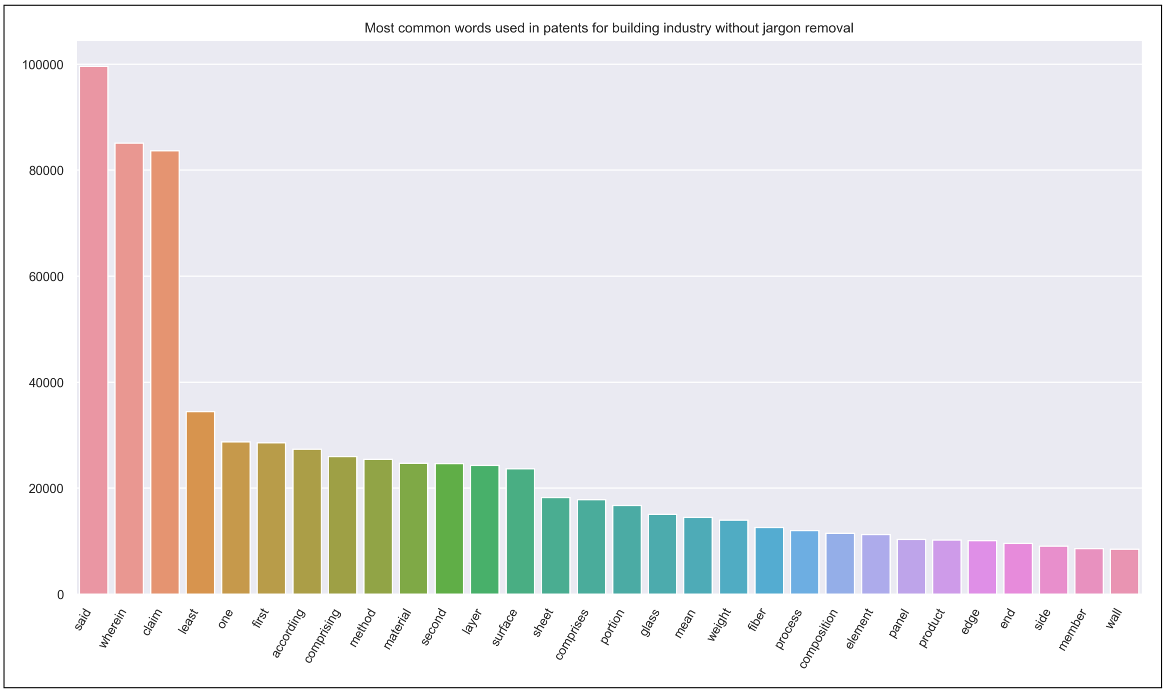


Figure 9: Patent Term Frequency Analysis Pre-Jargon Removal

blue highlighted low value combinations of terms while the right side represents term output prior to running the jargon removal step. By removing non-value-added terms the size of the inventive string was reduced by 69%. This includes removal of all numbers, punctuation, and words shorter than two characters, without losing high value inventive terms or their relative order within the string. A comparison between pre- and post-jargon removal can be referenced in Figure 11 where the left hand side of the figure highlights patent jargon in blue that was removed from the string. The patent jargon removal step reduced the inventive string by an additional 21% over standard pre-processing. The terms that were automatically generated on

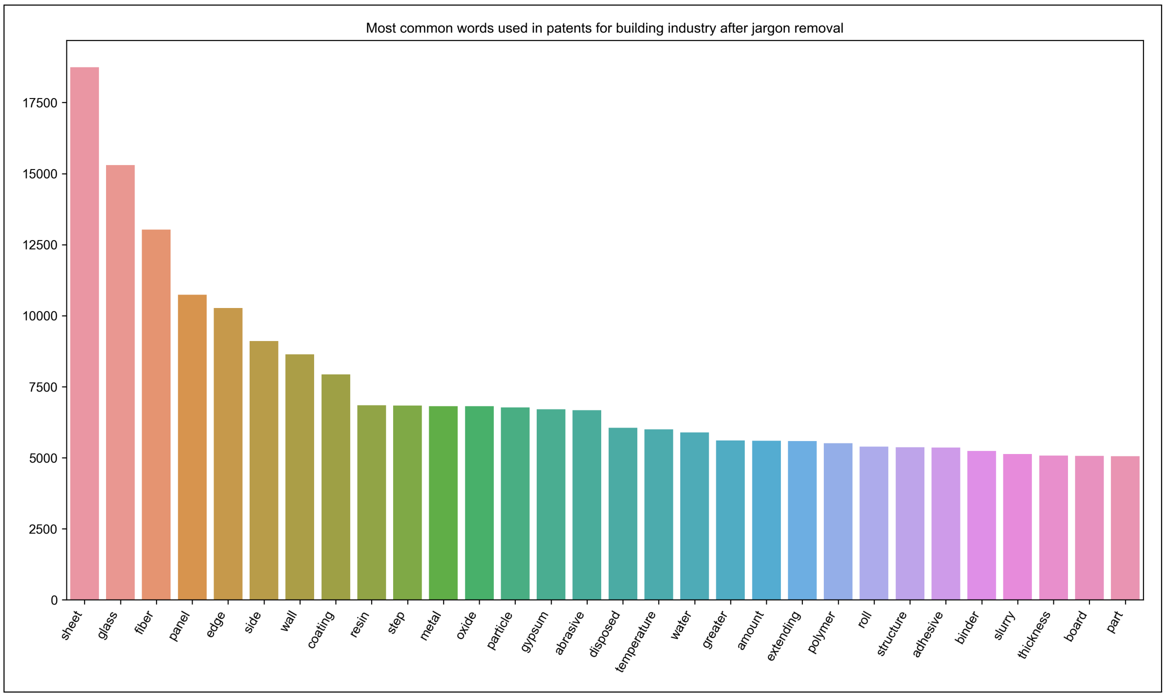
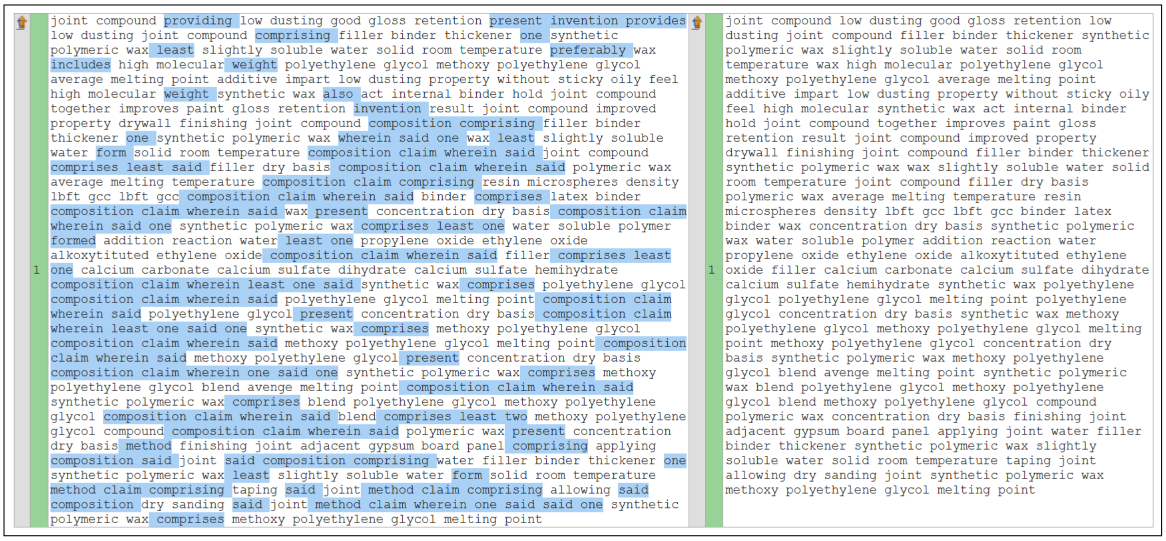


Figure 10: Patent Term Frequency Analysis Post-Jargon Removal



```
joint compound providing low dusting good gloss retention present invention provides
low dusting joint compound comprising filler binder thickener one synthetic
polymeric wax least slightly soluble water solid room temperature preferably wax
includes high molecular weight polyethylene glycol methoxy polyethylene glycol
average melting point additive impart low dusting property without sticky oily feel
high molecular weight synthetic wax also act internal binder hold joint compound
together improves paint gloss retention invention result joint compound improved
property drywall finishing joint compound composition comprising filler binder
thickener one synthetic polymeric wax wherein said one wax least slightly soluble
water form solid room temperature composition claim wherein said joint compound
comprises least said filler dry basis composition claim wherein said polymeric wax
average melting temperature composition claim comprising resin microspheres density
lbft gcc lbft gcc composition claim wherein said binder comprises latex binder
composition claim wherein said wax present concentration dry basis composition claim
wherein said one synthetic polymeric wax comprises least one water soluble polymer
formed addition reaction water least one propylene oxide ethylene oxide
alkoxytituted ethylene oxide composition claim wherein said filler comprises least
one calcium carbonate calcium sulfate dihydrate calcium sulfate hemihydrate
composition claim wherein least one said synthetic wax comprises polyethylene glycol
composition claim wherein said polyethylene glycol melting point composition claim
wherein said polyethylene glycol present concentration dry basis composition claim
wherein least one said one synthetic wax comprises methoxy polyethylene glycol
composition claim wherein said methoxy polyethylene glycol melting point composition
claim wherein said methoxy polyethylene glycol present concentration dry basis
composition claim wherein one said one synthetic polymeric wax comprises methoxy
polyethylene glycol blend avenge melting point composition claim wherein said
synthetic polymeric wax comprises blend polyethylene glycol methoxy polyethylene
glycol composition claim wherein said blend comprises least two methoxy polyethylene
glycol compound composition claim wherein said polymeric wax present concentration
dry basis method finishing joint adjacent gypsum board panel comprising applying
composition said joint said composition comprising water filler binder thickener one
synthetic polymeric wax least slightly soluble water form solid room temperature
method claim comprising taping said joint method claim comprising allowing said
composition dry sanding said joint method claim wherein one said said one synthetic
polymeric wax comprises methoxy polyethylene glycol melting point
```

```
joint compound low dusting good gloss retention low
dusting joint compound filler binder thickener synthetic
polymeric wax slightly soluble water solid room
temperature wax high molecular polyethylene glycol
methoxy polyethylene glycol average melting point
additive impart low dusting property without sticky oily
feel high molecular synthetic wax act internal binder
hold joint compound together improves paint gloss
retention result joint compound improved property
drywall finishing joint compound filler binder thickener
synthetic polymeric wax wax slightly soluble water solid
room temperature joint compound filler dry basis
polymeric wax average melting temperature resin
microspheres density lbft gcc lbft gcc binder latex
binder wax concentration dry basis synthetic polymeric
wax water soluble polymer addition reaction water
propylene oxide ethylene oxide alkoxytituted ethylene
oxide filler calcium carbonate calcium sulfate dihydrate
calcium sulfate hemihydrate synthetic wax polyethylene
glycol polyethylene glycol melting point polyethylene
glycol concentration dry basis synthetic wax methoxy
polyethylene glycol methoxy polyethylene glycol melting
point methoxy polyethylene glycol concentration dry
basis synthetic polymeric wax methoxy polyethylene
glycol blend avenge melting point synthetic polymeric
wax blend polyethylene glycol methoxy polyethylene
glycol blend methoxy polyethylene glycol compound
polymeric wax concentration dry basis finishing joint
adjacent gypsum board panel applying joint water filler
binder thickener synthetic polymeric wax slightly
soluble water solid room temperature taping joint
allowing dry sanding joint synthetic polymeric wax
methoxy polyethylene glycol melting point
```

Figure 11: U.S. Patent 6673144: Pre-Processing Comparison with Blue Highlighted Legal Jargon Removed

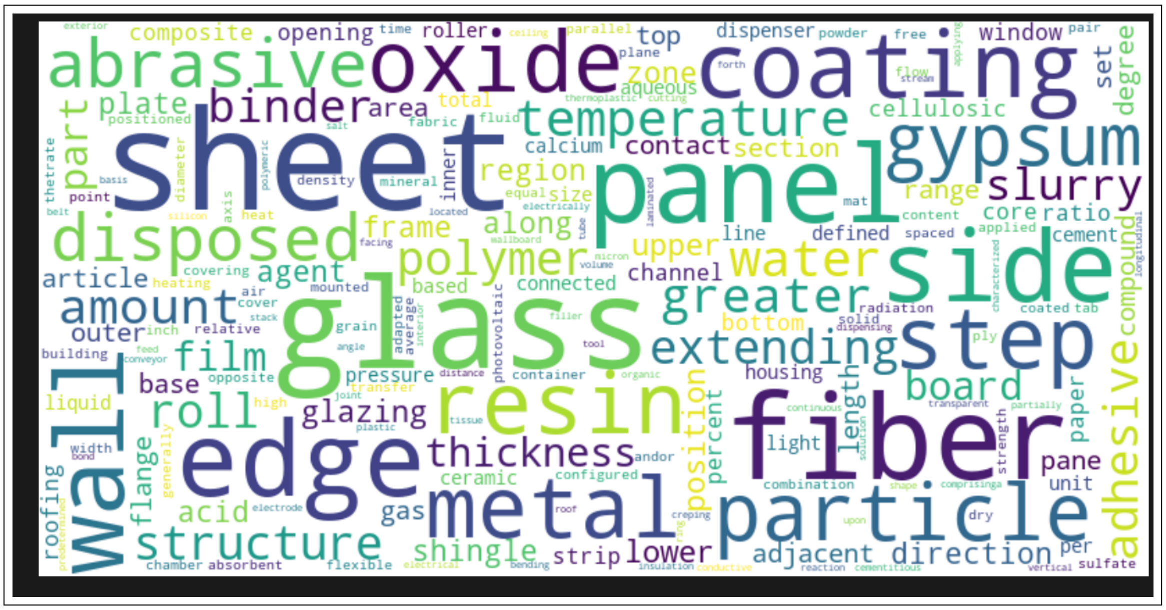


Figure 12: Patent Corpus Pre-processing Output Word Cloud

the right hand side of Figure 11 represent core inventive descriptors as validated by a domain expert. The application of the pre-processing function resulted in an inventive descriptor string for each patent which was then converted to a vector as described in Section 3.3.4.

3.3.4 Vectorization

The next step in the analysis is to convert the pre-processed strings of text into numerical vectors using the scikit-learn CountVectorizer package [79]. The count vectorizer algorithm assigns numerical values, counts terms, and then places the vectors into a sparse document-term matrix for use in a Vector Space Model (VSM). There are several important parameters in this model that require domain expert input. These include the maximum and minimum term frequency threshold values, case sensitivity, and the upper and lower boundaries for n-gram

```
jargon = {'according', 'also', 'apparatus', 'assembly', 'body', 'claim',
          'claimed', 'component', 'composition', 'comprise', 'comprises',
          'comprising', 'consisting', 'containing', 'device', 'disclosed',
          'element', 'embodying', 'end', 'face', 'first', 'form', 'formed',
          'forming', 'forms', 'group', 'include', 'includes', 'including',
          'invention', 'layer', 'le', 'least', 'made', 'making', 'material',
          'may', 'mean', 'means', 'member', 'method', 'mixture', 'one',
          'patent', 'plurality', 'portion', 'preferably', 'present',
          'process', 'product', 'provided', 'provides', 'providing', 'relates',
          'resulting', 'said', 'second', 'selected', 'substantially',
          'substrate', 'support', 'surface', 'system', 'technology', 'thereof',
          'third', 'two', 'web', 'weight', 'wherein', 'within', 'wt'}

```

Listing 3.2: Jargon dictionary

range. N-grams are a key part of NLP related to the identification of common sequences of words which play an important role in contextual meaning. The maximum threshold for term frequency across the building materials patent corpus was set to 65%. The result of this setting is that terms that exist across more than 65% of our patent documents will be dropped from the feature space. The minimum threshold was set to zero, meaning no infrequent or rare terms will be ignored, even if they exist only in a single patent document. The lowercase parameter was set to false, meaning that any terms containing capital letters would be converted to lowercase. Lastly, the n-gram range for feature extraction was set to include uni-gram, bi-gram, and tri-gram word combinations also known as n-grams. An example of a uni-gram is the chemical word “sodium” while a bi-gram could contain the words “sodium chloride”. Both example n-grams are important when trying to compare inventive language between patents as they clearly denote different contextual chemical meanings. Our n-gram range setting means that our terms can include one or more word combinations, in this case between one and three words, based upon how frequently they are co-located across the patent corpus. Optimization of n-gram settings is the subject of future research. The output of the CountVectorizer algorithm is a set of term frequency document vectors where the term frequency $tf(t, d)$ equals the raw count of the number of times term t occurs in document d. These vectors are then used as the input for the transformation step.

3.3.5 Transformation

There are a number of vector transformation techniques available to convert strings of unstructured text into structured data for analysis. Some example techniques include term frequency-inverse document frequency (tf-idf) [80–86], Latent Semantic Analysis (LSA) [87–89], and Latent Dirichlet Allocation (LDA) [90–94]. Tf-idf takes a vector and returns another vector of the same dimension, while both increasing the value of rarer features and reducing the value of overly frequent terms. It converts integer-valued vectors into real-valued ones, while leaving the number of dimensions intact. LSA can take output from tf-idf as input and correlate semantically related terms that are latent in a collection of text.

> Queries, or concept searches, against a set of documents that have undergone LSA will return results that are conceptually similar in meaning to the search criteria even if the results don't share a specific word or words with the search criteria [95].

This method did not appear to enhance outputs when applied to inventive language, especially when trying to derive semantic relationships between chemical terms. LDA is a generative statistical model that attempts to associate sets of terms with an unobserved topical group, then backtracks and tries to figure out what sets of unobserved topics would be required to create the original documents in the first place. The unobserved topic itself is then left for the human to label. LDA was not pursued as this research was focused on revealing latent core inventive terms and not unknown topics which have already been identified via CPC. The analysis presented here uses tf-idf, which weights the importance of a term in proportion to its use within a document and offsets that importance in proportion to the term's frequency of use across all documents. The tf-idf algorithm incorporates the term frequency (tf) aspect of the count vectorization step previously described. The inverse document frequency part of the equation is as follows:

$$idf(t) = \log \frac{1 + n_d}{1 + df(d,t)} + 1 \tag{3.1}$$

where n_d is the total number of documents in the corpus and $df(d,t)$ is the number of documents where term t appears. The tf-idf is then calculated by multiplying the term frequency by the inverse document frequency as follows:

$$tf\text{-}idf(t,d) = tf(t,d) \cdot idf(t) \tag{3.2}$$

where $tf(t,d)$ is the number of times term t appears in document d. The output of the transformation step is a set of real-valued numerical vectors representing each patent within the corpus which can then be placed into a document term matrix (DTM). Table XV is a small example of a document term matrix with the individual rows representing the patent vectors and the columns representing each word vector. The actual dimensions of the DTM for this research are given by the number of features (34,695 terms) across our patent corpus (5,651 patents).

The next step is to measure patent document vector relatedness using *cosine similarity*, which

		Terms		
		shaft	s-shaped	sulfate
Patents	8,323,429	0	0	1
	7,887,230	1	1	0
	6,673,144	0	0	1

TABLE XV: Example Document Term Matrix

measures the closeness of two non-zero vectors of an inner product space by calculating the angle between them.

3.3.5.1 Cosine Similarity

Cosine similarity is an extensively used measurement in the field of information retrieval and text analysis as well as a fundamental component of many ML clustering algorithms [5, 76, 83, 96–100]. Figure 13 provides a simple three-dimensional representation of the cosine angle between patent vectors a and b, which is calculated as follows:

$$sim(a,b) = cos(\theta) = \frac{a \cdot b}{||a||\,||b||} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \tag{3.3}$$

where $sim(a,b)$ is the similarity between patent vectors a and b, a_i and b_i are their respective components, and n is the total number of components. Cosine similarity was chosen over

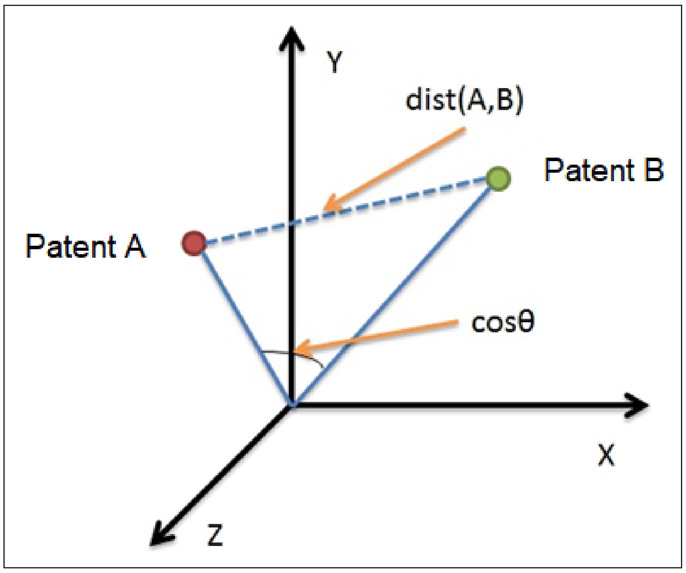


Figure 13: Cosine similarity example

Euclidean distance, see $dist(A, B)$ in Figure 13, because it is agnostic to differences in patent document length and can be calculated for an arbitrarily large number of dimensions. The cosine similarity between vectors is then used to create a square pairwise distance matrix of cosine similarity scores (CSS). This matrix can also be used for clustering, which is an unsupervised ML discovery technique used to discover relatedness among vectors. All diagonal entries in the cosine similarity matrix take the value 1, as each patent's set of terms is identical to itself. The first step was to measure the CSS for every pair of patent vectors within the patent corpus and then remove identical document pairs by subtracting the identity matrix. The remaining vectors are then placed into a patent CSS matrix. There is a wide body of research regarding the importance of removing identical documents from a corpus when attempting to measure document similarity [96, 101–103]. It was discovered that additional patent pairs with a CSS greater than (.99) existed within the set due to a process known as patent splitting. Patents can be split during the application review process if they are deemed to lack *unity of invention* which simply means they include more the one invention. In such cases the application is turned into one or more divisional applications which can result in patent documents with almost indistinguishable textual data (i.e. titles, abstract, and claims) yet different assigned patent numbers. Many times the resulting assigned patent numbers are consecutive. To ensure that only identical patents were removed, the code was modified to ensure only deletion of patent pairs with identical patent numbers.

3.3.5.2 Minimum CSS Threshold

To select for related patents using cosine similarity it is necessary to establish a method of relatedness. The proposed method is to assign a minimum cosine similarity threshold value. Specifically, we define a patent document vector space as follows:

1. Each patent document vector $v_d = \{t_1, t_2, \ldots, t_{m-1}, t_m\}$ is an element of the vector space, where m is the total number of term features, and each component (t_i) is the *tf-idf* of term (i) for patent document (d).
2. The vector space contains the set of patent document vectors $V = \{v_1, v_2, \ldots, v_n\}$, where n is the total number of patents,
3. A similarity function $sim(a, b)$ is defined for any pair of vectors in the vector space with its value given by the cosine angle $cos(\theta)$ between the vectors as described in Equation 3.3.

The goal is to select a similarity threshold value (q) so that $sim(a, b) \geq q$ for all patent pairs $a, b \in V$ such that a and b are related in inventive content, and $sim(a, b) < q$ otherwise. It seems plausible that the appropriate value of q is dependent on the set of patents under consideration and on the purpose of the comparison.

One approach to solving this problem is to establish a q through the use of a human-in-the-loop (HITL) that evaluates cosine similarity scores and provides subject relevance based upon domain expertise. Requiring human experts to establish a threshold value based upon subject relevance is counterproductive to developing an automated method of determining patent relatedness based upon inventive language and fails to exploit the wealth of data in the patent database. This research proposes a novel approach that relies upon the weighted mean CSS of known cited patent pairs to establish a reasonable minimum threshold value. This is based on a hypothesis that backward citations, and in turn forward citations, have some reasonable degree of relatedness due to patent office citation requirements of prior art and that this reasonable degree of relatedness could be used to establish a threshold value. A minimum threshold q for a set of cited patent pairs is shown in the following equation:

$$q = \mu + \alpha \cdot \sigma \tag{3.4}$$

where q is the minimum CSS threshold value, μ is the average CSS, α is a parameter, and σ is the standard deviation. Solving for α gives:

$$\alpha = \frac{-\mu + q}{\sigma} \tag{3.5}$$

Note that α indicates how many standard deviations above the mean the CSS value is for a pair of patents, thus higher values of α indicate greater relatedness. The first question to ask is whether patent pairs with high α values are likely to also have a citation relationship. The mean CSS score across all the patent pairs in the corpus is 0.018 with a standard deviation of 0.039, while the mean CSS score among only those patent pairs with a citation connection is 0.265, which is an α value of 6.33. Restricting the calculation to patent pairs in which the cited patent has a minimum number of citations yields the graph shown in Figure 14. The proposed approach to comparing cosine similarity scores with citation counts would be to declare a value of α as the metric by which two patents are declared similar while also minimizing the effect of potential over-citing within smaller samples. For the data set processed here and the results shown in Figure 14, a value for α was selected based upon the weighted mean CSS per the following equation:

$$\bar{q} = \frac{\sum\limits_{i=1}^{n_c} w_i x_i}{\sum\limits_{i=1}^{n_c} w_i} \tag{3.6}$$

where $\bar{q}$ represents the weighted mean CSS, n_c represents the total number of citations, w_i represents the count of patents with a count of i citations, and x_i represents the mean CSS of the set of patents with i citations. The weighted mean CSS of the set of patents shown in Figure 14 was 0.27 which when substituted for q in Equation 3.5 resulting in an α value of 6.9 which was then set as the declared α value. The set of related patent pairs as determined by minimum CSS threshold can then be compared to the set of related patent pairs as determined by citation relationships. Further optimization of the α value based upon known cited patent pairs is left for future research.

3.3.6 Dimension Reduction

Since we can not directly visualize cosine similarity within the higher dimension vector space several different dimension-reducing techniques were considered. After evaluating Principal Component Analysis (PCA) [104–106], Single Value Decomposition (SVD) [83, 106–109], and t-distributed Stochastic Neighbor Embedding (t-SNE) [110–112] it was decided to use both PCA and t-SNE for visualization.

3.3.6.1 Principal Component Analysis

PCA is the most computationally efficient dimension reduction technique with a minimal amount of information loss [104–106]. PCA finds optimal lower-dimensional approximations of a data set by projecting it to linear sub-spaces. A two-dimensional representation of the building material patent corpus can be seen in Figure 15.

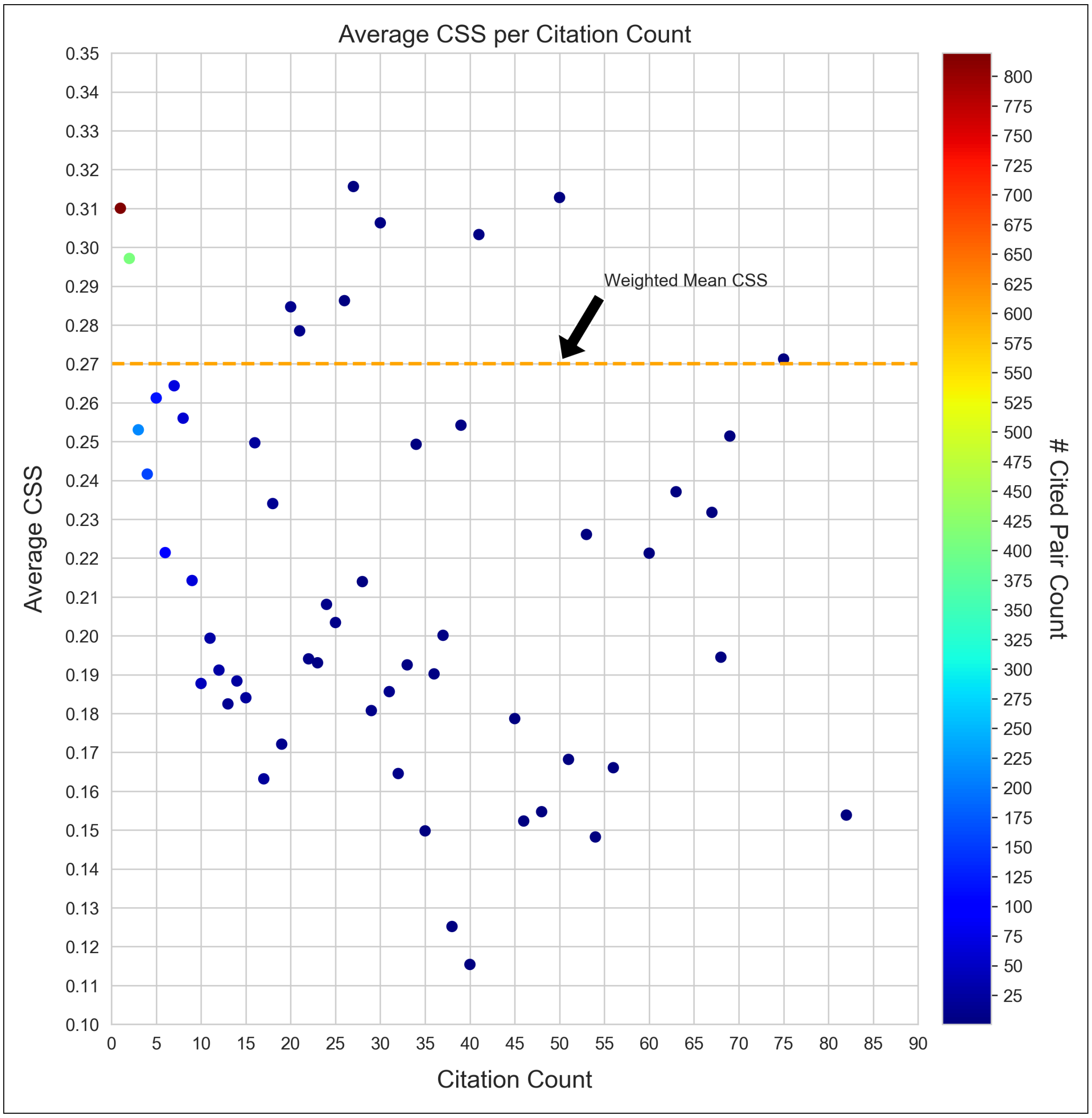


Figure 14: Changes in mean CSS for patent citation pairs based on total forward citation count.

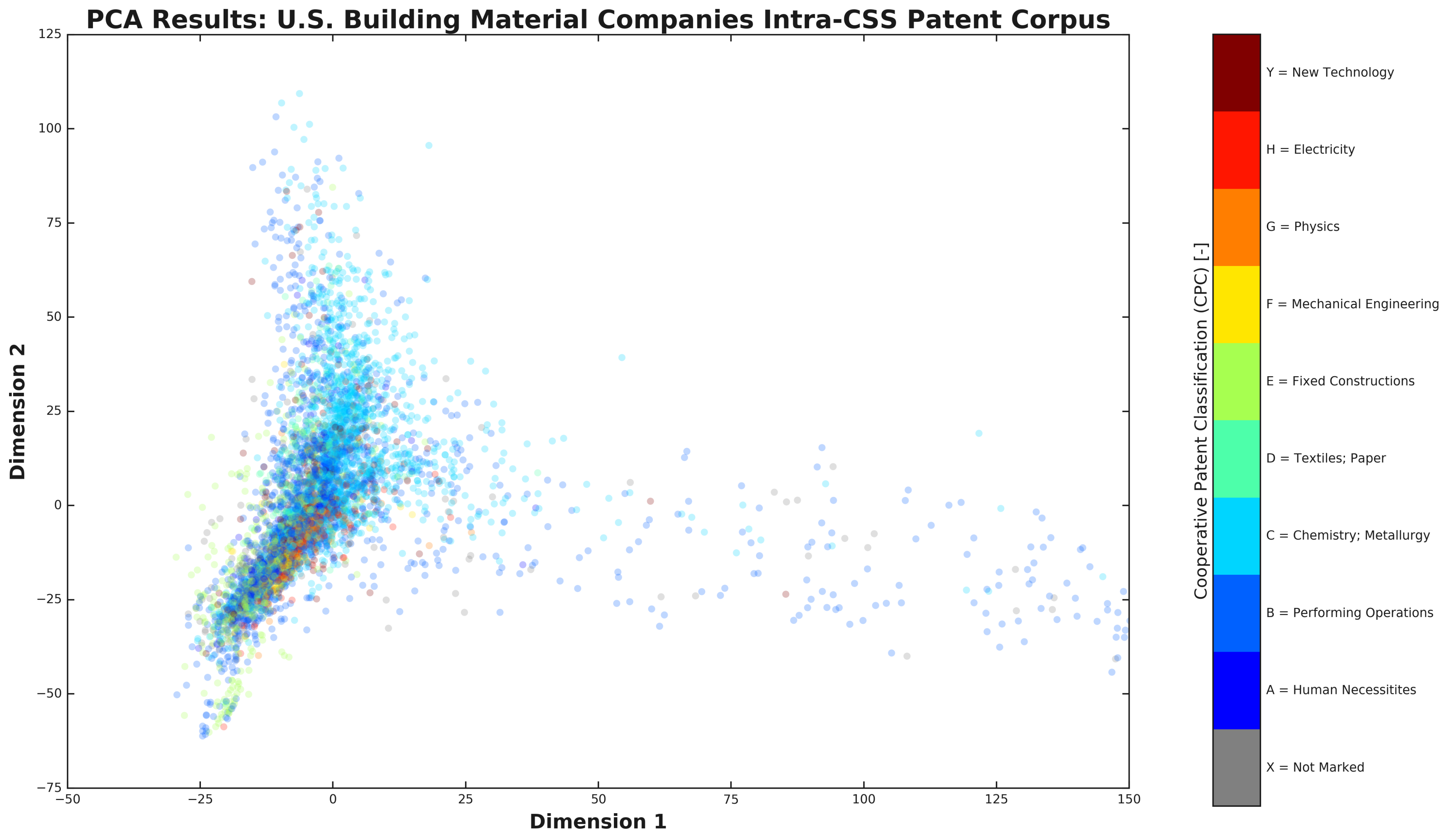


Figure 15: Dimension Reduction via Principal Component Analysis

Each dot represents a single patent document with the distance between each pair of dots representative of their similarity. Darker dots represent overlap of patent documents. The dots have also been color coded using the CPC main classification to test whether the similarity of patent documents aligns with assigned technical category. The decision to overlay a known technical category onto the dimensional reduced data was intended to overcome a lack of clear patterns using k-means clustering.

3.3.6.1.1 k-means Clustering

This research initially utilized an unsupervised ML technique called k-means clustering to reveal patterns among similar patents. [104, 113–115]. K-means is a vector quantization method designed to partition data by placing observations into clusters whose centroid represents the nearest mean. The result of this early research was inconclusive and the method required the user to set a somewhat arbitrary parameter for the number of clusters without knowing what variables make each cluster distinct. To decide on an appropriate cluster value a host of techniques were used including silhouette [116], Calinski-Harabaz [117], gap statistic [118], and elbow method [119]. All of these measures assess within-cluster dispersion and between-cluster dispersion in attempt to find the optimal cluster value. The results of this preliminary research suggested a cluster parameter of six was appropriate which can be seen in Figure 16. This figure shows a PCA-reduced two-dimensional plot of the data set with six distinct color coded clusters; however, there is no clear pattern as to what made each cluster distinct. Figure 15 and Figure 16 demonstrate a similar problem and highlight one of the drawbacks of using PCA to represent high-dimensional spaces: it has a tendency to tightly cluster outputs, making results difficult to interpret [120].

3.3.6.2 t-distributed Stochastic Neighbor Embedding

Further research indicated that t-SNE was effective for visualizing higher-dimensional data associated with document similarity; however due to the high computational cost the author of the algorithm (van der Matten) recommends the use of another dimension reduction technique first [110–112]. Based upon this guidance the model was first reduced to fifty dimensions using PCA and then reduced to a two-dimensional plot using t-SNE. The t-SNE algorithm has a perplexity parameter that influences the number of nearest neighbors and thus the degree to which clusters are visualized by expanding dense clusters and contracting sparse ones. The perplexity parameter was tested from 0 to 100 using a step value of one. A perplexity value of thirty provided the best visual aesthetic based upon an acceptable balance of dense and sparse clusters. The resulting t-SNE plot can be referenced in Figure 17 which represents a holistic 2-dimensional visualization of the patent corpus in terms of document similarity.

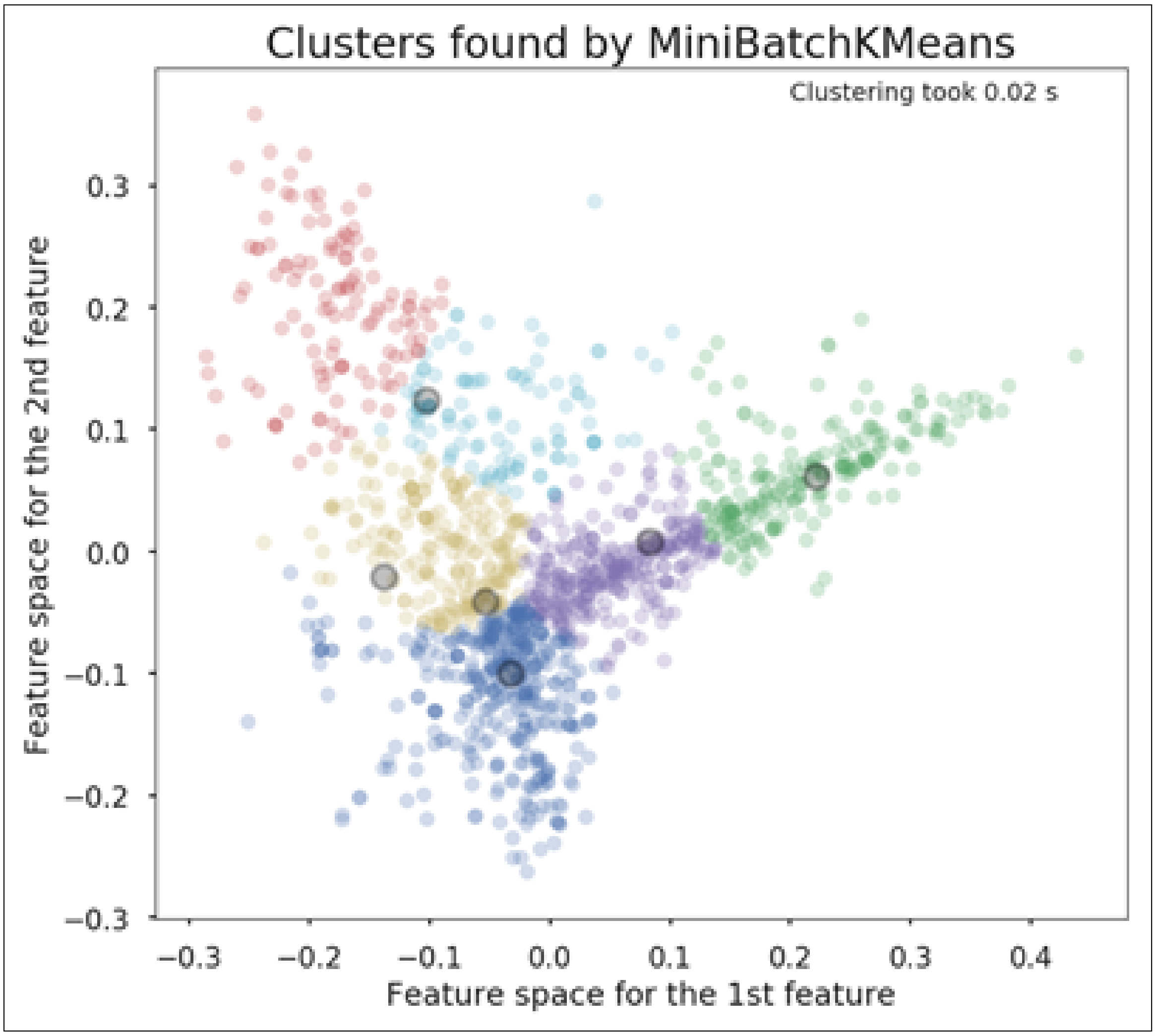


Figure 16: k-means clustering using six clusters

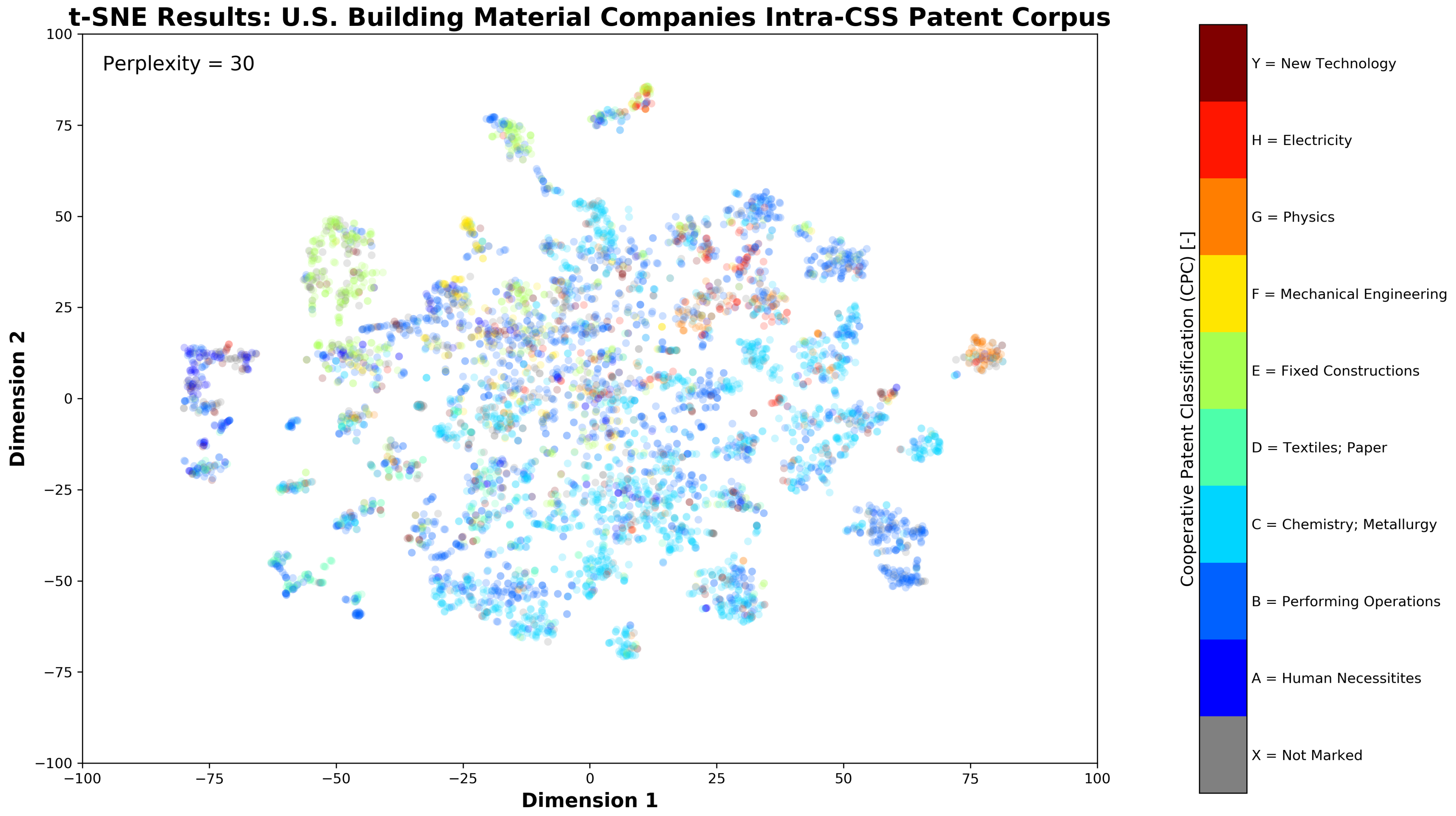


Figure 17: t-SNE results: U.S. Building Material Companies Intra-CSS Patent Corpus

To assess how well the patent relatedness measure worked the data points were color coded for each patent document using the USPTO assigned CPC categories. Figure 17 shows clear clusters aligned with specific CPC technology categories (e.g., physics, human necessities, and fixed constructions) as well as some areas of category mixing. The match between the relatedness clusters and CPC classes, while not formal validation, indicates that this approach is aligned with traditional classication methods. From a domain expert's perspective these clusters are of specific interest because they reveal patents with similar inventive knowledge. It is important to note that a given patent can have more than one CPC category assigned and that assignment is based upon when a technical category is first listed within an application and not by frequency or priority. The opportunity to use this method to reveal latent patterns among CPC technical category assignments is left for future research.

3.3.7 Invention Impact: Intra-CSS Visualization

Figure 6 has been reproduced using intra-CSS related patent counts in place of intra-forward-citation counts to demonstrate that this method is capable of measuring invention impact. The output can be seen in Figure 18.

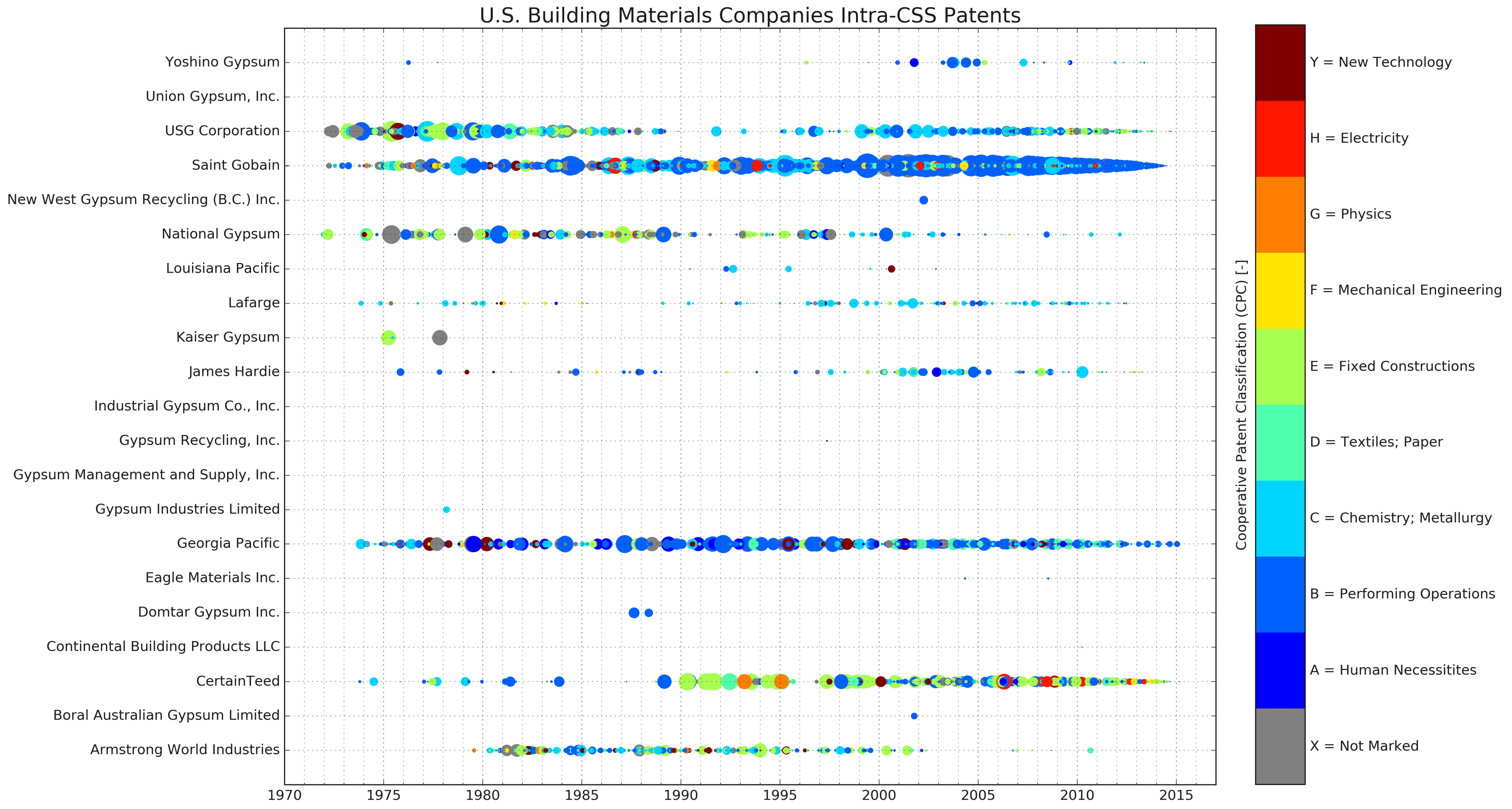


Figure 18: U.S. Building Material Patents: Invention Impact Using Intra-Cosine Similarity Threshold (1970 - 2015)

Figure 18 was generated by selecting intra-CSS forward related patents that exceeded the minimum cosine similarity threshold value while removing any intra-CSS related patents that occurred prior to the focus patent's application date. The application date of the intra-CSS related patent was used instead of the grant date because research showed that it can take a couple of years for a patent to be granted and during that time frame a patent application can receive forward citations. It is important to note that USPTO began to publish applications 18 months after filing effective November 29, 2000, contents of patents filed prior to this date were not made public until the actual grant date [121]. Initially, Figure 18 was created using a count of forward citations after the focus patent's grant date as it was a clean demarcation of when the inventive knowledge was made publicly available independent of any USPTO publishing policy changes; however, it was later discovered that a number of forward citing patents were dated before the focus patent's grant date. Prior to November 29, 2000 forward citations that occurred before the grant date of the focus patent were usually from the same organization or self-citing. These forward citations are typically a direct result of divisional (patent splitting) or *continuation-in-part* applications which are commonly refereed to as *family-to-family* citations. Continuation-in-part (CIP) applications occur as inventors iterate on their invention and add new inventive subject matter beyond what is stated in the original application. CIP's typically contain the same inventive content and share the original application date. They can be filed as long as the original filing application is still under review. The application review period can be extended by a *request for continued examination* or RCE. From an inventive knowledge dissemination perspective the application date is a better representation of when the inventor had enough information to reduce the invention to practice, the only drawback is a lack of clarity around when that knowledge became publicly available. Both Figure 18 and Figure 6 were changed to use the application date of the focus patent as well as the application date for all forward citations. In addition, the application date was used for the x-axis instead of the year which reduced patent bubble overlap and masking that occurred where a single heavily cited patent could cover up a less cited patent. For comparison purposes the original forward citation visualization based upon grant year can be referenced in Figure 31. Comparing Figure 18 to Figure 6 reveals similar patterns in terms of invention impact even though the first visualization is based upon forward citation counts and the latter is based upon CSS relatedness counts. There are also differences in the patterns, so it is clear that CSS relatedness is not equivalent to citation counting. It is not obvious *a priori* which approach is a more useful measure of knowledge flow, however, the fact that the intra-CSS patterns are different might be an indication of latent relationships between documents not revealed using citations. To further compare CSS relatedness and citations as proxies for knowledge flow the twelve most heavily cited patents in our corpus were compared.

3.3.8 Invention Knowledge Flow: Intra-CSS vs Intra-CIT

Figure 19 represents a method comparison of intra-CSS versus intra-citation counts over a 25-year time frame for the most cited patents in the corpus. A time of zero on the x-axis represents the official application filing date of the patent. Bars marked in orange represent counts of forward cited patents while bars marked in blue represent counts using the intra-CSS relatedness method. The difference in years for all forward patent related methods, either CSS

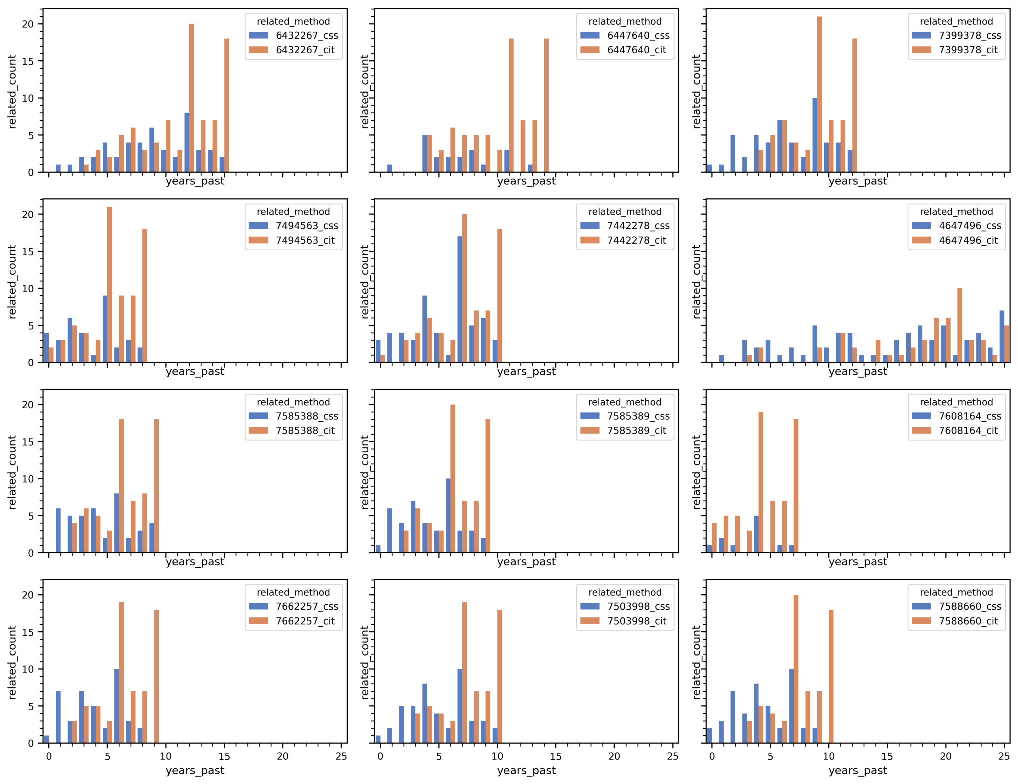


Figure 19: Inventive Knowledge Flow: Intra-CSS vs Intra-CIT for Top 12 Most Cited Patents

or cited, is calculated by taking the difference between the focus patent's application date and the application date of any related patents that follow. Only U.S. Patent 4,647,496 [122] shows forward related patents that span the entire 25-year time frame, as this patent was granted in 1984 and has since expired. Other patents exhibit truncation as they approach the last year of the data set which is 2015. For example, U.S. patent 6,432,267 [123] (located on the upper left of Figure 19) was filed in the year 2000, granted in 2002, and has no citations after 2015 (see Figure 20). In terms of knowledge flow the intra-CSS related patents demonstrate some similar patterns to counts of forward cited patents. A good example can be seen in Figure 21 which is a bar chart comparison between intra-CSS and intra-CIT methods by year. There is also dissimilarity between the methods. For example Figure 21 shows intra-CSS related counts in the years 1985, 1990-92, and 2013 with no corresponding forward citations in those years. There are also dissimilarity patterns that may be suggestive of anomalous citing behavior. For example, an interesting forward citation pattern exists within the top 12 cited patents, sans U.S. Patent 4,647,496 [122], that is not evident with intra-CSS related counts. This repeating pattern is marked by a large citation count spike followed by a two year lull and then another large spike. Figure 22 shows an example of this horn-like pattern in the years from 2012 through 2015 which is absent in the intra-CSS relatedness method, The same horn-like pattern exists for the years 2012 through 2015 for all the top cited patents except U.S. patent 4,647,496 [122]. Figure 25 is a heat map of the same top cited patents as shown in Figure 19. The x-axis lists both relatedness methods, intra-CSS and intra-CIT, for each patent while the y-axis shows elapsed years since the application filing date in reverse chronological order. The heat map shows heavy citation

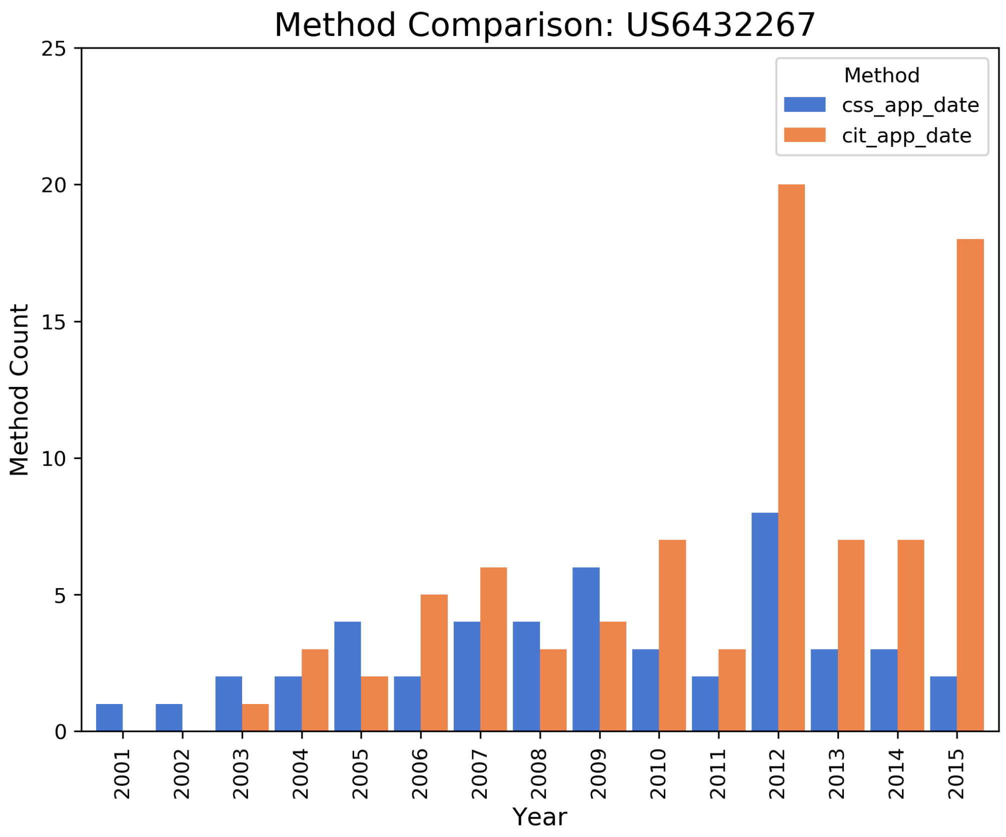


Figure 20: Inventive Knowledge Flow Comparison: U.S. Patent 4,647,496

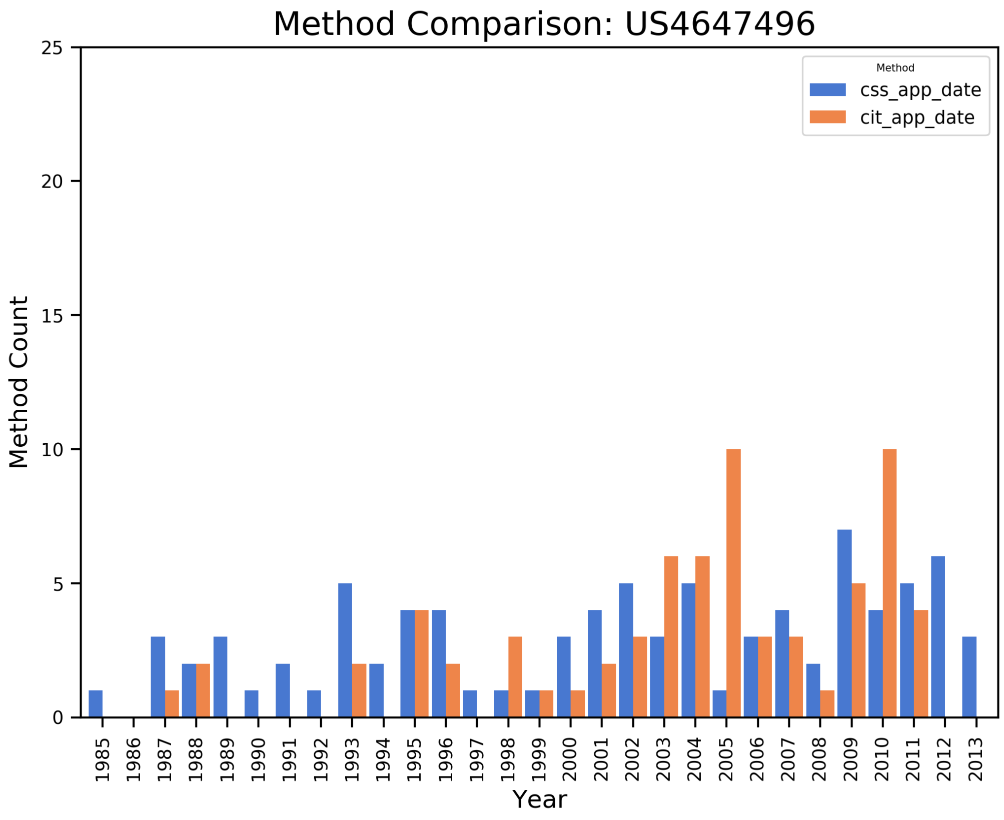


Figure 21: Inventive Knowledge Flow Comparison: U.S. Patent 4,647,496

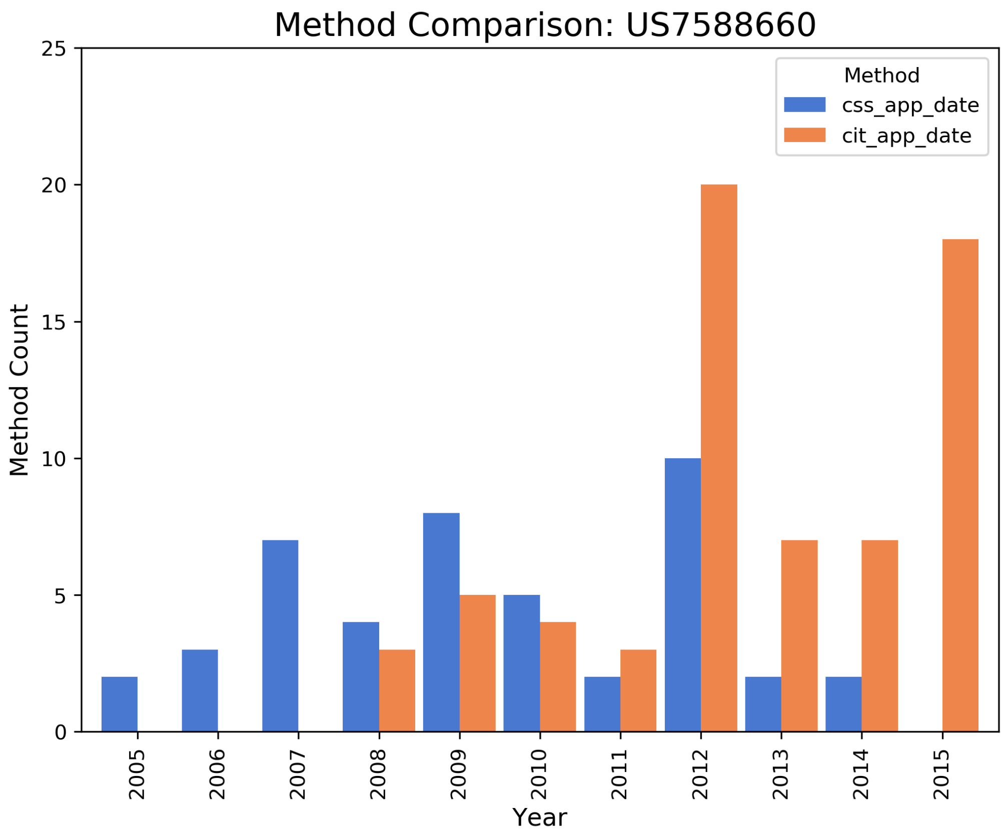


Figure 22: Inventive Knowledge Flow Comparison: U.S. Patent 7,588,660

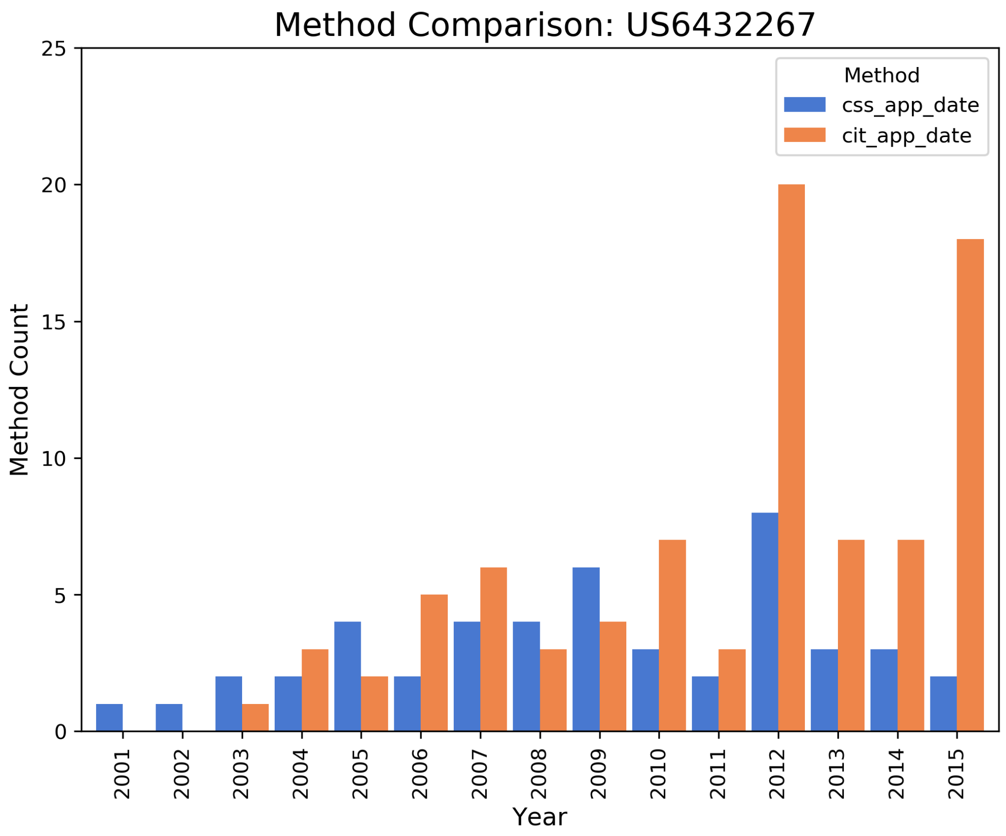


Figure 23: Inventive Knowledge Flow Comparison: U.S. Patent 6,432,267

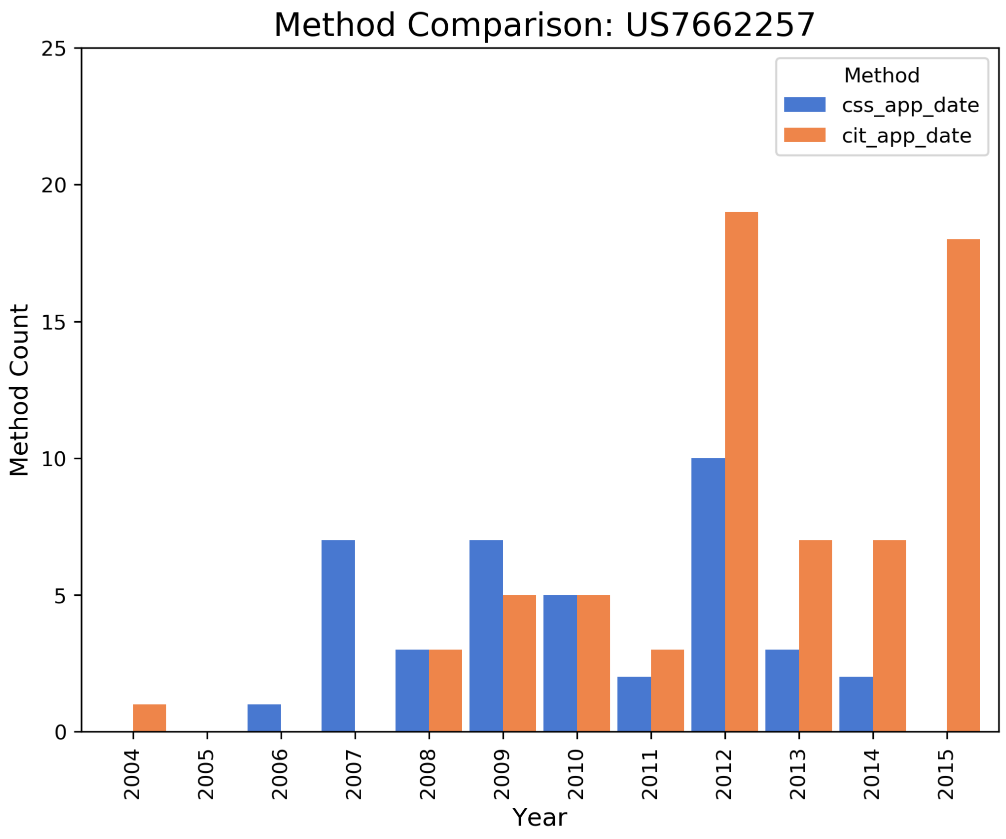


Figure 24: Inventive Knowledge Flow Comparison: U.S. Patent 7,662,257

activity in the last four years of many top cited patents with little corresponding intra-CSS activity. Areas marked in white represent when the patent reached the end of the data set which is the year 2015. Further investigation revealed that the top 12 most cited patents were all from

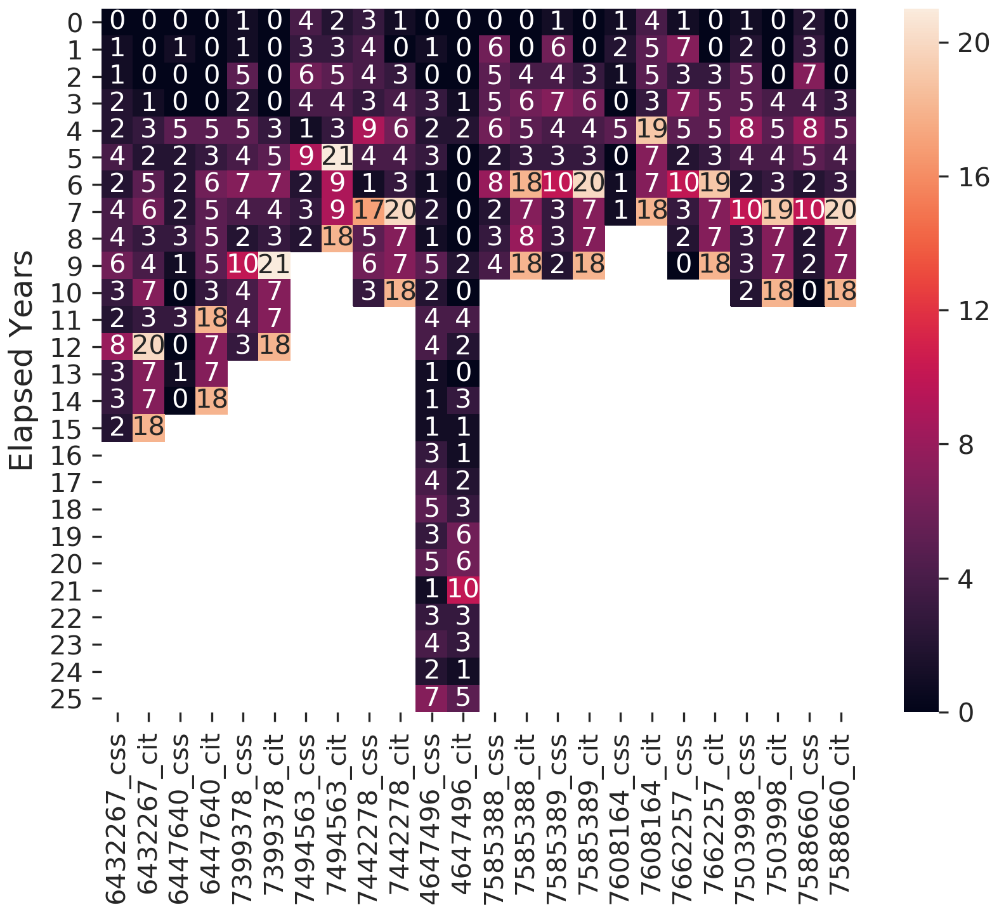


Figure 25: Inventive Knowledge Flow: Heat Map of Intra-CSS vs Intra-CIT for Top 12 Cited Patents

Georgia Pacific and that high citation counts in 2012 and 2015 were all family-to-family citations from the same subsidiary, Georgia Pacific Consumer Products (GPCP). Most of these forward citations were a result of divisional and/or continuation-in-part applications. The exception was U.S. Patent 4,647,496 [122] which was from a different subsidiary called Georgia Pacific Gypsum. This does raise the question as to why the intra-CSS method did not show a similar pattern as one would expect if the inventive language was similar enough.

3.3.9 Latent Relatedness: Intra-CSS Relatedness

To assess whether intra-CSS relatedness was in fact revealing latent relationships between patent documents the intersection between the two methods was measured. Figure 26 shows the intersection between intra-CSS and intra-CIT methods for the same top cited patents. Despite demonstrating some similar patterns in knowledge flow the methods clearly reveal different sets of forward related patents. For example only 17% of patents overlap for U.S. Patent 4,647,496 [122] between methods despite showing similar knowledge flow trends. Figure 27 shows a closer look at the overlap between methods for one of the top cited patents, U.S. Patent 7,588,660 [124] titled “Wet-pressed Tissue and Towel Products with Elevated CD Stretch and Low Tensile Ratios Made with High Solids Fabric Crepe Process”. This patent was filed on April 12, 2005 and granted on September 15, 2009. Note that U.S. Patent 7,588,661 [125] titled “Absorbent Sheet Made by Fabric Crepe Process” is listed on the left hand side of the Venn diagram as being intra-CSS related but does not show up on the right hand side as being forward cited. The application date of this patent is June 5, 2008 and it was granted on the same day as U.S. Patent 7,588,660 [124]. Closer scrutiny of the patent confirms that it does not cite

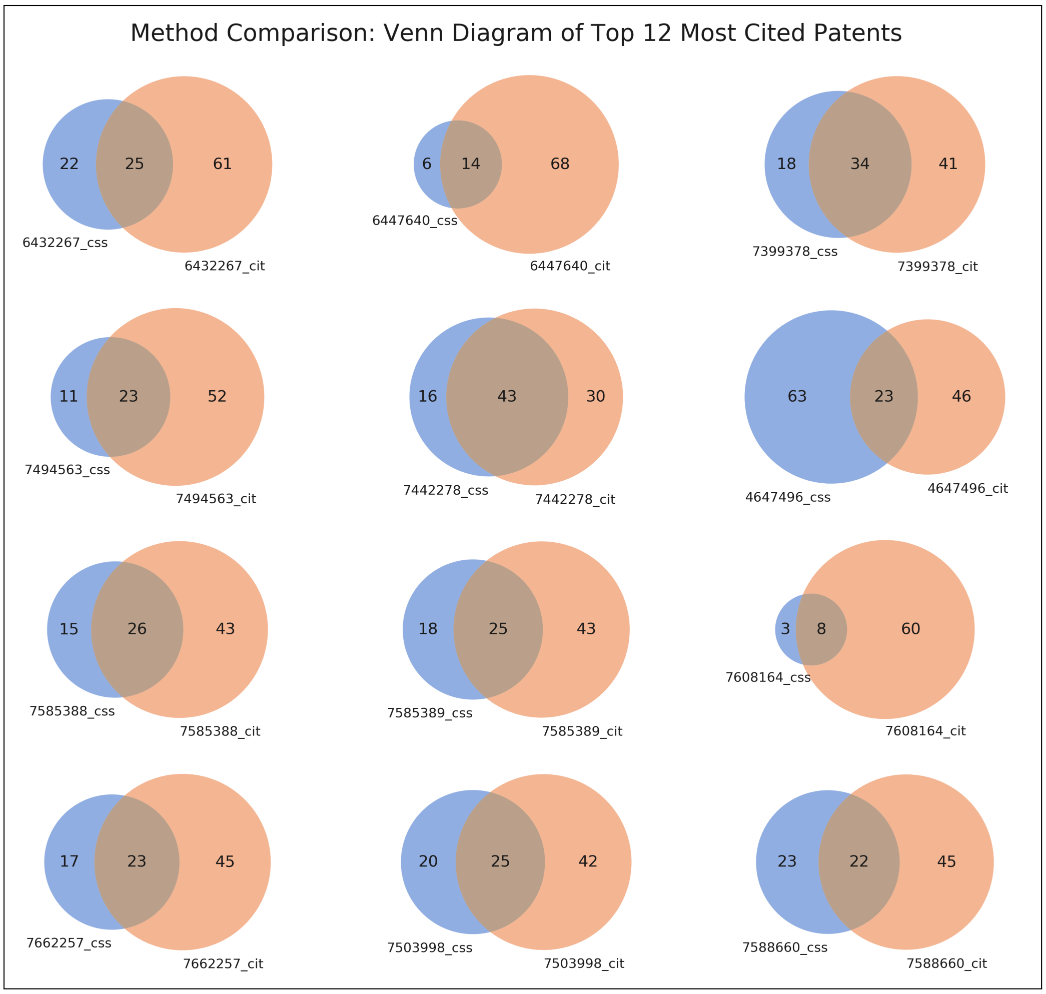


Figure 26: Inventive Knowledge Flow: VENN Intra-CSS vs Intra-CIT Top 12 Cited Patents

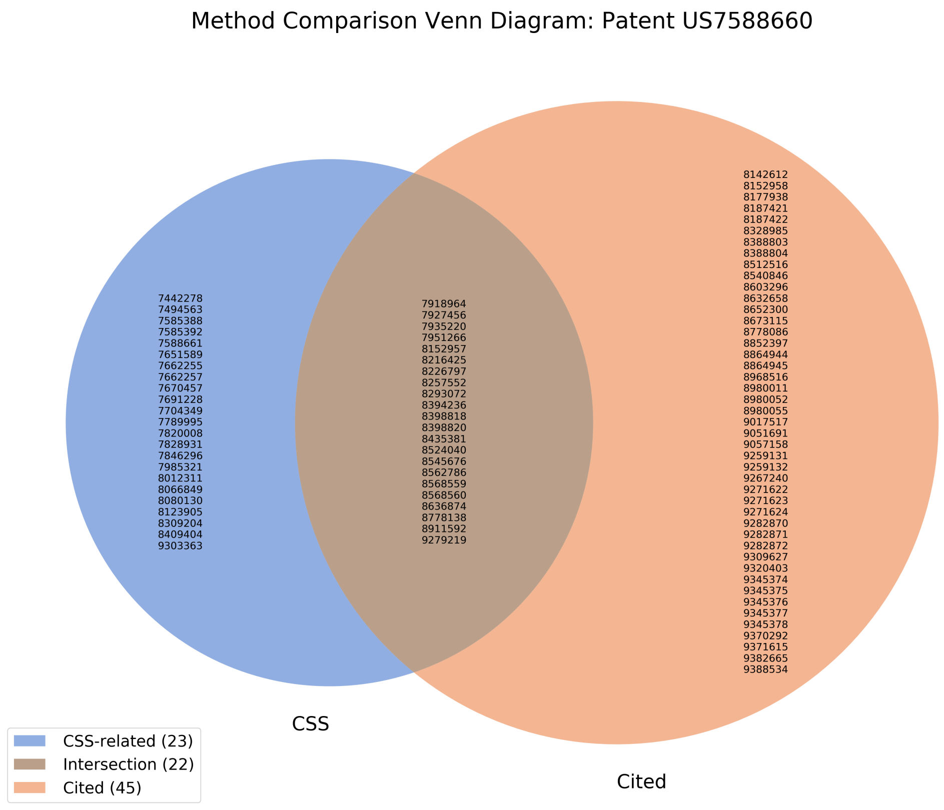


Figure 27: Method Comparison Venn Diagram: U.S. Patent 7,588,660

7,588,660 despite having consecutive patent numbers and a later application filing date. Patent 7,588,661 [125] does cite the patent application number of 7,5888,660 [124] and they both cite a common parent application for which one was a divisional and the other a continuation-in-part respectively. To further confirm that the intra-CSS related method is discovering related patents based on inventive language a patent was chosen that demonstrated only intra-CSS related counts with no citations. For example patent 6,500,493 [126] (Figure 28) from Saint-Gobain (SG) shows clear forward intra-CSS relatedness with no actual cited patents. The patent title and abstract are as follows:

> Patent US 6,500,493 - Electrostatic deposition process - Fine abrasive powders can be made more free-flowing and better adapted to electrostatic upward projection deposition in the production of coated abrasives by the control of the volume resistivity of the powder to a level that is not greater than 1014 ohms.cm by incorporation of a silica powder. [126]

The USPTO lists three citations for this patent all of which are assigned to 3M Corporation (3M). Note 3M is not part of the original set of building material companies selected for this analysis which explains why it did not show up in the citation count. The patent number, title, and abstract from the 3M patents are as follows:

> Patent US 8,551,577 - Method of electrostatic deposition of particles, abrasive grain and articles; Disclosed is a method of applying particles to a coated backing. A first layer of particles is created over a second layer of particles on a support surface and the first layer of particles is different in at least one property from the second

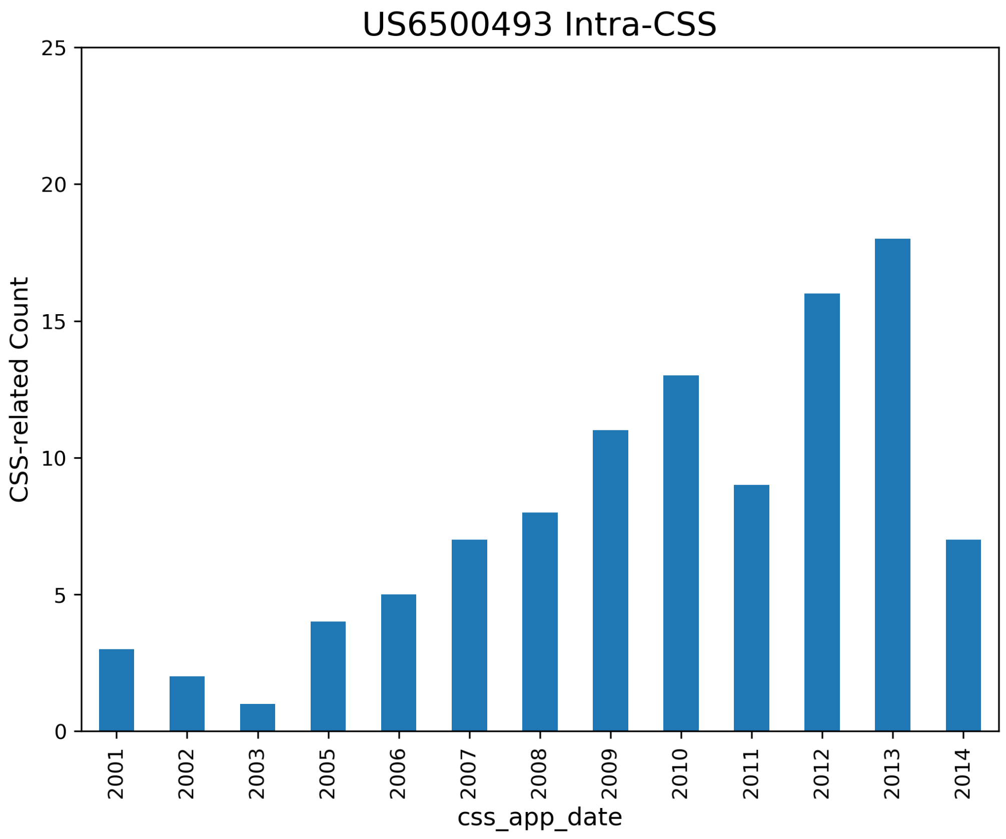


Figure 28: Method comparison Venn diagram: U.S. Patent 6,500,493

layer of particles. A coated backing is positioned above the first and second layer of particles. An electrostatic field is applied simultaneously to the first and second layer of particles such that the first layer of particles closer to the coated backing are preferentially attracted to the coated backing first before the second layer of particles. [127]

Patent US 8,869,740 - Layered particle electrostatic deposition process for making a coated abrasive article; Disclosed is a method of applying particles to a coated backing. A first layer of particles is created over a second layer of particles on a support surface and the first layer of particles is different in at least one property from the second layer of particles. A coated backing is positioned above the first and second layer of particles. An electrostatic field is applied simultaneously to the first and second layer of particles such that the first layer of particles closer to the coated backing are preferentially attracted to the coated backing first before the second layer of particles. [128]

Patent US 8,894,466 - Method of electrostatic deposition of particles, abrasive grain and articles; Presently described are methods of making an article via electrostatic deposition of particles, abrasive grains and articles, as well as a method of repairing a painted surface. The abrasive grain comprises a plurality of abrasive particles having a median primary particle size of less than 75 microns, and discrete hydrophobic nanoparticles. [129]

It is clear that the 3M patents are related to the focus patent and that the technology at hand is related to the manufacture of what appears to be sandpaper (i.e., coated abrasive article) using an electrostatic process with specific sizes of abrasive particles like silica powder.

The following are the top three intra-CSS related patents from our set including the patent number, CSS, title, and abstract:

> Patent US 9,221,151 (CSS 0.476) - Abrasive articles including a blend of abrasive grains and method of forming same, An abrasive article comprising a backing material and an abrasive layer disposed on the backing material, wherein the abrasive layer comprises a blend of abrasive particles comprising a first plurality of abrasive particles and a second plurality of abrasive particles. [130]

> Patent US 8,105,135 (CSS 0.449) - A polishing slurry includes liquid medium and particulate abrasive. The particulate abrasive includes soft abrasive particles, hard abrasive particles, and colloidal silica particles, wherein the soft abrasive particles have a Mohs hardness of not greater than 8 and the hard abrasive particles have a Mohs hardness of not less than 8, and wherein the soft abrasive particles and the hard abrasive particles are present at a weight ratio of not less than 2:1. [131]

> Patent US 8,944,893 (CSS 0.445) - Addressable bonded abrasive article has a body that includes a bond material comprising an organic material and a blend of abrasive particles. The particles include a first type of abrasive particle comprising an oxide and having a first hardness and a first toughness; a second type of abrasive particle

> comprising an oxide and having a second hardness greater than the first hardness, and the second type of abrasive particle has a second toughness less than the first toughness; and a third type of abrasive particle comprising an oxide and having a third hardness greater than the first hardness and less than the second hardness, and the third type of abrasive particle has a third toughness less than the first toughness. [132]

Based upon the title and abstract information of the intra-CSS related patents it does appear that Patent US 9,221,151 [130] and 8,944,893 [132] are similar as they both discuss creation of an abrasive article using abrasive grains (note the contextual similarity between abrasive powder, particles, and grain). Patent US 8,105,135 [131] seems to be focused on the creation of an abrasive polishing slurry. Nonetheless, the intra-CSS method has revealed patents that from an inventive knowledge perspective could be related and thus demonstrated the ability to reveal latent relationships between inventive documents that were not evident via citation analysis alone.

This research has shown that new computational techniques can be used to convert unstructured patent textual data into actionable knowledge by revealing latent relationships between patents. It accomplishes this by relying upon the language used by inventors to describe their inventions and in doing so lays the ground work to study *inventiveness* and *knowledge flow* with less dependence on domain expertise and metrics derived from patent counts and citation analysis. This approach began by extracting unstructured abstract, title, and claim textual data from a subset of patent documents selected from a competitive group of companies. Term frequency-inverse document frequency (tf-idf) was used to convert the strings of inventive descriptors associated with each patent into a representative numerical vectors within a vector space model. The cosine angle between patent vectors pairs within the higher-dimension vector space was measured. A cosine similarity threshold value was chosen to select for CSS-related patents based upon the weighted mean CSS between known cited patents. The selection of an appropriate threshold is usually dependent upon a human. This research has shown that the existing patent record can be used to establish a threshold value using statistical analysis applied to known forward cited patent pairs. To visualize patent cosine relatedness the higher-dimension vector space was reduced to a two-dimensional plot first by using principal component analysis and then t-distributed stochastic neighbor embedding. The resulting visualization was then color coded using patent office technical classifications to reveal some discrete technology class clusters. Lastly, a comparison of invention impact and inventive knowledge flow was demonstrated by by plotting patent citation data alongside cosine relatedness outputs to reveal both similar and dissimilar inventive patterns. The comparison also revealed that CSS-relatedness could potentially be used to reveal anomalous patent patterns through direct comparison of CSS-relatedness and forward citations.

3.3.10 Latent Relatedness: Google Patent Search Method Comparison

To further understand whether the CSS relatedness method was revealing latent relationships between patent documents a comparison was made of forward related patents outputs between CSS-relatedness, Forward Citations, and Google Patents. The focus patent for this analysis was

Patent US 3,935,021 for Water-resistant gypsum products [133]. The following is taken from the patent's abstract:

> Patent US 3,935,021 - The water resistance of gypsum products, such as gypsum wallboard, is improved by incorporating in the composition from which the gypsum product is made polyvinyl alcohol and wax-asphalt emulsion.

The intellectual property of this patent is focused on using additives, for example polyvinyl alcohol and wax-asphalt emulsions, to impart water resistance to gypsum drywall. The Venn diagram for this analysis can be referenced in Figure 29 where a third colored green set has been added for patents identified by **Google Patents**. Google Patents is a search engine from Google that indexes patents and patent applications from around the world [134]. It also includes related technical documents indexed from Google Scholar which is a broad search tool for scholarly literature [135].

Figure 29 lists all forward related patents using the various methods taken from the patent corpus of building product companies. Note that the Google set includes both ***Cited by*** patents and ***Similar documents***. Similar documents is a method used by Google to to find substantially similar patents which is described in U.S. Patent 9,189,482 titled **Similar document search** [136].

The following is taken from the similar document search patent's abstract:

> Patent US 9,189,482 - Described herein are methods for finding substantially similar/different sources (files and documents), and estimating similarity or difference

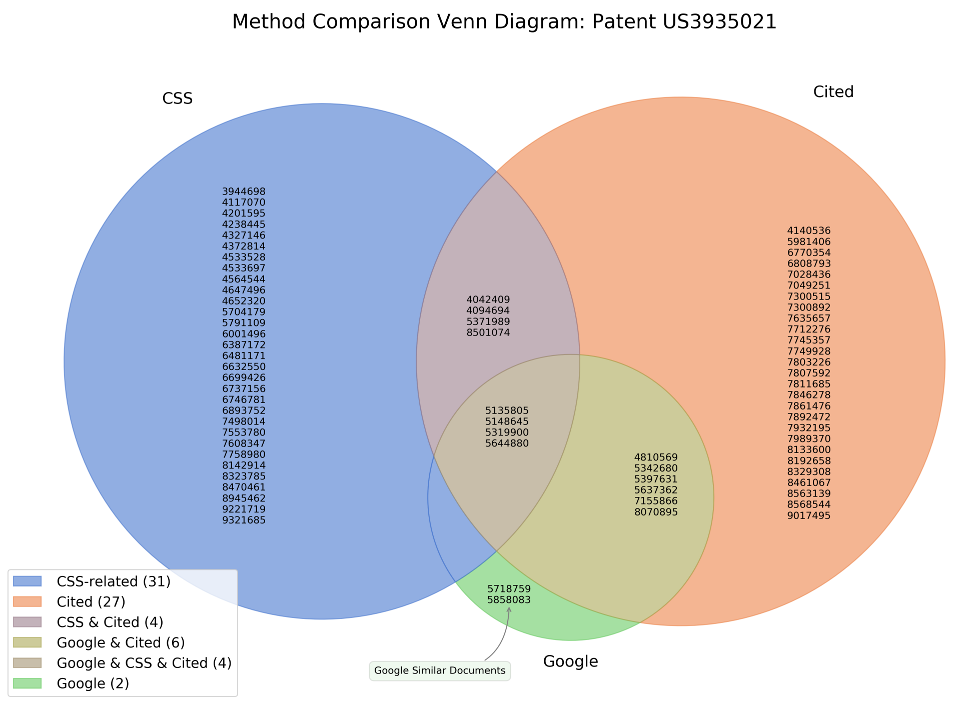


Figure 29: Method comparison Venn diagram: U.S. Patent 3,935,021

> between given sources. Similarity and difference may be found across a variety of formats. Sources may be in one or more languages such that similarity and difference may be found across any number and types of languages. A variety of characteristics may be used to arrive at an overall measure of similarity or difference including determining or identifying syntactic roles, semantic roles and semantic classes in reference to sources.

The patent authors compare document sentences via a method called language-independent semantic structure (LISS). The application for this patent mentions the use of cosine similarity but states that "such similarity measures have a drawback in that they do not actually capture the semantics"

Google similar document search identifies two patents as being similar to Patent U.S. 3,935,021. The first is Patent U.S. 5,718,759 [137] and the second is U.S. 5,858,083 [138].

> Patent U.S. 5,718,759 - A cementitious composition useful for water-resistant construction materials, including floor underlayments, backing boards, self-leveling floor materials, road patching materials, fiberboard, fire-proofing sprays, and fire-stopping materials includes about 20 wt. % to about 75 wt. % calcium sulfate beta-hemihydrate, about 10 wt. % to about 50 wt. % Portland cement, about 4 wt. % to about 20 wt. % silica fume and about 1 wt. % to about 50 wt. % pozzolanic aggregate. The Portland cement component may also be a blend of Portland cement with fly ash and/or ground blast slag.

The intellectual property of this patent is focused on using additives, for example calcium sulfate beta-hemihydrate (plaster), portland cement, and silica fume to make water-resistant materials like underlayments and self leveling floors. This patent does not cite Patent U.S. 3,935,021. The only inventive commonality between the two patents is the desire to impart water-resistance to construction materials. The key inventive descriptors of polyvinyl alcohol and wax-asphalt emulsions listed in Patent U.S. 3,935,021 are not mentioned in Patent U.S. 5,718,759.

> Patent U.S. 5,858,083 - Cementitious binders include calcium sulfate beta-hemihydrate, a cement component comprising Portland cement, and either silica fume or rice-husk ash. The silica fume or rice-husk ash component is at least about 92 wt % amorphous silica and has an alumina content of about 0.6 wt % or less.

The intellectual property of this patent focuses on the use of calcium sulfate beta-hemihydrate (plaster), portland cement, and silica fume to make water-resistant materials like underlayments and self leveling floors. It does list an additional ingredient of rice-husk ash. It does not cite Patent U.S. 3,935,021 but it does cite Patent U.S. 5,718,759 for which it has significant inventive commonality including the same assignee and authors. The key inventive descriptors used in 3,935,021 of polyvinyl alcohol and wax-asphalt emulsions are not mentioned in Patent U.S. 5,858,083.

As you can see in Figure 29 there were a number of common cited patents between those listed as forward cited by Google and those actually cited on Patent U.S. 3,935,021. In fact, all the patents listed by Google as forward cited fall within our USPTO PatentsView data set as being cited. Note that Google Patents did not list 27 patents that were actually forward citations of Patent U.S. 3,935,021.

Figure 30 lists the seventy-four forward related patents shown in Figure 29. Each patent was manually searched by a subject matter expert for any references to U.S. 3,935,021. This includes any references within the cited section as well as the body of the patent. The results of the manual review are listed under the table column marked “human”. The CSS-relatedness method identified six patents, highlighted in red font, that were verified as listed within the forward related patent text. This six patent are not listed as forward cited by either our USPTO Patentsview database or Google Patents. A possible explanation for this discrepancy, in other words, a reason as to why they were not listed by USPTO or Google Patents as being “cited by” when querying Patent U.S. 3,935,021, could be a mistake in the optical character recognition process or forward cited look-up function used by the USPTO to identify forward cited patents. Nonetheless, the CSS-Relatedness method was able to correctly identify forward cited patents that Google Patents and USPTO PatentsView had not.

patent_id	app_date	grant_date	organization	css	google	human	cited	title	abstract
3935021	11/5/1973	1/27/1976	Georgia Pacific					Water-resistant gypsum products	The water resistance of gypsum products, such as gypsum wallboard, is improved by incorporating in the composition from which the gypsum product is made polyvinyl alcohol and wax-asphalt emulsion.
3944698	11/14/1973	3/16/1976	USG Corporation	Y	N	N	N	Gypsum wallboard and process for making same	A specially prepared fiber reinforcement and improved gypsum wallboard are disclosed. The fiber reinforcement includes a multiplicity of relatively long fibers which are disposed at the interface of the core and cover sheets of the wallboard and are adhesively bonded to the cover sheets and incorporated predominantly into the portion of the core immediately adjacent to the cover sheets.
4042409	4/1/1976	8/16/1977	Yoshino Gypsum	Y	N	Y	Y	Water repellent gypsum composition	A water repellent gypsum composition comprising a gypsum, and a paraffin emulsion prepared by emulsifying (a) paraffin hydrocarbon having a melting point of 40.degree.-80.degree. C and (b) an oxidized paraffin having an acid value of 10-70 at a ratio of from 97:3 to 50:50 by weight, respectively, in the presence of a water soluble alkali compound. Optionally, a polymer emulsion or solution may be added.
4094694	5/16/1977	6/13/1978	USG Corporation	Y	N	Y	Y	Water-resistant gypsum composition and products, and process of making same	An improved water-resistant cementitious composition and products made therefrom are provided by forming an aqueous cementitious slurry, as for example of calcined gypsum and adding to the slurry a composition in the form an aqueous emulsion of asphalt and wax, a minor proportion of polyvinyl alcohol and a minor proportion of a borate compound, that is, one having an anion comprising boron and oxygen, as for example borax. The slurry is set in conventional manner by heating and drying. The resulting product has a high degree of water-resistance while utilizing less asphalt and wax composition than required with conventional asphalt-wax emulsions thereby accomplishing a large savings in raw material costs. Additionally, the use of a small amount of the borate compound permits a smaller amount of the relatively expensive polyvinyl alcohol to be used without a reduction in the water-resistance of the final product. Alternatively, an amount of asphalt and wax composition may be utilized equal to or greater than that conventionally used, but obtaining a much greater degree of water-resistance.
4117070	3/14/1977	9/26/1978	USG Corporation	Y	N	N	N	Process for preparing calcined gypsum	An improved process for producing calcined gypsum which comprises continuously treating a mass of calcined gypsum by adding, with thorough blending agitation, small metered portions of water to result in the incorporation of about 1-8% free water in the mass by weight of the gypsum, allowing the blended mass to heal the calcium sulfate hemihydrate surface fissures and thereafter continuously supplying the treated gypsum mass into gypsum board production.
4140536	3/1/1978	2/20/1979	Gypsum Industries Limited	N	N	Y	Y	Gypsum products	This invention relates to a method of making a gypsum product wherein a hot homogeneous mixture of pitch and a suitable organic material are mixed with gypsum and water to form a slurry. This slurry is then formed, allowed to set and then heated to a temperature above the melting points of both constituents of the mixture but below the temperature at which there is any significant deterioration of the product.
4201595	9/5/1978	5/6/1980	USG Corporation	Y	N	N	N	Process for preparing calcined gypsum and gypsum board	A process for preparing calcined gypsum (stucco) which comprises treating a mass of calcined gypsum by adding, with thorough blending, small portions of water (about 1-10% by weight) to the calcined gypsum, allowing it to heal, and grinding the healed stucco to recapture the rate of strength development and the ultimate strength which are adversely affected by the water addition. The principal advantage provided by the addition of small portions of water is a reduction in water demand which is retained despite the grinding and optional drying of the healed stucco. If the treated calcined gypsum is not used shortly after the healing procedure, it should be dried to provide storage stability. The reduced water demand is particularly useful in gypsum board manufacture.
4238445	7/2/1979	12/9/1980	USG Corporation	Y	N	N	N	Process for manufacturing gypsum board	A process and apparatus for producing a healed stucco having lowered water demand without loss of normal strength development potential which comprises, while blending a small amount of water with the calcined gypsum, simultaneously or substantially simultaneously grinding the calcined gypsum so as to increase the surface area of the calcined gypsum particles while incorporating about 1-10% by weight of the calcined gypsum of free water.
4327146	10/27/1980	4/27/1982	National Gypsum	Y	N	N	N	High density interface gypsum board and method for making same	A gypsum wallboard, and the method of manufacture, wherein a defoamer is disposed at the gypsum-paper interface during manufacture, causing the foam, present in the core forming gypsum slurry, to break down at the gypsum-paper interface, increasing substantially the density of the gypsum at the interface, relative to the density throughout the center portion of the gypsum core.
4372814	5/13/1981	2/8/1983	USG Corporation	Y	N	N	N	Paper having mineral filler for use in the production of gypsum wallboard	A composite paper particularly adapted for use as cover sheets in the production of gypsum wallboard, the paper being sufficiently porous to permit better drainage and more rapid drying in the production of the paper, and when applied to the surfaces of a gypsum slurry for forming wallboard, permits less heat to be utilized in the wallboard conversion, thereby saving energy in the board production required for drying the board. The paper comprises in weight percent: PA1 (A) fibers in an amount of from about 65% to about 90% and having a fiber freeness of from about 350 to 550 ml. Canadian Standard Freeness, PA1 (B) a mineral filler in an amount from about 10% to about 35%, PA1 (C) a binder in an amount from about 1% to about 31/2%, PA1 (D) a flocculant in an amount of from about 2 to about 4 lb./ton, and PA1 (E) a sizing agent in an effective amount to prevent water penetration. In an preferred embodiment the paper is treated with an internal sizing agent during its formation, and subsequently treated with a surface sizing agent after formation, in order to provide better adhesion to the gypsum core.
4533528	7/15/1983	8/6/1985	USG Corporation	Y	N	N	N	Process for continuously calcining gypsum to low dispersed consistency stucco	Wet chemical gypsum cake may be directly fed to a continuous kettle calciner to produce a stucco having lowered dispersed consistency. Pre-drying of the chemical gypsum is eliminated and the thus produced stucco may be used in the formulation of building plasters and in gypsum wallboard manufacture that will use less fuel for drying excess gauging water.
4533697	2/18/1983	8/6/1985	Saint Gobain	Y	N	N	N	Process for preparing polyvinyl butyral	The invention relates to a process for preparing polyvinyl butyral comprising simultaneously introducing into a reaction medium containing a mixture of water and a portion of reactive polyvinyl alcohol, maintained initially at a temperature below about 20.degree. C., a stream of an aqueous solution of polyvinyl alcohol corresponding to the complement of the reactive polyvinyl alcohol and a stream of butyraldehyde and cooling the reaction mixture during the introduction of the polyvinyl alcohol and butyraldehyde. The reaction is also carried out in the presence of an acid catalyst and an emulsifying agent. The polyvinyl butyral obtained is used, after plasticizing, as an interlayer in laminated glass.
4564544	12/1/1983	1/14/1986	National Gypsum	Y	N	N	N	Fire-resistant gypsum board	A fire-retardant gypsum wallboard having, in the gypsum core, about 2% by weight of a feldspar-free muscovite, in combination with other minor additives including 1/2 inch long glass fibers.
4647496	2/27/1984	3/3/1987	Georgia Pacific	Y	N	Y	N	Use of fibrous mat-faced gypsum board in exterior finishing systems for buildings	An exterior finishing system for a building, including particularly an exterior insulation system, which includes a fibrous mat-faced gypsum board, preferably a board in which the set gypsum core thereof is water resistant, and preferably one in which the set gypsum core is sandwiched between two sheets of porous glass mat, with the outer surface of at least one of said mats being substantially free of set gypsum, and means for preparing the board, including control of the viscosity of the aqueous gypsum slurry from which the set gypsum core of the board is formed. Also, the use of fibrous mat-faced gypsum board as the shaft liner panel in a shaft wall assembly.
4652320	5/13/1985	3/24/1987	Saint Gobain	Y	N	N	N	Process for making polyvinyl butyral glass laminates	The invention relates to a process for preparing polyvinyl butyral comprising simultaneously introducing into a reaction medium containing a mixture of water and a portion of reactive polyvinyl alcohol, maintained initially at a temperature below about 20.degree. C., a stream of an aqueous solution of polyvinyl alcohol corresponding to the complement of the reactive polyvinyl alcohol and a stream of butyraldehyde and cooling the reaction mixture during the introduction of the polyvinyl alcohol and butyraldehyde. The reaction is also carried out in the presence of an acid catalyst and an emulsifying agent. The polyvinyl butyral obtained is used, after plasticizing, as an interlayer in laminated glass.
4810569	3/2/1987	3/7/1989	Georgia Pacific	N	Y	Y	Y	Fibrous mat-faced gypsum board	An exterior finishing system for a building. including particularly an exterior insulation system, which includes a fibrous mat-faced gypsum board, preferably a board in which the set gypsum core thereof is water resistant, and preferably one in which the set gypsum core is sandwiched between two sheets of porous glass mat, with the outer surface of at least one of said mats being substantially free of set gypsum, and means for preparing the board, including control of the viscosity of the aqueous gypsum slurry from which the set gypsum core of the board is formed. Also, the use of fibrous mat-faced gypsum board as the shaft liner panel in a shaft wall assembly.
5135805	7/27/1990	8/4/1992	Georgia Pacific	Y	Y	Y	Y	Method of manufacturing a water-resistant gypsum composition	A method for incorporating siloxane into a water-resistant, gypsum-based article is disclosed, comprising, (A) adding said siloxane to water; (B) mixing said siloxane/water mixture with calcined gypsum to form an aqueous slurry; and (C) allowing said slurry to set to form a set gypsum-based, water-resistant article.

Figure 30: All forward related patents to U.S. 3,935,021

patent_id	app_date	grant_date	organization	css	google	human	cited	title	abstract
5148645	8/6/1991	9/22/1992	Georgia Pacific	Y	Y	Y	Y	Use of fibrous mat-faced gypsum board in shaft wall assemblies and improved fire resistant board	Glass mat-faced gypsum boards having a set core containing gypsum dihydrate and at least a minimum amount of chopped fibers, as well as other optional fire-resistant additives, are provided. The resulting boards obtain superior fire resistance properties over conventional paper-faced boards of like thickness which include a similar amount of chopped glass fibers.
5319900	5/6/1993	6/14/1994	Georgia Pacific	Y	Y	Y	Y	Finishing and roof deck systems containing fibrous mat-faced gypsum boards	Finishing systems and roof decks are provided which include a gypsum board having a set gypsum core faced with a fibrous mat. The gypsum core includes one or more additives which are effective in simultaneously improving the water and fire resistance of the board. In preferred embodiments, the board has sufficient water-resistant additive for absorbing less than about 10% water in an ASTM C-473 test.
5342680	10/15/1993	8/30/1994	Georgia Pacific	N	Y	Y	Y	Glass mat with reinforcing binder	In gypsum board faced with a fibrous mat, for example, a mat of glass filaments adhesively bound together, improvements are realized by the use of a reinforcing resinous binder in the mat.
5371989	2/19/1992	12/13/1994	Georgia Pacific	Y	N	Y	Y	Use of fibrous mat-faced gypsum board in exterior finishing systems for buildings and shaft wall assemblies	An exterior finishing system for a building, including particularly an exterior insulation system, which includes a fibrous mat-faced gypsum board, preferably a board in which the set gypsum core thereof is water resistant, and preferably one in which the set gypsum core is sandwiched between two sheets of porous glass mat, with the outer surface of at least one of said mats being substantially free of set gypsum, and means for preparing the board, including control of the viscosity of the aqueous gypsum slurry from which the set gypsum core of the board is formed. Also, the use of fibrous mat-faced gypsum board as the shaft liner panel in a shaft wall assembly.
5397631	7/19/1993	3/14/1995	Georgia Pacific	N	Y	Y	Y	Coated fibrous mat faced gypsum board resistant to water and humidity	A fibrous mat-faced gypsum board is coated with a water-resistant resinous coating.
5637362	6/7/1995	6/10/1997	Louisiana Pacific	N	Y	Y	Y	Thin, sealant-coated, fiber-reinforced gypsum panel	A process for making a thin, sealant-coated, fire and indentation resistant gypsum panel. A layer of a mixture of cellulosic fibers moistened with water and an additive to restrict the adhesion of cellulosic fibers to one another, is deposited on at least two separate conveyors to form a fibrous matt on each conveyor. A layer of dry calcined gypsum containing set accelerator is deposited on the fibrous matt on each conveyor which contains, by weight on a dry weight basis, about 70-90% gypsum, about 10%-19% cellulosic fiber and about 1%-9% of combined additive and set accelerator. Each layer on each conveyor is directed to a mixing station to form a homogeneous layer, and the formed homogeneous layers on separate conveyors are placed atop one another on a single conveyor to form a combined homogeneous layer. Water is added to the layer or layers to rehydrate calcined gypsum. The combined homogeneous layer is subjected to pressure to form a pressed, homogeneous fiber-reinforced gypsum panel, and dried. The surface of the dried panel is sealed with wax-free, water resistant sealant. Sealant-coated, fire resistant, indentation resistant gypsum panels having a thickness of about 0.25 inch and a density of about 60 to 80 lbs./ft..sup.3 are made from 2 homogeneous layers of fiber-reinforced gypsum.
5644880	6/7/1995	7/8/1997	Georgia Pacific	Y	Y	Y	Y	Gypsum board and systems containing same	Finishing systems and roof decks are provided which include a gypsum board having a set gypsum core faced with a fibrous mat. The gypsum core includes one or more additives which are effective in simultaneously improving the water and fire resistance of the board. In preferred embodiments, the board has sufficient water-resistant additive for absorbing less than about 10% water in an ASTM C-473 test.
5704179	1/26/1994	1/6/1998	Georgia Pacific	Y	N	Y	N	Finishing and roof deck systems containing fibrous mat-faced gypsum boards	Finishing systems and roof decks are provided which include a gypsum board having a set gypsum core faced with a fibrous mat. The gypsum core includes one or more additives which are effective in simultaneously improving the water and fire resistance of the board. In preferred embodiments, the board has sufficient water-resistant additive for absorbing less than about 10% water in an ASTM C-473 test.
5718759	10/18/1996	2/17/1998	National Gypsum	N	Y	N	N	Cementitious gypsum-containing compositions and materials made therefrom	A cementitious composition useful for water-resistant construction materials, including floor underlayments, backing boards, self-leveling floor materials, road patching materials, fiberboard, fire-proofing sprays, and fire-stopping materials includes about 20 wt. % to about 75 wt. % calcium sulfate beta-hemihydrate, about 10 wt. % to about 50 wt. % Portland cement, about 4 wt. % to about 20 wt. % silica fume and about 1 wt. % to about 50 wt. % pozzolanic aggregate. The Portland cement component may also be a blend of Portland cement with fly ash and/or ground blast slag.
5791109	11/6/1996	8/11/1998	Georgia Pacific	Y	N	Y	N	Gypsum board and finishing system containing same	Finishing systems and roof decks are provided which include a gypsum board having a set gypsum core faced with a fibrous mat. The gypsum core includes one or more additives which are effective in simultaneously improving the water and fire resistance of the board. In preferred embodiments, the board has sufficient water-resistant additive for absorbing less than about 10% water in an ASTM C-473 test.
5858083	5/19/1997	1/12/1999	National Gypsum	N	Y	N	N	Cementitious gypsum-containing binders and compositions and materials made therefrom	Cementitious binders include calcium sulfate beta-hemihydrate, a cement component comprising Portland cement, and either silica fume or rice-husk ash. The silica fume or rice-husk ash component is at least about 92 wt % amorphous silica and has an alumina content of about 0.6 wt % or less.
5981406	1/23/1998	11/9/1999	Georgia Pacific	N	N	Y	Y	Glass mat with reinforcing binder	In a gypsum board faced with a fibrous mat, for example, a mat of glass filaments adhesively bound together, improvements are realized by the use of a reinforcing binder in the mat.
6001496	8/16/1996	12/14/1999	Georgia Pacific	Y	N	N	N	Mat-faced gypsum board and method of manufacturing same	Disclosed is a mat-faced gypsum board of the general type having a gypsum-based core formed from a gypsum slurry compressed through an extrusion ratio of at least about 3:1 wherein the fibrous mat facing on at least one side of the board has a weight per unit surface area of greater than about 1.85 lb./100 ft..sup.2 and consists essentially of inorganic fibers having a diameter of less than about 15 microns. Also disclosed are methods for manufacturing such a gypsum board and the use of such a gypsum board in exterior finishing systems, interior lath systems and as a door core.
6387172	4/25/2000	5/14/2002	USG Corporation	Y	N	N	N	Gypsum compositions and related methods	A set gypsum composition and methods for the preparation thereof are disclosed. The set gypsum composition comprises a continuous phase of interlocking set gypsum matrix having an enhanced water voids volume and/or is prepared from a mixture (e.g., slurry) comprising an elevated ratio of water to calcined gypsum. Also disclosed is an article comprising the set gypsum composition.
6481171	11/1/2001	11/19/2002	USG Corporation	Y	N	N	N	Gypsum compositions and related methods	A set gypsum composition and methods for the preparation thereof are disclosed. The set gypsum composition comprises a continuous phase of interlocking set gypsum matrix having an enhanced water voids volume and/or is prepared from a mixture (e.g., slurry) comprising an elevated ratio of water to calcined gypsum. Also disclosed is an article comprising the set gypsum composition.
6632550	2/16/1999	10/14/2003	USG Corporation	Y	N	N	N	Gypsum-containing product having increased resistance to permanent deformation and method and composition for producing it	The invention provides a set gypsum-containing product having increased resistance to permanent deformation and a method for preparing it comprising forming a mixture of a calcium sulfate material, water, and an appropriate amount of one or more enhancing materials chosen from condensed phosphoric acids, each of which comprises 2 or more phosphoric acid units; and salts or ions of condensed phosphates, each of which comprises 2 or more phosphate units. The mixture is then maintained under conditions sufficient for the calcium sulfate material to form a set gypsum material.
6699426	5/10/2000	3/2/2004	National Gypsum	Y	N	N	N	Gypsum wallboard core, and method and apparatus for making the same	A gypsum wallboard core, and methods and apparatus for making the same are disclosed. Methods of making a gypsum wallboard core include extruding a gypsum slurry containing water, gypsum, slip agents, water-reducing agents, surfactants and, optional additives, through a die and onto a substantially flat, smooth, moving surface. The die has provisions at its outer sides for the introduction of slip agents into the slurry, and provisions at its lateral outer edges for the introduction of a strength-enhancing agent. Once extruded onto the conveyor belt, the slurry is chemically-activated to set and form a hardened board core which then may be easily removed from the conveyor belt and dried.
6737156	9/18/2002	5/18/2004	Georgia Pacific	Y	N	Y	N	Interior wallboard and method of making same	A gypsum wallboard may have a paper-covered first face with shaped regions formed along side portions near the wallboard edges, and a fibrous mat-covered second face. The fibrous mat material covering the second face extends around the wallboard edges and is overlapped by portions of the paper on the first face. The wallboard can be manufactured by depositing a gypsum slurry onto a moving web of the fibrous mat material, applying a web of the paper to the deposited gypsum slurry, and forming shaped regions in the side portions of the top surface.

Figure 30: **Continued:** All forward related patents to U.S. 3,935,021

patent_id	app_date	grant_date	organization	css	google	human	cited	title	abstract
6746781	8/21/2002	6/8/2004	Georgia Pacific	Y	N	Y	N	Gypsum board having polyvinyl alcohol binder in interface layer and method for making the same	Paper and/or mat-faced gypsum board is prepared by applying a relatively thin coating of aqueous gypsum slurry containing a polyvinyl alcohol binder to one or two facer sheets. The polyvinyl alcohol binder provides adequate adhesion between the set gypsum core and the adjacent facer sheet(s) without the need for starch or other conventional binders. In one embodiment, polyvinyl alcohol is concentrated in one or more regions of the core adjacent to the facer sheet(s). In another embodiment, polyvinyl alcohol is applied to an aqueous gypsum slurry used to form the bulk core, such that the polyvinyl alcohol is present throughout the core.
6770354	4/19/2001	8/3/2004	Georgia Pacific	N	N	Y	Y	Mat-faced gypsum board	A moisture-tolerant structural panel comprising a gypsum board comprising a set gypsum core sandwiched between and faced with mats of glass fibers, wherein a free surface of one of said mats is coated with a combination of a mineral pigment, an inorganic adhesive binder and a polymer latex adhesive binder applied to said surface as an aqueous coating composition, said aqueous coating composition upon drying and setting, covering said mat to the extent that substantially none of the fibers of said mat protrude from said coating.
6808793	2/21/2003	10/26/2004	Georgia Pacific	N	N	Y	Y	Pre-coated mat-faced gypsum board	A moisture-tolerant structural panel comprising a gypsum board comprising a set gypsum core sandwiched between and faced with mats of glass fibers, wherein a free surface of one of said mats is coated with a combination of a mineral pigment, an inorganic adhesive binder and a polymer latex adhesive binder applied to said surface as an aqueous coating composition, said aqueous coating composition upon drying and setting, covering said mat to the extent that substantially none of the fibers of said mat protrude from said coating.
6893752	6/28/2002	5/17/2005	USG Corporation	Y	N	N	N	Mold-resistant gypsum panel and method of making same	A mold-resistant gypsum panel includes a core of an interlocking matrix of calcium sulfate dihydrate crystals, a facing material on at least one side of the panel and a salt of pyrithione dispersed through both the core and the facing materials. A method of making a mold-resistant gypsum product is also provided. A slurry of calcined gypsum, water and a water-soluble pyrithione salt is formed, then deposited on a sheet of facing material. The slurry on the facing material is shaped into a panel and maintained under conditions sufficient for the calcined gypsum to react with the water to form a core comprising an interlocking matrix of set gypsum crystals. Heating of the panel causes evaporation of the water that did not react with the calcined gypsum.
7028436	11/5/2002	4/18/2006	CertainTeed	N	N	Y	Y	Cementitious exterior sheathing product with rigid support member	Cementitious exterior sheathing products are provided which include a rigid support member affixed to a cementitious layer. The rigid support member includes at least one nailing flange disposed along one of its lateral sides for allowing the sheathing product to be affixed to an exterior wall of a building. Preferred mechanical and adhesive bonding techniques are suggested for combining the cementitious layer and rigid support member together to form an integrated product. Such products are lighter in weight and are more crack resistant than currently available fiber cement trim boards.
7049251	1/21/2003	5/23/2006	Saint Gobain	N	N	Y	Y	Facing material with controlled porosity for construction boards	This invention provides facing materials for cementitious boards such as those including Portland cement or gypsum cores. The preferred facing material includes, in a first embodiment, a facing layer having an areal weight of about 300 grams/M, and an air permeability rating of no greater than about 300 CFM/ft(FG 436-910 test method). The facing layer reduces the penetration of a slurry of cementitious material during the manufacture of a cementitious board, while permitting the water vapor from the slurry to pass therethrough. The facing materials of this invention can be designed to substantially eliminate the fouling of rolls used in continuous processing of such boards without the use, or with greatly reduced use, of costly viscosity control agents in the slurry. In addition, further embodiments of this invention can include binders, coatings or saturants which are designed to decrease pore size, increase or decrease the contact angle of liquids, or promote greater adhesion to cementitious cores, greater adhesion to other layers in the facing material, or greater adhesion or affinity to various types of adhesive compositions used to join cementitious boards to insulation and exterior finishing systems (EIS or EIFS).
7155866	1/15/2003	1/2/2007	CertainTeed	N	Y	Y	Y	Cementitious exterior sheathing product having improved interlaminar bond strength	The present invention provides exterior building products, such as roofing and siding, shake, shingles, siding, sheathing, panels, planks, vertical siding, soffit panels, fencing, decking, fascia, corner posts, column corners and trim boards in which a plurality of cementitious layers are provided with an improved interlaminar bond by employing a resinous bond promoter, a rheological agent, mechanical means to distribute fibers in a direction which is perpendicular to the machine direction so as to bridge between layers in the product, or a combination thereof. These techniques help to increase interlaminar bond strength to improve the mechanical properties of the product. When certain resinous bond promoters are used, the additional benefits of water absorption resistance and pigmentation throughout the product can be provided with minimal expense. Improvements in interlaminar bond strength of about 10ÃƒÂƒÃ,ÂƒÃƒÂ,Ã,ÂƒÃƒÂƒÃ,Â,ÃƒÂ,Ã,Â,ÃƒÂƒÃ,ÂƒÃƒÂ,Ã,Â,ÃƒÂƒÃ,Â,ÃƒÂ,Ã,Â–46% were observed with a percent elongation improvement of about 7%.
7300515	11/16/2005	11/27/2007	Saint Gobain	N	N	Y	Y	Facing material with controlled porosity for construction boards	This invention provides facing materials for cementitious boards such as those including Portland cement or gypsum cores. The preferred facing material includes, in a first embodiment, a facing layer having an areal weight of about 300 grams/M, and an air permeability rating of no greater than about 300 CFM/ft(FG 436-910 test method). The facing layer reduces the penetration of a slurry of cementitious material during the manufacture of a cementitious board, while permitting the water vapor from the slurry to pass therethrough. The facing materials of this invention can be designed to substantially eliminate the fouling of rolls used in continuous processing of such boards without the use, or with greatly reduced use, of costly viscosity control agents in the slurry. In addition, further embodiments of this invention can include binders, coatings or saturants which are designed to decrease pore size, increase or decrease the contact angle of liquids, or promote greater adhesion to cementitious cores, greater adhesion to other layers in the facing material, or greater adhesion or affinity to various types of adhesive compositions used to join cementitious boards to insulation and exterior finishing systems (EIS or EIFS).
7300892	11/16/2005	11/27/2007	Saint Gobain	N	N	Y	Y	Facing material with controlled porosity for construction boards	This invention provides facing materials for cementitious boards such as those including Portland cement or gypsum cores. The preferred facing material includes, in a first embodiment, a facing layer having an areal weight of about 300 grams/M, and an air permeability rating of no greater than about 300 CFM/ft(FG 436-910 test method). The facing layer reduces the penetration of a slurry of cementitious material during the manufacture of a cementitious board, while permitting the water vapor from the slurry to pass therethrough. The facing materials of this invention can be designed to substantially eliminate the fouling of rolls used in continuous processing of such boards without the use, or with greatly reduced use, of costly viscosity control agents in the slurry. In addition, further embodiments of this invention can include binders, coatings or saturants which are designed to decrease pore size, increase or decrease the contact angle of liquids, or promote greater adhesion to cementitious cores, greater adhesion to other layers in the facing material, or greater adhesion or affinity to various types of adhesive compositions used to join cementitious boards to insulation and exterior finishing systems (EIS or EIFS).
7498014	1/12/2007	3/3/2009	CertainTeed	Y	N	N	N	System and method for the production of alpha type gypsum using heat recovery	The present invention relates to a system and associated method for the production of gypsum in manufacturing plant. More specifically, the invention relates to the production of alpha-type gypsum in a gypsum board manufacturing plant. The system yields increased efficiencies by capturing heat given off during processing steps and using that heat to reduce the energy needed for calcination. The invention finds particular application in the production alpha-type gypsum. The present invention is described in greater detail hereinafter in conjunction with the following specific embodiments.
7553780	12/12/2003	6/30/2009	Georgia Pacific	Y	N	Y	N	Gypsum panel having UV-cured moisture resistant coating and method for making the same	A fibrous mat faced gypsum panel having on at least one of the facing sheets a moisture resistant, cured coating of a radiation curable, e.g., UV curable, polymer.
7608347	6/9/2006	10/27/2009	USG Corporation	Y	N	N	N	Modifiers for gypsum slurries and method of using them	An improved gypsum slurry that includes water, calcium sulfate hemihydrate, a polycarboxylate dispersant and a modifier. The modifier is chemically configured to improve the efficacy of the polycarboxylate dispersant. Preferred modifiers include cement, lime, slaked lime, soda ash, carbonates, silicates and phosphates.
7635657	4/25/2005	12/22/2009	Georgia Pacific	N	N	Y	Y	Interior wallboard and method of making same	A gypsum wallboard suitable for Level 4 finishing having a coated non-woven first glass fiber mat facing material on one major surface and an optionally coated second glass fiber mat where on the other major surface. The first glass fiber mat has a majority of fibers of a nominal fiber diameter between 8 and 11 microns and a fiber length between and Ãinch and has a basis weight between about 1.7 lb./100 ft.and about 2.0 lb./100; the second glass fiber mat has a majority of fibers of a nominal fiber diameter of at least 13 microns but no greater than about 16 microns and a fiber length between and 1 inch and has a basis weight between about 1.8 lb./100 ft.and about 2.2 lb./100, and wherein the fibers in both of the non-woven glass fiber mats are bound together with an acrylic-type adhesive binder.

Figure 30: **Continued:** All forward related patents to U.S. 3,935,021

patent_id	app_date	grant_date	organization	css	google	human	cited	title	abstract
7712276	3/30/2005	5/11/2010	CertainTeed	N	N	Y	Y	Moisture diverting insulated siding panel	A siding panel product is provided comprising a first polymeric siding panel having a butt end and a top end, a front surface comprising a plurality of front faces defined between the top and butt ends and separated by at least one shoulder surface to define a stepped contour, and a rear surface. An insulation backing is coupled to the rear surface of the siding panel. The insulation backing comprises at least first and second insulation members coupled to the rear surface of said siding panel. The first insulation member has a bottom edge thereof coupled proximate to the stepped contour and the second insulation member has a top edge thereof coupled proximate to the stepped contour.
7745357	3/12/2004	6/29/2010	Georgia Pacific	N	N	Y	Y	Use of pre-coated mat for preparing gypsum board	A gypsum board which comprises a set gypsum core sandwiched between and faced with fibrous mats, wherein a free surface of one of said mats is pre-coated with a combination of a mineral pigment, optionally an inorganic adhesive binder and an organic binder, preferably a hydrophobic, UV resistant polymer latex adhesive binder applied to said surface as an aqueous coating composition, said aqueous coating composition upon drying and setting providing a pre-coated mat satisfying certain morphology requirements.
7749928	4/22/2009	7/6/2010	Georgia Pacific	N	N	Y	Y	Use of pre-coated mat for preparing gypsum board	A gypsum board which comprises a set gypsum core sandwiched between and faced with fibrous mats, wherein a free surface of one of said mats is pre-coated with a combination of a mineral pigment, optionally an inorganic adhesive binder and an organic binder, preferably a hydrophobic, UV resistant polymer latex adhesive binder applied to said surface as an aqueous coating composition, said aqueous coating composition upon drying and setting providing a pre-coated mat satisfying certain morphology requirements.
7758980	8/12/2008	7/20/2010	USG Corporation	Y	N	N	N	Gypsum-containing board and tile, and method for producing same	The invention provides a set gypsum-containing product having increased resistance to permanent deformation and a method for preparing it comprising forming a mixture of a calcium sulfate material, water, and an appropriate amount of one or more enhancing materials chosen from condensed phosphoric acids, each of which comprises 2 or more phosphoric acid units; and salts or ions of condensed phosphates, each of which comprises 2 or more phosphate units. The mixture is then maintained under conditions sufficient for the calcium sulfate material to form a set gypsum material.
7803226	7/29/2005	9/28/2010	USG Corporation	N	N	Y	Y	Siloxane polymerization in wallboard	Polymerization of siloxane is improved using a gypsum-based slurry that includes stucco, Class C fly ash, magnesium oxide and an emulsion of siloxane and water. This slurry is used in a method of making water-resistant gypsum articles that includes making an emulsion of siloxane and water, then combining the slurry with a dry mixture of stucco, magnesium oxide and Class C fly ash. The slurry is then shaped as desired and the stucco is allowed to set and the siloxane polymerizes. The resulting product is useful for making a water-resistant gypsum panel having a core that includes interwoven matrices of calcium sulfate dihydrate crystals and a silicone resin, where the interwoven matrices have dispersed throughout them a catalyst comprising magnesium oxide and components from a Class C fly ash.
7807592	10/28/2009	10/5/2010	Georgia Pacific	N	N	Y	Y	Interior wallboard and method of making same	A gypsum wallboard suitable for Level 4 finishing having a coated non-woven first glass fiber mat facing material on one major surface and an optionally coated second glass fiber mat where on the other major surface. The first glass fiber mat has a majority of fibers of a nominal fiber diameter between 8 and 11 microns and a fiber length between and inch and has a basis weight between about 1.7 lb./100 ft.and about 2.0 lb./100; the second glass fiber mat has a majority of fibers of a nominal fiber diameter of at least 13 microns but no greater than about 16 microns and a fiber length between ¾ and 1 inch and has a basis weight between about 1.8 lb./100 ft.and about 2.2 lb./100, and wherein the fibers in both of the non-woven glass fiber mats are bound together with an acrylic-type adhesive binder.
7811685	11/12/2009	10/12/2010	USG Corporation	N	N	Y	Y	Siloxane polymerization in wallboard	Polymerization of siloxane is improved using a gypsum-based slurry that includes stucco, Class C fly ash, magnesium oxide and an emulsion of siloxane and water. This slurry is used in a method of making water-resistant gypsum articles that includes making an emulsion of siloxane and water, then combining the slurry with a dry mixture of stucco, magnesium oxide and Class C fly ash. The slurry is then shaped as desired and the stucco is allowed to set and the siloxane polymerizes. The resulting product is useful for making a water-resistant gypsum panel having a core that includes interwoven matrices of calcium sulfate dihydrate crystals and a silicone resin, where the interwoven matrices have dispersed throughout them a catalyst comprising magnesium oxide and components from a Class C fly ash.
7846278	10/29/2003	12/7/2010	Saint Gobain	N	N	Y	Y	Methods of making smooth reinforced cementitious boards	A composite fabric for use in reinforcement of cementitious boards and similar prefabricated building wall panels. The fabric includes an open mesh first component of continuously coated, high modulus of elasticity strands and a nonwoven second component fabricated from alkali resistant thermoplastic material. The high modulus strands of the first component are preferably bundled glass fibers encapsulated by alkali and water resistant thermoplastic material. The composite fabric also has suitable physical characteristics for embedment within the cement matrix of the panels or boards closely adjacent the opposed faces thereof. The reinforcement provides long-lasting, high strength tensile reinforcement and impact resistance for the panels or boards. The reinforcement also enables the boards to have smooth outer faces suitable for painting, papering, tiling or other finishing treatment. Included as part of the invention are methods for making the reinforcement, cementitious boards and panels including the reinforcement, and methods for manufacturing such boards and panels.
7861476	9/19/2005	1/4/2011	CertainTeed	N	N	Y	Y	Cementitious exterior sheathing product with rigid support member	Cementitious exterior sheathing products are provided which include a rigid support member affixed to a cementitious layer. The rigid support member includes at least one nailing flange disposed along one of its lateral sides for allowing the sheathing product to be affixed to an exterior wall of a building. Preferred mechanical and adhesive bonding techniques are suggested for combining the cementitious layer and rigid support member together to form an integrated product. Such products are lighter in weight and are more crack resistant than currently available fiber cement trim boards.
7892472	8/12/2004	2/22/2011	USG Corporation	N	N	Y	Y	Method of making water-resistant gypsum-based article	A moisture resistant gypsum-based product, e.g., a gypsum board, is made by adding a small amount of a siloxane to the aqueous slurry used to make the gypsum-based product along with a small amount of a dead burned magnesium oxide catalyst to enhance the curing of the siloxane. In the preferred embodiment, the siloxane is formed into an aqueous emulsion in situ with no chemical emulsifier.
7932195	5/17/2010	4/26/2011	Georgia Pacific	N	N	Y	Y	Use of pre-coated mat for preparing gypsum board	A gypsum board which comprises a set gypsum core sandwiched between and faced with fibrous mats, wherein a free surface of one of said mats is pre-coated with a combination of a mineral pigment, optionally an inorganic adhesive binder and an organic binder, preferably a hydrophobic, UV resistant polymer latex adhesive binder applied to said surface as an aqueous coating composition, said aqueous coating composition upon drying and setting providing a pre-coated mat satisfying certain morphology requirements.
7989370	10/5/2004	8/2/2011	Georgia Pacific	N	N	Y	Y	Interior wallboard and method of making same	A gypsum wallboard suitable for Level 4 finishing having a coated non-woven glass fiber mat facing material where the glass fiber mat has a majority of fibers of a fiber diameter between 8 and 11 microns and a fiber length between and inch and preferably between and inch and preferably has a basis weight between about 0.8 lb./100 ft.and about 2.2 lb./100, and wherein the fibers in the non-woven glass fiber are bound together with an acrylic adhesive binder and wherein the non-woven glass mat has a coating of a dried aqueous mixture of (i) a mineral pigment, (ii) a polymer latex adhesive binder and optionally (iii) an inorganic adhesive binder such that the coated non-woven glass mat facing material has a porosity which allows water to evaporate through said coated mat from the gypsum core during preparation of the wallboard.
8070895	4/20/2007	12/6/2011	USG Corporation	N	Y	Y	Y	Water resistant cementitious article and method for preparing same	A fibrous mat-faced cementitious article comprising (a) a cementitious core, and (b) a first fibrous mat comprising polymer or mineral fibers and a hydrophobic finish on at least one surface thereof, wherein the hydrophobic finish is in contact with the cementitious core, and a method of preparing a fibrous mat-faced cementitious article, as well as a method of preparing a water-resistant cementitious article comprising (a) preparing an aqueous siloxane dispersion, wherein the dispersion comprises about 4 wt. % to about 8 wt. % siloxane, (b) combining the siloxane dispersion with a cementitious mixture to provide a cementitious slurry, (c) depositing the cementitious slurry onto a substrate, and (d) allowing the cementitious slurry to harden, thereby providing a cementitious article.
8133600	9/22/2010	3/13/2012	USG Corporation	N	N	Y	Y	Siloxane polymerization in wallboard	Polymerization of siloxane is improved using a gypsum-based slurry that includes stucco, Class C fly ash, magnesium oxide and an emulsion of siloxane and water. This slurry is used in a method of making water-resistant gypsum articles that includes making an emulsion of siloxane and water, then combining the slurry with a dry mixture of stucco, magnesium oxide and Class C fly ash. The slurry is then shaped as desired and the stucco is allowed to set and the siloxane polymerizes. The resulting product is useful for making a water-resistant gypsum panel having a core that includes interwoven matrices of calcium sulfate dihydrate crystals and a silicone resin, where the interwoven matrices have dispersed throughout them a catalyst comprising magnesium oxide and components from a Class C fly ash.

Figure 30: **Continued:** All forward related patents to U.S. 3,935,021

patent_id	app_date	grant_date	organization	css	google	human	cited	title	abstract
8142914	2/15/2011	3/27/2012	USG Corporation	Y	N	N	N	Gypsum-containing product and gypsum board	The invention provides a set gypsum-containing product having increased resistance to permanent deformation and a method for preparing it comprising forming a mixture of a calcium sulfate material, water, and an appropriate amount of one or more enhancing materials chosen from condensed phosphoric acids, each of which comprises 2 or more phosphoric acid units; and salts or ions of condensed phosphates, each of which comprises 2 or more phosphate units. The mixture is then maintained under conditions sufficient for the calcium sulfate material to form a set gypsum material.
8192658	11/29/2006	6/5/2012	CertainTeed	N	N	Y	Y	Cementitious exterior sheathing product having improved interlaminar bond strength	The present invention provides exterior building products, such as roofing and siding, shake, shingles, siding, sheathing, panels, planks, vertical siding, soffit panels, fencing, decking, fascia, corner posts, column corners and trim boards in which a plurality of cementitious layers are provided with an improved interlaminar bond by employing a resinous bond promoter, a rheological agent, mechanical means to distribute fibers in a direction which is perpendicular to the machine direction so as to bridge between layers in the product, or a combination thereof. These techniques help to increase interlaminar bond strength to improve the mechanical properties of the product. When certain resinous bond promoters are used, the additional benefits of water absorption resistance and pigmentation throughout the product can be provided with minimal expense. Improvements in interlaminar bond strength of about 10 46% were observed with a percent elongation improvement of about 7%.
8323785	2/17/2012	12/4/2012	USG Corporation	Y	N	N	N	Lightweight, reduced density fire rated gypsum panels	A reduced weight, reduced density gypsum panel that includes high expansion vermiculite with fire resistance capabilities that are at least comparable to (if not better than) commercial fire rated gypsum panels with a much greater gypsum content, weight and density.
8329308	3/31/2009	12/11/2012	USG Corporation	N	N	Y	Y	Cementitious article and method for preparing the same	A cementitious article and a method of making a cementitious article are disclosed. The cementitious article comprises a cementitious component that comprises a polyvinyl acetate type polymer, a monobasic phosphate, and optionally boric acid. Cementitious articles, such as board, are prepared such that the polyvinyl acetate type polymer, the monobasic phosphate, and optionally boric acid can be present in the cementitious core, and/or in dense layers if present. The concentration of the polyvinyl acetate type polymer, monobasic phosphate, and optionally boric acid in the cementitious article can increase from a central region A to peripheral regions B and C, respectively. In some embodiments, the polyvinyl acetate type polymer is a polyvinyl alcohol and the monobasic phosphate is monoammonium phosphate.
8461067	4/22/2011	6/11/2013	Georgia Pacific	N	N	Y	Y	Use of pre-coated mat for preparing gypsum board	A gypsum board which comprises a set gypsum core sandwiched between and faced with fibrous mats, wherein a free surface of one of said mats is pre-coated with a combination of a mineral pigment, optionally an inorganic adhesive binder and an organic binder, preferably a hydrophobic, UV resistant polymer latex adhesive binder applied to said surface as an aqueous coating composition, said aqueous coating composition upon drying and setting providing a pre-coated mat satisfying certain morphology requirements.
8470461	6/11/2012	6/25/2013	USG Corporation	Y	N	N	N	Light weight gypsum board	The invention generally provides gypsum-containing slurries including stucco, naphthalenesulfonate dispersant, and pregelatinized starch. The naphthalenesulfonate dispersant is present in an amount of about 0.1%-3.0% by weight based on the weight of dry stucco. The pregelatinized starch is present in an amount of at least about 0.5% by weight up to about 10% by weight of pregelatinized starch by weight based on the weight of dry stucco in the formulation. Other slurry additives can include trimetaphosphate salts, accelerators, binders, paper fiber, glass fiber, and other known ingredients. The invention also comprises the gypsum-containing products made with such slurries, for example, gypsum wallboard, and a method of making gypsum wallboard.
8501074	4/21/2011	8/6/2013	USG Corporation	Y	N	Y	Y	Siloxane polymerization in wallboard	Polymerization of siloxane is improved using a gypsum-based slurry that includes stucco, Class C fly ash, magnesium oxide and an emulsion of siloxane and water. This slurry is used in a method of making water-resistant gypsum articles that includes making an emulsion of siloxane and water, then combining the slurry with a dry mixture of stucco, magnesium oxide and Class C fly ash. The slurry is then shaped as desired and the stucco is allowed to set and the siloxane polymerizes. The resulting product is useful for making a water-resistant gypsum panel having a core that includes interwoven matrices of calcium sulfate dihydrate crystals and a silicone resin, where the interwoven matrices have dispersed throughout them a catalyst comprising magnesium oxide and components from a Class C fly ash.
8563139	4/22/2008	10/22/2013	USG Corporation	N	N	Y	Y	Non-hydrating plaster composition and method	A method of finishing an interior wall includes the steps of preparing a substrate of building panels comprising gypsum, cement or combinations thereof, said substrate having a surface, followed by applying a coating to the substrate, said coating comprising 5-20% by weight of a latex emulsion binder, 40-80% by weight calcium sulfate hemihydrate, 0.05-2% by weight of a set preventer and 20-60% by weight water.
8568544	10/28/2011	10/29/2013	USG Corporation	N	N	Y	Y	Water resistant cementitious article and method for preparing same	A fibrous mat-faced cementitious article comprising (a) a cementitious core, and (b) a first fibrous mat comprising polymer or mineral fibers and a hydrophobic finish on at least one surface thereof, wherein the hydrophobic finish is in contact with the cementitious core, and a method of preparing a fibrous mat-faced cementitious article, as well as a method of preparing a water-resistant cementitious article comprising (a) preparing an aqueous siloxane dispersion, wherein the dispersion comprises about 4 wt. % to about 8 wt. % siloxane, (b) combining the siloxane dispersion with a cementitious mixture to provide a cementitious slurry, (c) depositing the cementitious slurry onto a substrate, and (d) allowing the cementitious slurry to harden, thereby providing a cementitious article.
8945462	4/16/2007	2/3/2015	Yoshino Gypsum	Y	N	N	N	Methods for manufacturing a calcined gypsum and a gypsum board	The present invention provides a method for manufacturing a calcined gypsum wherein the mixing water amount is reduced and the setting time does not increase. As a raw gypsum is compounded with a carboxylic acid-type material and calcined, a calcined gypsum can be manufactured wherein the mixing water amount is small and the setting time does not increase. Furthermore, a regular gypsum board can be manufactured without reducing the productivity of the gypsum board even if a large quantity of recycled gypsum causing increase of the mixing water amount is used as a raw gypsum, because the mixing water amount is small and the setting time does not increase for the calcined gypsum manufactured as described above.
9017495	11/10/2010	4/28/2015	Saint Gobain	N	N	Y	N	Methods of making smooth reinforced cementitious boards	Methods and a reinforcement fabric are disclosed for making a reinforced smooth cementitious board having a cement skin adjacent to an outer face, by depositing a reinforcement fabric and a layer of hydraulic cementitious material, one on the other, wherein the reinforcement fabric comprises an open mesh united with a thin, porous nonwoven web.
9221719	2/23/2012	12/29/2015	National Gypsum	Y	N	N	N	Gypsum wallboard slurry and method for making the same	A slurry for manufacturing gypsum board is disclosed. The slurry comprises calcined gypsum, water, a foaming agent, and a thickening agent. The thickening agent of the present disclosure acts to improve the cohesiveness of the slurry without adversely affecting the setting time of the slurry, the paper-to-core bond (wet and dry), or the head of the slurry by acting as a defoaming agent or coalescing agent. Examples of suitable thickening agents include cellulose ether and co-polymers containing varying degrees of polyacrylamide and acrylic acid. A gypsum board and method of forming the slurry and the gypsum board are also disclosed. The gypsum board comprises a gypsum layer formed from the slurry.
9321685	5/21/2013	4/26/2016	Yoshino Gypsum	Y	N	N	N	Gypsum composition, gypsum slurry, gypsum hardened body, gypsum-based building material, gypsum board, and manufacturing method for a gypsum-based building material	A gypsum composition includes a calcined gypsum and a starch urea phosphate.

Figure 30: **Continued:** All forward related patents to U.S. 3,935,021

CHAPTER 4

DISCUSSION

The purpose of this research is to understand if ML techniques and natural language processing (NLP) can be applied to patent data to help answer questions about inventiveness and how inventive knowledge disseminates over time. There is good reason to believe that historical methods using numerical metrics such as patent counts and citation analysis cannot completely capture inventive knowledge flow. This research has shown that ML computational techniques can be used to convert unstructured patent textual data into actionable knowledge, and in doing so, laid the ground work to study inventiveness and knowledge flow based upon how inventors describe their inventions over time. The potential exists to use both methods to reveal interesting and perhaps anomalous patenting behavior through direct comparison. For example an an interesting anomalous pattern was revealed in Section 3.3.8 when a distinct divergence of forward citation counts occurred over a four-year time frame for a majority of the top cited patents that was not seen with CSS-related counts. This approach is not unlike the use of ML to scan and detect potential fraudulent financial transactions within the credit card industry and bring those alerts to the attention of security experts so that they can further investigate only high probability events [139–141]. More research is needed to determine if CSS-relatedness or other ML approaches to determine inventive similarity among patents are better measures than citation counts alone. There is significant opportunity for continued use of these techniques to explore the USPTO patent data set and perhaps the World Intellectual Property Organization

(WIPO) data set. The approach presented here also holds forth the possibility of tools that could be applied to locally held corporate proprietary databases of inventive records to identify which inventions are suited for patent protection. There are also opportunities to further refine and improve upon the methods used in this research. This includes further exploration of the n-gram parameter and its effect on identification of inventive descriptors, optimization of the maximum term frequency threshold parameter during the vector transformation step, and feeding the output of the tf-idf step into latest language modeling and feature learning techniques such as word and phrase embedding [142] and tensors [143]. Opportunities to further explore and build upon this research are many and beyond the scope of what can be accomplished in a single thesis. Nonetheless, the sections that follow reflect on some potential areas of exploration.

4.1 Inventive Descriptors

There is an opportunity for an improved process to enhance how domain experts determine the core inventive aspects of any patented invention. This is not a matter of finding broad technical topics or using common word searches. In today's hyper-competitive environment it is important for domain experts to be able to distill these lengthy intellectual property documents down to core combinations of words that capture the essence of the invention. This is becoming increasingly difficult as the number of intellectual property documents grow worldwide. Currently, without using any ML tools, a domain expert who is concerned about technological disruption or a competitor infringing upon their intellectual property would conduct a patent search by first filtering for competitor names and then sorting in reverse chronological order. This would be followed by a quick review of the patent title and abstract for some number of patents that meet the search criteria, with a deeper dive into the patent claims for inventions of interest. The domain expert may never read the unstructured textual data that is key to deciphering the core aspects of the invention if the patent does not pass their initial screening. The approach presented here would allow the domain expert to replace this screening with on a ML approach that would search the entire patent corpus for similar inventive content supplied by the domain expert, whether or not the authors of the original patents used helpful keywords or chose to cite relevant patents. In this workflow, the domain expert would still need to read in detail some number of patents identified as of potential interest, but the list of patents for review would be generated by a ML step. Ideally, this would reduce or eliminate some time-consuming aspects of manual patent searches so that valuable expert time can be used more efficiently. This would require improvements to our current method especially the importation, pre-processing, and transformation steps. For example, additional textual data from the patent should be imported beyond the title, abstract, and claims which would be an obvious next step for future research. This includes the body of the patent which contains the technical field, background, summary of the invention, brief description of drawings, and detailed description. In addition, the pre-processing step should include a chemical dictionary to expand element names such as "Fe" for iron, convert chemical nomenclature to common names such a "CaSO4" to calcium sulfate, and be able to differentiate between important preferred usage amounts and general numerical measures. There is specific language that typically precedes a best mode description which can be used to differentiate an important range of values. An example of such language would be "the preferred process includes at least" which is distinctly different from stating "the invention improves efficiency by 30%". The former is an indication of key aspect of the invention more akin to a recipe while the latter says nothing about core invention itself.

4.2 Patent Jargon and Enabling Language - "Start Words"

A key part of our pre-processing function was a step to remove common low-value intellectual property language through comparison of text to a dictionary called *patent jargon.* The selection of words was based upon both domain expertise and term frequency analysis. Close scrutiny of the patent jargon dictionary revealed that some terms were consistently proximate to inventive terms and that the potential existed to use these words as proximate indicators or *start words* of inventive descriptors close by. Some examples of start words would be "according", "claim", and "consisting" in the phrase "The absorbent sheet according to claim 1, consisting of predominately hardwood fiber". In this example we would want the inventive descriptors "absorbent sheet" and "hardwood fiber" to be weighted higher because they fall within a certain proximity of a start word. The hypothesis is that specific patent phrases, while having no intrinsic domain content, can be used to identify other terms that are intended to satisfy the patent office general and description requirements and thus contain significant domain content.

4.3 Weighted Patent Claims

Claims are intended to define the exclusivity zone of an invention by delineating, using an ordered list of detailed text statements, what inventors are seeking to prevent others from making or selling. Patent claims are considered the heart of an invention which is why they are one of the first areas studied by domain experts when trying to understand key inventive elements. [144] discussed how the evolution of the written requirement has resulted in placing emphasis on clear invention definition in the claims section while the patent detail is used to discuss prior art, invention rationale, and inventive contemplation that can be broader than the claims. It is not uncommon to have generic title descriptions and obfuscating language used in the abstract that makes it difficult to understand the nature of the invention, however, since the claims represent what the inventor seeks to have exclusivity over there is a strong incentive to avoid misunderstandings. It is therefore quite clear that the textual data in the patent claims have greater importance than the title and abstract. This research treated textual data equally, but it is possible to weight claims higher than text from other sections. Tong and Frame [145] recognized the importance of claims when they posed that the number of patent claims was a better indicator of "inventiveness" than patent counts when measuring relative global technological performance of countries. In their research each claim was treated as an individual unit of inventiveness, and in turn, the total number of claims was used to measure the overall inventiveness of a patent. There are some important considerations that need to be made when it comes to treating claims as individual units of inventiveness. First, not all claims are created equal in terms of importance and scope. For example, claim order is relevant with the most important claim of the broadest scope being listed first. Second, there are two different types of claims, *independent* and *dependent*. Independent claims represent core inventive pieces that are necessary to practice the invention while dependent claims help expand and clarify their respective independent claim. An independent patent claim typically stands by itself as a discrete inventive step while dependent claims include all of the features of their respective independent claims with some degree of additional specificity. Thus dependent claims should not be treated as inventive steps, instead it would be more appropriate to use them to numerically weight their corresponding independent claims. These considerations make numerical counts of claims less appealing as a metric for inventiveness, however, the importance of claims in delineating the boundaries of an invention make them ripe for numerical weighting by using the counts of dependent claims as a multiplier on the corresponding independent claim. It is relatively trivial to determine the type of claim by assessing whether it refers to another claim, this is typically marked by a distinct combination of words, for example, a claim beginning with the words, "The method of Claim 1". A weighted approach to the number of independent claims would be appropriate when assessing term frequency of inventive descriptors contained within a patent because each dependent claim is considered to have all the word features of its original claim even if not stated. Therefore, from a textual analysis perspective the inventive descriptors contained within an independent claim would be weighted by the number of dependent claims because those terms are assumed to be repeated.

4.4 Predictive models

This research explored the potential to use ML to predict the CPC technical category of patents using both Decision Trees and ANN in Chapter 2. This raises questions as to whether ML predicative capabilities can be used to model inventiveness and inventive knowledge flow. For example, a number of citation-based calculated metrics were introduced in Section 2.2, including *number of citations made & received*, *backward & forward citation lag*, *percent citation total and self-citation*, and *measure of generality and originality.* The number of forward citations received metric is a direct proxy for "inventiveness" while the *mean forward citation lag*, the average time between the application date of the originating patent and those that cite it, is directly related to the diffusion of inventive knowledge over time. It would be interesting to explore whether these metrics can be modeled as dependent variables using a ML classification technique such as k-nearest neighbors (k-NN) [76, 85, 108, 146]. If so, this can be used to create a useful predictive model for future behavior. It will be necessary to identify the appropriate independent variables that can be evaluated using ML from a host of numerical data that are already available in the patent record. Both independent and dependent variables can make use of cosine similarity measures of the patent in question against the extant corpus and thus provide potentially new predictive models for inventiveness and knowledge flow. For example, substitution of CSS-relatedness counts for forward citations could be used to assess both similarity and dissimilarity of inventiveness prediction outcomes between the two methods. Note that cosine similarity could augment or even entirely replace actual citation counts as a measure of mean citation lag. The dependent variable that measures knowledge flow or commercial success can likewise incorporate cosine similarity as well as citation lag, but will only be measurable for patents if there is a sufficient patent corpus following them in time. It may also be interesting to investigate inter-industry versus intra-industry knowledge flow by looking at whether high cosine similarity documents occur within or outside a given CPC technology area. The research outlined in this thesis suggests the possible adaptation of generality and originality to use cosine similarity between inventive descriptors rather than citations. Furthermore, metrics based upon CSS-relatedness counts could provide a timelier and more complete measure of technological impact that could be applied to corporate invention record filings and trade secrets prior to the decision to publicly disclose an invention. The ultimate goal of this research would be to predict the impact and dissemination time of a given invention by exploiting the wealth of technical data within the patent record. This knowledge would invaluable when making intellectual property decisions related to competitive market behavior. For example, a spike in CSS-relatedness counts for a patent within a given CPC technological category could be indicative of competitive fast follower behavior and that this sudden increase would occur when competitors have developed an alternative means to achieve the same inventive purpose as the original patent even though there may not be direct citations. The time it takes to reach this “sudden increase” may be indicative of the strength of the patent related to the difficulty of others to circumvent its original protective claims without infringement; that increases in citations upon and after a patent’s expiration is more representative of knowledge flow; that what defines a “sudden increase” is unique to a given industry or technology area; and that it is based upon a user-defined suitable threshold related to the expected rate of technology change common for that industry.

CHAPTER 5

CONCLUSION

This research has presented the use of computational techniques from ML and NLP to convert unstructured patent textual data into structured actionable knowledge that can be used to study *inventiveness* and *knowledge flow.* This approach differs from historical techniques used to assess the "quality"and "impact" of patents, such as counting of citations [43], total cost and willingness over time to pay patent maintenance fees [147], or counting the number of countries in which patents are filed [148] or the number of claims listed [145]. The approach was twofold. First, develop an improved computational process to enhance how domain experts determine the core inventive aspects of any patented invention. Ideally, this process could be used to reduce the time-consuming aspects of manual patent searches by culling out core inventive knowledge quickly, so that valuable expert time can be used more efficiently. Second, develop new techniques to measure inventiveness and knowledge flow through the application of supervised ML using newly constructed text-based cosine similarity values as dependent variables for patent relatedness selection. To demonstrate the application of these techniques several visualizations were created to compare and contrast intra-CSS-relatedness measures with a more traditional method based on citation analysis. These two ways of measuring inventiveness and inventive knowledge flow show both similarities and differences. There is good reason to believe that citation counts cannot completely capture knowledge flow. More research is needed to determine if CSS-relatedness or other ML approaches to determine inventive similarity among patents are better measures than citation counts alone. The potential exists to use both methods to reveal interesting and perhaps anomalous patenting behavior by comparing core inventive language to numerical data. The conclusion here is that ML techniques are ready to be used in such research. Jaffe and de Rassenfosse [1] provided an elegant geometric interpretation of the relationship between inventive language in the form of patent claims, technological innovation, and the purpose of citations which was prescient to this research and repeated below.

> *"First, we can think of all possible technologies as mapping onto a high-dimensional technology space, such that a given invention can be located in that space, and a patent represents the right to exclude others from marketing products that impinge upon specific region (or regions) of that space. Second, the invention process is cumulative, that is, inventions build on those that came before and, in turn, facilitate those that come after. In this geometric interpretation, the patent claims delineate the metes and bounds of the region of technology space over which exclusivity is being granted, while the citations indicate previously marked off areas that are in some sense built upon by or connected to the invention being granted."* (7)

This research has shown that ML techniques can be used to create this geometric interpretation by converting patent title, abstract, and claim textual data into numerical vectors that can be placed into a higher-dimensional technology space. Furthermore, the metes and bounds between these vectors can be delineated by measuring the cosine angle between them and thus establishing relatedness. Lastly, the patent application filing dates can be used to define previously marked off areas. This was achieved using the following steps.

First, unstructured abstract, title, and claim textual data from a subset of patent documents from a competitive group of companies were extracted. This included the application of NLP techniques to convert the unstructured textural data into term frequency based numerical vectors. A key part of this first step was a pre-processing function that removed common legal language called *patent jargon* as well as steps to retain n-gram inventive word pairs. A statistical method called term frequency-inverse document frequency (tf-idf) was then used to convert the sets of numerical vectors into real-valued numerical vectors within a vector space model. The transformation was modified to ensure that all rare inventive terms were captured no matter how infrequent, while at the same time removing common low value information words associated with intellectual property language. Selecting for core inventive terms diverges from standard NLP methodology in that we choose to seek out rare combinations of words. Second, the cosine angle between all patent vectors within the higher-dimension vector space was measured. A parameter α was selected to screen for patent relatedness based upon a minimum cosine threshold value. This value was selected based upon the weighted mean cosine similarity score of known cited patent pairs. In doing so, this research utilized the existing patent record to establish a minimum threshold value using forward cited patent pairs instead of a domain expert's subjective assessment of relatedness.

Third, to visualize patent cosine relatedness the higher-dimension vector space was reduced to a two-dimensional plot using principal component analysis and then t-distributed stochastic neighbor embedding. The resulting visualization was then color coded using patent office technical classifications to reveal some discrete technology class clusters. The match between the CSS relatedness clusters and CPC classes, while not formal validation, indicates that this approach is aligned with traditional classication methods.

Lastly, a comparison of invention impact and inventive knowledge flow is demonstrated by plotting patent citation data alongside cosine relatedness outputs to reveal both similar and dissimilar inventive patterns. Knowing that CSS-relatedness is based upon the similarity of core inventive terms and that formal backward citing is based upon inventor compliance with USPTO prior art requirements raises some interesting opportunities to detect anomalous citation behavior. Advances in computational power and ML tools have opened the door to investigate a number of interesting questions in intellectual property decision modeling. This research has shown that new computational techniques can be used to convert unstructured patent textual data into actionable knowledge, and in doing so, laid the groundwork to study inventiveness and knowledge flow with less dependence on numerical metrics such as patent counts and citation analysis. The increased understanding can be used to study inventive knowledge flow which has both scientific and practical applications. The conversion of unstructured patent textual data into structured data that can be used for intellectual property decision modeling, and the methodological enhancements provided to researchers in the field, provide increased understanding of the nature of *inventiveness* & *inventive knowledge flow*. There is also significant potential to reduce the most time-consuming and tedious aspects of patent searches, not only to free up valuable domain expert time, but also provide them with more actionable knowledge to make informed intellectual property decisions. Going forward these techniques could be used in a number of applications including novelty assessment of patent applications prior to filing, competitive surveillance to monitor published patent applications, an automated patentability assessment tool, or as a means to suggest new inventive steps by combining inventive descriptors between CSS related patents.

CITED LITERATURE

1. Jaffe, A. B. and de Rassenfosse, G.: Patent Citation Data in Social Science Research: Overview and Best Practices. Working Paper 21868, National Bureau of Economic Research, January 2016.

2. World Intellectual Property Organization: Directory of Intellectual Property Offices. http://www.wipo.int/directory/en/urls.jsp, April 2018.

3. World Intellectual Property Organization: Member States. http://www.wipo.int/members/en/index.jsp, February 2018.

4. International Bureau of WIPO: What is Intellectual Property? *WIPO Pub. No. 450* , 2003.

5. Tseng, Y.-H., Lin, C.-J., and Lin, Y.-I.: Text mining techniques for patent analysis. *Information Processing & Management* , 43(5):1216–1247, September 2007.

6. United States Patent and Trademark Office: Utility Patent Counts By Country, State, and Year (December 2015) — USPTO. https://www.uspto.gov/web/offices/ac/ido/oeip/taf/cst_utl.htm, March 2018.

7. Schmookler, J.: *Invention and Economic Growth* . Cambridge, Mass., Harvard University Press, first edition edition, January 1966.

8. Schiffel, D. and Kitti, C.: Rates of invention: International patent comparisons. *Research Policy* , 7(4):324–340, October 1978.

9. Pavitt, K. and Soete, L.: International differences in economic growth and the international location of innovation. *Emerging Technologies: The Consequences for Economic Growth, Structural Change and Employment* , pages 105–133, 1982.

10. Chakrabarti, A. K.: Competition in high technology: Analysis of patents of US, Japan, UK, France, West Germany, and Canada. *IEEE Transactions on Engineering Management* , 38(1):78–84, February 1991.

11. Griliches, Z.: Patent statistics as economic indicators: A survey. In *R&D and Productivity: The Econometric Evidence* , pages 287–343. University of Chicago Press, 1998.

12. Chakrabarti, A. K., Dror, I., and Eakabuse, N.: Interorganizational transfer of knowledge: An analysis of patent citations of a defense firm. *IEEE Transactions on Engineering Management* , 40(1):91–94, 1993.

13. Crépon, B., Duguet, E., et al.: Estimating the innovation function from patent numbers: GMM on count panel data. *Journal of Applied Econometrics* , 12(3):243–263, 1997.

14. Grandjean, N., Charpiot, B., Pena, C. A., and Peitsch, M. C.: Competitive intelligence and patent analysis in drug discovery: Mining the competitive knowledge bases and patents. *Drug Discovery Today: Technologies* , 2(3):211–215, 2005.

15. Xu, K., Liao, S. S., Li, J., and Song, Y.: Mining comparative opinions from customer reviews for Competitive Intelligence. *Decision support systems* , 50(4):743–754, 2011.

16. Canongia, C.: Synergy between Competitive Intelligence (CI), Knowledge Management (KM) and Technological Foresight (TF) as a strategic model of prospecting — The use of biotechnology in the development of drugs against breast cancer. *Biotechnology Advances* , 25(1):57–74, January 2007.

17. Hall, B. H., Jaffe, A. B., and Trajtenberg, M.: The NBER patent citation data file: Lessons, insights and methodological tools. Technical report, National Bureau of Economic Research, 2001.

18. Crépon, B., Duguet, E., and Mairessec, J.: Research, Innovation And Productivity: An Econometric Analysis At The Firm Level. *Economics of Innovation and new Technology* , 7(2):115–158, 1998.

19. Albert, M. B., Avery, D., Narin, F., and McAllister, P.: Direct validation of citation counts as indicators of industrially important patents. *Research Policy* , 20(3):251–259, June 1991.

20. Breitzman, A. F. and Mogee, M. E.: The many applications of patent analysis. *Journal of Information Science* , 28(3):187–205, 2002.

21. Lemley, M. A. and Sampat, B.: Examiner characteristics and patent office outcomes. *Review of Economics and Statistics* , 94(3):817–827, 2012.

22. Keller, R. T.: Job involvement and organizational commitment as longitudinal predictors of job performance: A study of scientists and engineers. *Journal of Applied Psychology* , 82(4):539, 1997.

23. Wang, X., Zhang, X., and Xu, S.: Patent co-citation networks of Fortune 500 companies. *Scientometrics* , 88(3):761–770, 2011.

24. Hibben, Mark: The Difference Between Apple And Google. http://seekingalpha.com/article/4051172-difference-apple-google, January 2017.

25. Audretsch, B.: Agglomeration and the location of innovative activity. *Oxford review of economic policy* , 14(2):18–29, 1998.

26. Jaffe, A. B., Trajtenberg, M., and Henderson, R.: Geographic localization of knowledge spillovers as evidenced by patent citations. *the Quarterly journal of Economics* , 108(3):577–598, 1993.

27. Li, G.-C., Lai, R., D'Amour, A., Doolin, D. M., Sun, Y., Torvik, V. I., Amy, Z. Y., and Fleming, L.: Disambiguation and co-authorship networks of the US patent inventor database (1975–2010). *Research Policy* , 43(6):941–955, 2014.

28. Castaldi, C. and Los, B.: Geographical patterns in US inventive activity 1977–1998: The "regional inversion" was underestimated. *Research Policy* , 46(7):1187–1197, September 2017.

29. Statista: Total number of plant patents issued in the U.S. FY 1997-FY 2018 — Statistic. https://www.statista.com/statistics/256586/number-of-plant-patent-grants-in-the-us/, 2018.

30. Statista: Patents in the United States. https://www.statista.com/study/14876/patents-in-the-united-states-statista-dossier/, 2016.

31. Trajtenberg, M., Henderson, R., and Jaffe, A.: University versus corporate patents: A window on the basicness of invention. *Economics of Innovation and new technology* , 5(1):19–50, 1997.

32. Foglia, P.: Patentability search strategies and the reformed IPC: A patent office perspective. *World Patent Information* , 29(1):33–53, March 2007.

33. Vijvers, W. G.: The international patent classification as a search tool. *World Patent Information* , 12(1):26–30, January 1990.

34. Kisliuk, B.: *Introduction to the Cooperative Patent Classification (CPC)* . United States Patent and Trademark Office, 2010.

35. Montecchi, T., Russo, D., and Liu, Y.: Searching in Cooperative Patent Classification: Comparison between keyword and concept-based search. *Advanced Engineering Informatics* , 27(3):335–345, August 2013.

36. United States Patent and Trademark Office: Manual of Patent Examining Procedure (MPEP) Ninth Edition, Revision 08.2017, Last Revised January 2018. https://www.uspto.gov/web/offices/pac/mpep/index.html, 2018.

37. Heckadon, D.: New Ways to Challenge Patents Both Before and After They Issue. https://www.gordonrees.com/newsroom/2012/new-ways-to-challenge-patents-both-before-and-after-they-issue, November 2012.

38. Acs, Z. J. and Audretsch, D. B.: Patents as a measure of innovative activity. *Kyklos* , 42(2):171–180, 1989.

39. Acs, Z. J., Anselin, L., and Varga, A.: Patents and innovation counts as measures of regional production of new knowledge. *Research Policy* , 31(7):1069–1085, September 2002.

40. Jaffe, A. B. and Trajtenberg, M.: International knowledge flows: Evidence from patent citations. *Economics of Innovation and New Technology* , 8(1-2):105–136, 1999.

41. Agrawal, A., Cockburn, I., and McHale, J.: Gone but not forgotten: Knowledge flows, labor mobility, and enduring social relationships. *Journal of Economic Geography* , 6(5):571–591, 2006.

42. Archibugi, D. and Planta, M.: Measuring technological change through patents and innovation surveys. *Technovation* , 16(9):451–519, September 1996.

43. Trajtenberg, M.: A penny for your quotes: Patent citations and the value of innovations. *The Rand Journal of Economics* , pages 172–187, 1990.

44. Caballero, R. J. and Jaffe, A. B.: How high are the giants' shoulders: An empirical assessment of knowledge spillovers and creative destruction in a model of economic growth. *NBER macroeconomics annual* , 8:15–74, 1993.

45. He, Z.-L. and Deng, M.: The evidence of systematic noise in non-patent references: A study of New Zealand companies' patents. *Scientometrics* , 72(1):149–166, 2007.

46. Rosenzweig, S. and Mazursky, D.: Constraints of internally and externally derived knowledge and the innovativeness of technological output: The case of the United States. *Journal of Product Innovation Management* , 31(2):231–246, 2014.

47. Cooper, M. J., Knott, A. M., and Yang, W.: Measuring Innovation. *Available at SSRN 2631655* , 2015.

48. Mello, J. P.: Technology Licensing and Patent Trolls. *BUJ Sci. & Tech. L.* , 12:388, 2006.

49. Frost, G. E.: The 1967 patent law debate: First-to-invent vs. first-to-file. *Duke Law Journal* , pages 923–942, 1967.

50. DeBari, V. J.: International Harmonization of Patent Law: A Proposed Solution to the United States' First-to-File Debate. *Fordham Int'l LJ* , 16:687, 1992.

51. Kortum, S. and Lerner, J.: Stronger protection or technological revolution: What is behind the recent surge in patenting? *Carnegie-Rochester Conference Series on Public Policy* , 48:247–304, June 1998.

52. Kortum, S. and Lerner, J.: What is behind the recent surge in patenting? *Research policy* , 28(1):1–22, 1999.

53. Pedersen, B. and Braginsky, V.: The Rush to the First-to-File Patent System in the United States: Is a Globally Standardized Patent Reward System Really Beneficial to Patent Quality and Administrative Efficiency. *Minn. JL Sci. & Tech.* , 7:757, 2005.

54. Meridith, John and Grzelak, K.: Letter to House and Senate Leaders and Judiciary Committee Members Opposing Adoption of the Patent Reform Act of 2007 (S. 1145/H.R. 1908)., August 2007.

55. Lo, S.-t. and Sutthiphisal, D.: Does it matter who has the right to patent: First-to-invent or first-to-file? Lessons from Canada. Technical report, National Bureau of Economic Research, 2009.

56. World Trade Organization: Trade-Related Aspects of Intellectual Property Rights. https://www.wto.org/english/tratop_e/trips_e/trips_e.htm#WhatAre, April 2018.

57. Cambia: What is the difference between a filing date and a priority date? http://www.bios.net/daisy/patentlens/2343.html, 2018.

58. Standard and Poor's Global: S&P Global Market Intelligence. https://www.spglobal.com/marketintelligence/en/?product=compustat-research-insight, 2018.

59. Hirschman, A. O.: *National Power and the Structure of Foreign Trade* , volume 105. Univ of California Press, 1980.

60. Hirschman, A. O.: The paternity of an index. *The American Economic Review* , 54(5):761–762, 1964.

61. Herfindahl, O. C.: Concentration in the Steel Industry. PhD Thesis, Columbia University New York, 1950.

62. Omiecinski, E. R.: Alternative interest measures for mining associations in databases. *IEEE Transactions on Knowledge and Data Engineering* , 15(1):57–69, January 2003.

63. Kannan, S. and Bhaskaran, R.: Association rule pruning based on interestingness measures with clustering. *arXiv preprint arXiv:0912.1822* , 2009.

64. Ian, H.: Witten, Eibe Frank and Mark A. Hall Data Mining: Practical Machine Learning Tools and Techniques.-. *3rd Edition. Morgan Kaufmann* , page 664, 2011.

65. Quinn: The Best Mode Requirement: Not disclosing preferences in a patent application still a big mistake— Patents & Patent Law, February 2016.

66. United States Patent and Trademark Office: Patent Examiner Count System. https://www.uspto.gov/patent/initiatives/patent-examiner-count-system, March 2010.

67. United States Patent and Trademark Office: Recently Announced Changes to USPTO's Examiner Count System Go Into Effect. https://www.uspto.gov/about-us/news-updates/recently-announced-changes-usptos-examiner-count-system-go-effect, February 2010.

68. Mueller, J. M.: The Evolving Application of the Written Description Requirement to Biotechnological Inventions. *Berkeley Tech. LJ* , 13:615, 1998.

69. United States Patent and Trademark Office: New USPTO Tool Allows Exploration of 40 Years of Patent Data. https://www.uspto.gov/about-us/news-updates/new-uspto-tool-allows-exploration-40-years-patent-data, September 2015.

70. United States Patent and Trademark Office: PatentsView. https://www.uspto.gov/learning-and-resources/ip-policy/economic-research/patentsview, February 2017.

71. United States Patent and Trademark Office, P.: Data Download Tables — PatentsView. http://www.patentsview.org/download/, 2016.

72. Van Rossum, G. and Drake Jr, F. L.: *Python Tutorial* . Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

73. Immordino Jr, S. C. and Stevens, R. B.: Joint compound providing low dusting and good gloss retention, January 2004. U.S. 9,388,534 (C04B26/02), 10/093,771, 8 Mar 2002; 14 pp, 6 Jan 2004.

74. United States Patent and Trademark Office: Patent Classification. https://www.uspto.gov/patents-application-process/patent-search/classification-standards-and-development, 2019.

75. United States Patent and Trademark Office: Cooperative Patent Classfication. https://www.uspto.gov/web/patents/classification/cpc.html, April 2013.

76. Tseng, C.-Y.: Technological innovation and knowledge network in Asia: Evidence from comparison of information and communication technologies among six countries. *Technological Forecasting and Social Change* , 76(5):654–663, 2009.

77. Wang, W. M. and Cheung, C. F.: A Semantic-based Intellectual Property Management System (SIPMS) for supporting patent analysis. *Engineering Applications of Artificial Intelligence* , 24(8):1510–1520, December 2011.

78. Bird, S., Klein, E., and Loper, E.: *Natural Language Processing with Python* . " O'Reilly Media, Inc.", 2009.

79. scikit-learn: Sklearn.feature_extraction.text.CountVectorizer. http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html, 2017.

80. Luhn, H. P.: A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development* , 1(4):309–317, October 1957.

81. Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* , 28(1):11–21, 1972.

82. Manning, C. D., Raghavan, P., and Schütze, H.: Scoring, term weighting and the vector space model. *Introduction to information retrieval* , 100:2–4, 2008.

83. Gomaa, W. H. and Fahmy, A. A.: A survey of text similarity approaches. *International Journal of Computer Applications* , 68(13):13–18, 2013.

84. Stack Overflow: Systematic threshold for cosine similarity with TF-IDF weights. https://stackoverflow.com/questions/28882302/systematic-threshold-for-cosine-similarity-with-tf-idf-weights, 2016.

85. Trstenjak, B., Mikac, S., and Donko, D.: KNN with TF-IDF based Framework for Text Categorization. *Procedia Engineering* , 69:1356–1364, January 2014.

86. Perone, Christian: Machine Learning :: Text feature extraction (tf-idf) – Part I — Terra Incognita, September 2011.

87. Dumais, S. T.: Latent semantic analysis. *Annual review of information science and technology* , 38(1):188–230, 2004.

88. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R.: Indexing by latent semantic analysis. *Journal of the American society for information science* , 41(6):391, 1990.

89. Deerwester, S. C., Dumais, S. T., Furnas, G. W., Harshman, R. A., CA, Landauer, T. K., Lochbaum, K. E., and Streeter, L. A.: United States Patent: 4839853 - Computer information retrieval using latent semantic structure, June 1989. U.S. 4839853 (G06F17/21), 07/244,349 , 15 Sep 1988; 11 pp, 13 Jun 1989.

90. Blei, D. M., Ng, A. Y., and Jordan, M. I.: Latent dirichlet allocation. *Journal of machine Learning research* , 3(Jan):993–1022, 2003.

91. Blei, D. and Hoffman, M.: Online Learning for Latent Dirichlet Allocation. In *Neural Information Processing Systems* , 2010.

92. Sievert, C. and Shirley, K. E.: LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces* , pages 63–70, 2014.

93. Lu, Y., Mei, Q., and Zhai, C.: Investigating task performance of probabilistic topic models: An empirical study of PLSA and LDA. *Information Retrieval* , 14(2):178–203, 2011.

94. Anaya, L. A.: Comparing Latent Dirichlet Allocation and Latent Semantic Analysis as Classifiers. University of North Texas, 2011. Doctoral dissertation, University of North Texas, 2011.

95. Latent semantic analysis - Wikipedia. https://en.wikipedia.org/wiki/Latent_semantic_analysis.

96. Bayardo, R. J., Ma, Y., and Srikant, R.: Scaling Up All Pairs Similarity Search. In *Proceedings of the 16th International Conference on World Wide Web* , WWW '07, pages 131–140, New York, NY, USA, 2007. ACM.

97. Sidorov, G., Gelbukh, A., Gómez-Adorno, H., and Pinto, D.: Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas* , 18(3):491–504, 2014.

98. Huang, A.: Similarity measures for text document clustering. In *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand* , volume 4, pages 9–56, 2008.

99. Salton, G.: Developments in automatic text retrieval. *science* , 253(5023):974–980, 1991.

100. Sahami, M. and Heilman, T. D.: A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th International Conference on World Wide Web* , pages 377–386. AcM, 2006.

101. Raveena.S, N, and ini.V: Near Duplicate Document Detection Using Document-Level Features and Supervised Learning. *International Journal of Innovative Research in Computer and Communication Engineering* , 2(1), January 1970.

102. Alsulami, B. S., Abulkhair, M. F., and Eassa, F. E.: Near duplicate document detection survey. *International Journal of Computer Science and Communications Networks* , 2(2):147–151, 2012.

103. Ren, L. and Xu, Q.: Near Duplicate Document Detection: Mathematical Modeling and Algorithms. *ResearchGate* , 2012.

104. Han, J., Kamber, M., and Pei, J.: *Data Mining: Concepts and Techniques* . Elsevier, 2011.

105. Kaur, M. and Sapra, R.: Classification of Patents by Using the Text Mining Approach Based On PCA and Logistics. *International Journal of Engineering and Advanced Technology* , 2013.

106. Sorzano, C. O. S., Vargas, J., and Montano, A. P.: A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877* , 2014.

107. Alghamdi, R. and Alfalqi, K.: A Survey of Topic Modeling in Text Mining. *International Journal of Advanced Computer Science and Applications (IJACSA)* , 6(1), 2015.

108. Navigli, R.: Word Sense Disambiguation: A Survey. *ACM Comput. Surv.* , 41(2):10:1–10:69, February 2009.

109. Box, G. E. and Cox, D. R.: An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)* , pages 211–252, 1964.

110. van der Maaten, L. and Hinton, G.: Visualizing data using t-SNE. *Journal of machine learning research* , 9(Nov):2579–2605, 2008.

111. van der Maaten, L.: Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research* , 15(1):3221–3245, 2014.

112. van der Maaten, L.: T-SNE. https://lvdmaaten.github.io/tsne/, 2019.

113. Lloyd, S. P.: Least square quantization in PCM. Bell Telephone Laboratories Paper. Published in journal much later: Lloyd, SP: Least squares quantization in PCM. *IEEE Trans. Inform. Theor.(1957/1982)* , 18, 1957.

114. Sculley, D.: Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web* , pages 1177–1178. ACM, 2010.

115. Buchta, C., Kober, M., Feinerer, I., and Hornik, K.: Spherical k-Means Clustering. *Journal of Statistical Software* , 50:1–22, September 2012.

116. Rousseeuw, P. J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* , 20:53–65, November 1987.

117. Caliński, T. and Harabasz, J.: A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods* , 3(1):1–27, 1974.

118. Tibshirani, R., Walther, G., and Hastie, T.: Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* , 63(2):411–423, 2001.

119. Thorndike, R. L.: Who belongs in the family? *Psychometrika* , 18(4):267–276, 1953.

120. Lee, Y. K., Lee, E. R., and Park, B. U.: Principal Component Analysis In Very High-Dimensional Spaces. *Statistica Sinica* , 22(3):933–956, 2012.

121. United States Patent and Trademark Office: 1120 Eighteen-Month Publication of Patent Applications [R-07.2015]. https://www.uspto.gov/web/offices/pac/mpep/s1120.html, 2015.

122. Lehnert, C. W. and Randall, B. G.: Use of fibrous mat-faced gypsum board in exterior finishing systems for buildings, March 1987. U.S. 4,647,496 (B32B13/14), US06/583,874, 27 Feb 1984; 13 pp, 3 March 1987.

123. Watson, G. M.: Wet crepe, impingement-air dry process for making absorbent sheet, August 2002. U.S. 6,432,267 (D21F5/181), 60/171,070, 16 Dec 1999; 28 pp, 13 Aug 2002.

124. Edwards, S. L. and McCullough, S. J.: Wet-pressed tissue and towel products with elevated CD stretch and low tensile ratios made with a high solids fabric crepe process, September 2009. U.S. 7,588,660 (C04B26/02), 11/104,014, 12 Apr 2005; 31 pp, 15 Sep 2009.

125. Edwards, S. L., Super, G. H., McCullough, S. J., Baumgartner, D. J., Eggen, R. W., Duggan, D. P., Krueger, J. E., Lomax, D. W., and Jones, C. A.: Absorbent sheet made by fabric crepe process, September 2009. U.S. 7,558,661 (D21F11/145), 112/156,820 , 5 Jun 2008; 67 pp, 15 Sep 2009.

126. Swei, G. S. and Petigny, S.: Electrostatic deposition process, December 2002. U.S. 6,500,493 (C09K3/1409), 9/810,857 , 16 Mar 2001; 14 pp, 31 Dec 2002.

127. Moren, L. S., Koethe, B. G., and Thurber, E. L.: Layered particle electrostatic deposition process for making a coated abrasive article, October 2013. U.S. 8,551,577 (B24D11/005), 12/786,622 , 25 May 2010; 12 pp, 8 Oct 2013.

128. Moren, L. S., Koethe, B. G., and Thurber, E. L.: Layered particle electrostatic deposition process for making a coated abrasive article, October 2014. U.S. 8,869,740 (B24D11/005), 14/016,368 , 3 Sep 2013; 12 pp, 28 Oct 2014.

129. Jungbauer, K., Jr, J. R. B., and Boehmer, R. A.: Method of electrostatic deposition of particles, abrasive grain and articles, November 2014. U.S. 8,894,466 (B24D11/005), C09K3/1436 , 20 Jun 2012; 11 pp, 25 Nov 2014.

130. Seth, A.: Abrasive articles including a blend of abrasive grains and method of forming same, December 2015. U.S. 9,221,151 (B24D3/001), 14/145,900 , 31 Dec 2013; 15 pp, 29 Dec 2015.

131. Laconto, R. W. and Haerle, A. G.: Polishing slurries, January 2012. U.S. 8,105,135 (C09K3/1409), 11/541,431 , 29 Sep 2006; 11 pp, 31 Jan 2012.

132. Francois, E. C.: Dressable bonded abrasive article, February 2015. U.S. 8,994,893 (B24D3/04), 13/598,855 , 30 Aug 2012; 11 pp, 3 Feb 2015.

133. Greve, D. R. and O'Neill, E. D.: Water-resistant gypsum products, January 1976.

134. Coverage - Google Help. https://support.google.com/faqs/answer/7049585.

135. About Google Scholar. https://scholar.google.com/intl/en/scholar/about.html.

136. Danielyan, T. and Zuev, K.: Similar document search, November 2015.

137. Stav, E., Burkard, E. A., and Finkelstein, R. S.: Cementitious gypsum-containing compositions and materials made therefrom, February 1998.

138. Stav, E., Burkard, E. A., Finkelstein, R. S., Winkowski, D. A., Metz, L. J., and Mudd, P. J.: Cementitious gypsum-containing binders and compositions and materials made therefrom, January 1999.

139. Bezerra, F. and Wainer, J.: Anomaly detection algorithms in logs of process aware systems. In *Proceedings of the 2008 ACM Symposium on Applied Computing* , pages 951–952. ACM, 2008.

140. Bezerra, F. and Wainer, J.: Fraud detection in process aware systems. *International Journal of Business Process Integration and Management* , 5(2):121–129, 2011.

141. Bezerra, F. and Wainer, J.: Algorithms for anomaly detection of traces in logs of process aware information systems. *Information Systems* , 38(1):33–44, 2013.

142. Lilleberg, J., Zhu, Y., and Zhang, Y.: Support vector machines and word2vec for text classification with semantic features. In *Cognitive Informatics & Cognitive Computing (ICCI* CC), 2015 IEEE 14th International Conference On* , pages 136–140. IEEE, 2015.

143. McClure, N.: *TensorFlow Machine Learning Cookbook: Explore Machine Learning Concepts Using the Latest Numerical Computing Library - TensorFlow - with the Help of This Comprehensive Cookbook* . s.l, Packt Publishing - ebooks Account, February 2017.

144. Sampson, M.: The Evolution of the Enablement and Written Description Requirements Under 35 U.S.C. § 112 in the Area of Biotechnology. *Berkeley Technology Law Journal* , 15(3):1233–1274, 2000.

145. Tong, X. and Frame, J. D.: Measuring national technological performance with patent claims data. *Research Policy* , 23(2):133–141, March 1994.

146. Caruana, R. and Niculescu-Mizil, A.: An Empirical Comparison of Supervised Learning Algorithms. *Proceedings of the 23rd International Conference on Machine Learning* , pages 161–168, 2006.

147. Pakes, A., Simpson, M., Judd, K., and Mansfield, E.: Patent renewal data. *Brookings papers on economic activity. Microeconomics* , 1989:331–410, 1989.

148. Schmoch, U. and Kirsch, N.: Analysis of international patent flows. *Final report* , 1993.

VITA

VITA

NAME:	Salvatore Charles Immordino Jr.
EDUCATION:	Ph.D., Industrial Engineering, University of Illinois at Chicago, Chicago, Illinois, 2019
	M.S., Engineering Management, Marquette University, Milwaukee, Wisconsin, 2002
	B.S., Chemistry, Northern Illinois University, Dekalb, Illinois, 1992
EMPLOYMENT:	Director, Futuring & Digital Innovation, USG Corporation, 2018 - present
	Director, Performance Surfaces & Analytical Laboratory, USG Corporation, 2012 - 2018
	Program Manager, Performance Surfaces Laboratory, USG Corporation, 2008 - 2012
	Group Leader, Specialty Finishes Laboratory, USG Corporation, 2006 - 2008
	Senior Researcher, Interior Finishes Laboratory, USG Corporation, 2001 - 2006
	Researcher, Industrial Gypsum Products Laboratory, USG Corporation, 1995 - 1998
	Staff Researcher, Construction Plasters Laboratory, USG Corporation, 1992 - 1995
HONORS:	Beta Gamma Sigma Honor Society, Marquette University, Milwaukee, Wisconsin, 2002
AWARDS:	Chicago Innovation Award - SHEETROCK ® Lightweight All Purpose Joint Compound with Dust Control, Chicago, Il, 2006

PATENTS:	U.S. Patent 10,150,603: "Package for delivery of additives for powdered compositions", December 11, 2018
	U.S. Patent 9,944,443: "Water soluble package for delivery of additives for powdered compositions", April 17, 2018
	U.S. Patent 9,849,649: "Magnet receptive panels and methods", December 26, 2017
	U.S. Patent 9,783,998: "Nonwoven joint tape having low moisture expansion properties and method for using same", October 10, 2017
	U.S. Patent 9,376,824: "Nonwoven joint tape having low moisture expansion properties and method for using same", June 28, 2016
	U.S. Patent 9,334,410: "Use of aldehyde scavengers in interior building products", May 10, 2016
	U.S. Patent 9,328,023: "Low water drying type joint compound", May 3, 2016
	U.S. Patent 9,174,881: "Ready mixed setting type joint compound and set initiator in chambered pouch", November 3, 2015
	U.S. Patent 8,642,346: "Tagged joint compound and method of identification", February 4, 2014
	U.S. Patent 8,323,429: "Method for preparing three-dimensional plaster objects", December 4, 2012
	U.S. Patent 7,887,230: "Mixer having S-shaped paddles for mixing viscous materials", February 15, 2011
	U.S. Patent 7,543,708: "Plastic bag for fine powders", June 9, 2009
	U.S. Patent 7,516,909: "Continuous slurry dispenser apparatus", April 14, 2009

U.S. Patent 7,503,430: "Reduced dust acoustic panel", March 17, 2009

U.S. Patent D576,186: "Mixer for viscous materials", September 2, 2008

U.S. Patent 7,374,611: "Sprayable machinable media", May 20, 2008

U.S. Patent D566,143: "Mixer for viscous materials", April 8, 2008

U.S. Patent 6,673,144: "Joint compound providing low dusting and good gloss retention", January 6, 2004

U.S. Patent 6,545,066: "Lightweight ready-mix joint compound", April 8, 2003

U.S. Patent 6,409,823: "Hydration Enhancing Additives", June 25, 2002

U.S. Patent 6,406,537: "High-strength joint compound", June 18, 2002

U.S. Patent 6,398,864: "Pottery plaster formulations for the manufacture of plaster molds", June 4, 2002

U.S. Patent 6,355,099: "Plaster mixture for forming a machinable composition", March 12, 2002

U.S. Patent 6,273,345: "High performance slurry spray machine", August 14, 2001

U.S. Patent 5,534,059: "Machinable plaster", July 9, 1996

PROFESSIONAL MEMBERSHIPS:

American Chemical Society

Industrial Research Institute

Product Development and Management Association

SELECTED PRESENTATIONS:	Immordino, S (November, 2017). *Comparing Similarity of Patent Textual Data Through the Application of Unsurpervised Machine Learning.* Presented to attendees of the PDMA Research Forum at the PDMA annual conference, Swissotel Chicago, Chicago, IL.
SELECTED PUBLICATIONS:	Immordino, S. C., and Scott, M. J.: Comparing Similarity of Patent Textual Data Through the Application of Machine Learning. In preparation.

APPENDICES

Appendix A

PATENT DATA IMPORTATION

Listing A.1: Patent data importation python code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title                :1_20190220_patent_data_import.py
description          :Patent Analysis Machine Learning
author               :Salvatore Immordino
date created         :Mon Apr 18 19:04:13 2016
date last modified   :Wed Feb 20 18:07:29 2019
version              :0.1
python_version       :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions.

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.

This file uses the patent.tsv and raw_assignee.tsv files and merges the datasets
    ↪  and saves it to 'input_data1.csv'. Due to the large size of the patent.
    ↪ tsv file, this may take a few minutes to run
"""
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
import pandas as pd

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

#==============================================================================
# METHODS
#==============================================================================

# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'

# date parser for pandas import
parser = lambda x : pd.to_datetime(x, format='%Y-%m-%d %H:%M:%S',
errors='coerce')

# load the data; type 'patent_data.dtypes' to show dtypes
patent_data = pd.read_csv(working + path + 'patent.tsv',
                          sep='\t', # for tab delimited
                          header=0, # set header columns row 0
                          usecols = ['id','number','date'], # select columns
                          dtype = {'id':object,
                                   'number':object,
                                   'abstract':object,
                                   'title':object,
                                   'num_claims':float}, # sets dtype
                          index_col=['id'], # sets index column
                          na_values = ['no info', '.'],
                          parse_dates = ['date'],
                          date_parser=parser,
                          encoding = "iso-8859-1")

raw_assignee_data = pd.read_csv(working + path + 'rawassignee.tsv',
                                sep='\t',
                                header=0,
                                usecols = ['uuid',
                                           'patent_id',
                                           'organization'], # selects columns
                                dtype = {'uuid':object,
                                         'patent_id':object,
                                         'organization':object}, # sets dtype
                                index_col=['uuid'], # sets index column
                                na_values = ['no info', '.'],
                                encoding = "iso-8859-1")

# clean date columns in the patent table and select year only
patent_data['pat_year'] = patent_data['date'].dt.year
patent_data = patent_data.drop('date', 1)

# rename number column to patent_id
patent_data = patent_data.rename(index=str, columns={"number": "patent_id"})

# join the patent data with raw assignee data using patent number as key
merged = pd.merge(left=patent_data,
                  right=raw_assignee_data,
                  how='left',
                  left_on='patent_id',
                  right_on='patent_id',
                  sort=True,
                  left_index=True)

# save the data as a csv to the working directory
merged.to_csv(working + path + 'input_data1.csv', index=False)
```

Appendix B

PATENT DATA SELECTION

Listing B.1: Patent data selection python code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title               :2_20190220_patent_data_select.py
description         :Patent Analysis Machine Learning
author              :Salvatore Immordino
date created        :Mon Apr 18 19:04:13 2016
date last modified  :Wed Feb 20 18:07:29 2019
version             :0.1
python_version      :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions.

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.

This file uses the 'input_data1.csv' file from step 1 and the 'cpc_current.tsv'
    ↪ files. You can then create a list of organizations and then select only
    ↪ patents held by those organizations and save that to 'input_data2.csv"""
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
import pandas as pd

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

#==============================================================================
# METHODS
#==============================================================================

# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'

# load the data; type 'patent_data.dtypes' to show dtypes
input_data1 = pd.read_csv(working + path + 'input_data1.csv',
                          header=0, # set header columns row 0
                          usecols = ['patent_id',
                                     'pat_year',
                                     'organization'],
                          dtype = {'patent_id':object,
                                   'abstract':object,
                                   'title':object,
```

Appendix B (Continued)

```
                                 'num_claims':float},
                        na_values = ['no info', '.'],
                        converters={'organization': str},
                        encoding = "iso-8859-1")

cpc_current = pd.read_csv(working + path + 'cpc_current.tsv',
                        sep='\t',
                        header=0,
                        usecols = ['uuid',
                                   'patent_id',
                                   'section_id',
                                   'sequence'],
                        dtype = {'uuid':object,
                                 'patent_id':object,
                                 'citation_id':object,
                                 'sequence':float}, # sets dtype
                        index_col=['uuid'], # sets index column
                        na_values = ['no info', '.'],
                        iterator = True,
                        chunksize = 10000,
                        encoding = "iso-8859-1")

# select first listed CPC code to manage memory we do iterator and chunks
cpc_current  = pd.concat([chunk[chunk['sequence'] == 1] for chunk in cpc_current
    ↪ ])
cpc_current.head(10) # worked

# coloumn names & data types
list(input_data1)
input_data1.dtypes

# create org search for list, look for any that contain "armstrong"
organizations = input_data1['organization'].unique().tolist()
sub = 'Armstrong'
print ("\n".join(s for s in organizations if sub.lower() in s.lower()))

#==============================================================================
# search merged for construction orgs that contain gypsum or known competitors
# USG Corporation Competitors = https://www.nasdaq.com/symbol/usg/competitors
# regex expressions, e.g. '^(?=.*Saint)(?=.*Gobain)', anything that contains
#==============================================================================
mylist = ['gypsum',
          'Eagle Materials',
          '^(?=.*Armstrong)(?=.*World)',
          'USG Interiors, LLC',
          'USG Acoustical Products Company',
          'USG Industries, Inc.',
          'USG Corporation',
          '^(?=.*Lafarge)',
          'Continental Building',
          '^(?=.*Saint)(?=.*Gobain)',
          '^(?=.*James)(?=.*Hardie)',
          '^(?=.*Certain)(?=.*Teed)',
          '^(?=.*Georgia)(?=.*Pacific)',
          '^(?=.*Louisiana)(?=.*Pacific)']

# note: don't forget to screen patent citations for other competitors
# note: Saint Gobain has acquired both Hardie and Certainteed
pattern = '|'.join(mylist) # parse list of building companies

# first - count total patents from pattern organizations
```

Appendix B (Continued)

```python
patent_count = input_data1[input_data1['organization'].str.contains(pattern,case
    ↪ =False)==True].count()

# second - select only patents from the competitive list
input_data2 = input_data1[input_data1['organization'].str.contains(pattern, case
    ↪ =False)==True]

# third - join the selected patent data with cpc data using patent number as key
input_data2 = pd.merge(left=input_data2,
                       right=cpc_current,
                       how='left',
                       left_on='patent_id',
                       right_on='patent_id',
                       sort=True,
                       left_index=True)

# design patents don't have claims or section ID, remove them for now
input_data2 = input_data2[input_data2['patent_id'].str.contains('D', case=False)
    ↪ ==False]

# reissue patents exsist, e.g. RE45923 in 2016 is from U.S. Pat. No. 6,645,612,
    ↪ remove for now
input_data2 = input_data2[input_data2['patent_id'].str.contains('RE', case=False
    ↪ )==False]

# fourth rename number patent_year column to grant_date_year
input_data2 = input_data2.rename(index=str, columns={"pat_year": "
    ↪ grant_date_year"})

# fifth - save the data as a csv to the working directory
input_data2.to_csv(working + path + 'input_data2.csv', index=False)
```

Appendix C

PATENT DATA ORGANIZATION DISAMBIGUATION

Listing C.1: Patent data organization disambiguate python code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title                :3_20190220_patent_data_disambiguate.py
description          :Patent Analysis Machine Learning
author               :Salvatore Immordino
date created         :Sat Apr 30 09:36:03 2016
date last modified   :Wed Feb 20 19:12:17 2019
version              :0.1
python_version       :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions.

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.

'input_data2' from the previous step is copied as 'input_data3'. Some patents
    ↪ held by a company are registered under different permutations of the
    ↪ organization's name (ex: United States Gypsum Corporation, U.S., both for
    ↪  USG Corporation). This file renames all those to make them uniform (ex:
    ↪  under 'U.S.G. Corporation) and saves it to 'input_data4.csv'."""
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
import pandas as pd

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# extensions; conda install -c conda-forge ipython-autotime
# '%load_ext autotime' in header to show time

#==============================================================================
# METHODS
#==============================================================================
# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'

# ============================================================================
#load the patent data files
input_data3 = pd.read_csv(working + path + 'input_data2.csv',
                          header=0, # set header columns row 0
                          usecols = ['patent_id',
```

Appendix C (Continued)

```
                                     'grant_date_year',
                                     'organization',
                                     'section_id'],
                          dtype = {'patent_id':object,
                                   'abstract':object,
                                   'title':object,
                                   'section_id':object}, # set dtype
                         na_values = ['no info', '.'],
                         converters={'organization': str},
                         encoding = "iso-8859-1")
# ============================================================================
unique_org = input_data3['organization'].unique()

# merge the same companies or aquired assets, order matters!
disambiguate_before = input_data3.groupby('organization').count()

input_data3.loc[input_data3['organization'].str.contains('^(?=.*Armstrong)(?=.*
    ↪ World)',
                case=False)==True,'organization'] = 'Armstrong World Industries'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*Saint)(?=.*
    ↪ Gobain)',
                case=False)==True, 'organization'] = 'Saint Gobain'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*James)(?=.*
    ↪ Hardie)',
                case=False)==True, 'organization'] = 'James Hardie'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*Certain)(?=.*
    ↪ Teed)',
                case=False)==True, 'organization'] = 'CertainTeed'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*Georgia)(?=.*
    ↪ Pacific*)',
                case=False)==True, 'organization'] = 'Georgia Pacific'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*National)(?=.*
    ↪ Gypsum)',
                case=False)==True, 'organization'] = 'National Gypsum'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*Louisiana)(?=.*
    ↪ Pacific)',
                case=False)==True, 'organization'] = 'Louisiana Pacific'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*United)',
                case=False)==True, 'organization'] = 'USG Corporation'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*USG)',
                case=False)==True, 'organization'] = 'USG Corporation'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*Yoshino)',
                case=False)==True, 'organization'] = 'Yoshino Gypsum'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*Kaiser)',
                case=False)==True, 'organization'] = 'Kaiser Gypsum'

input_data3.loc[input_data3['organization'].str.contains('^(?=.*Lafarge)',
                case=False)==True, 'organization'] = 'Lafarge'

input_data3 = input_data3.replace(
        {'Unites States Gypsum Company': 'USG Corporation'},regex=True)

```

Appendix C (Continued)

```
input_data3 = input_data3.replace(
        {'U.S.': 'USG Corporation'}, regex=True)

input_data3 = input_data3.replace(
        {'USG Corporation Gypsum Company': 'USG Corporation'}, regex=True)

input_data3 = input_data3.replace(
        {'G-P Gypsum Corporation': 'Georgia Pacific.'}, regex=True)

input_data3 = input_data3.replace(
        {'GP Gypsum Corp.': 'Georgia Pacific'}, regex=True)

input_data3 = input_data3.replace(
        {'Gypsum Management.': 'Gypsum Management and Supply, Inc.'},
        regex=True)

input_data3 = input_data3.replace(
        {'GYPSUM MANAGEMENT AND SUPPLY, INC.': 'Gypsum Management and Supply,
    ↪ Inc.'},
        regex=True)

input_data3.loc[input_data3['organization'].str.contains('^(?=.*Geor)',
case=False)==True, 'organization'] = 'Georgia Pacific'

disambiguate_after = input_data3.groupby('organization').count()

# save the merged data with cpc classification data
input_data3.to_csv(working + path + 'input_data3.csv', index=False)
```

Appendix D

PATENT DATA FORWARD CITATION LOOKUP

Listing D.1: Patent data forward citation lookup python code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#================================================================================
'''
title                :4_20190220_patent_data_cit.py
description          :Patent Analysis Machine Learning
author               :Salvatore Immordino
date created         :Sat Apr 30 09:36:03 2016
date last modified   :Wed Feb 20 19:12:17 2019
version              :0.1
python_version       :3.7.1
'''
#================================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions.

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.

This file creates a copy of input_data3 is created as 'input_data4', and also
    ↪ uses 'uspatentcitations.tsv' and 'applications.tsv'. A list of cross
    ↪ citations and counts the selected patents from the citation table. It
    ↪ then goes through the patent numbers and finds every patent that
    ↪ backwards cites throught the entire patent database *(Note: this section
    ↪  takes a long time to run, I let it run overnight and saved the
    ↪ resulting file so it wouldn't have to be repeated)* and then looks up
    ↪ the application date for these patents. - This is then saved as '
    ↪ internal_citations.csv'. This data-frame is then combined with '
    ↪ input_data4' and saved as 'input_data4.csv
"""
#================================================================================
# IMPORT STATEMENTS
#================================================================================
# Libraries needed to run the tool; analysis:ignore
import pandas as pd
import numpy as np
import sys # needed for patent search loop status print

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# extensions; conda install -c conda-forge ipython-autotime
# '%load_ext autotime' in header to show time

#================================================================================
# METHODS
```

Appendix D (Continued)

```
#=============================================================================
# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'

# ===========================================================================

parser = lambda x : pd.to_datetime(x,format='%Y-%m-%d %H:%M:%S',errors='coerce')

#load the patent data files
input_data4 = pd.read_csv(working + path + 'input_data3.csv',
                          header=0,
                          usecols = ['patent_id',
                                     'grant_date_year',
                                     'organization',
                                     'section_id'],
                          dtype = {'patent_id':object,
                                   'abstract':object,
                                   'title':object,
                                   'section_id':object},
                          na_values = ['no info', '.'],
                          converters={'organization': str},
                          encoding = "iso-8859-1")

citations = pd.read_csv(working + path + 'uspatentcitation.tsv',
                        sep='\t',
                        header=0,
                        usecols = ['uuid',
                                   'patent_id',
                                   'citation_id',
                                   'date'],
                        dtype = {'uuid':object,
                                 'patent_id':object,
                                 'citation_id':object},
                        index_col=['uuid'],
                        na_values = ['no info', '.'],
                        parse_dates = ['date'],
                        iterator = True,
                        chunksize = 10000,
                        date_parser=parser,
                        encoding = "iso-8859-1")

# citations to manage memory (81 million citations) we do iterator and chunks
citations  = pd.concat(chunk for chunk in citations)

applications = pd.read_csv(working + path + 'application.tsv',
                           sep='\t',
                           header=0,
                           usecols = ['id','patent_id','date'], # select columns
                           dtype = {'id':object,'patent_id':object}, # set dtype
                           index_col=['id'], # sets index column
                           na_values = ['no info', '.'],
                           parse_dates = ['date'],
                           date_parser=parser,
                           encoding = "iso-8859-1")
# ===========================================================================

# First create list for cross citations within the building companies selected
patent_list = input_data4['patent_id'].unique().tolist()

# Second - count selected patents from citations table
citation_count = citations.loc[citations['citation_id'].isin(patent_list)].count
    ↪ ()
```

Appendix D (Continued)

```

# Third - select only cross citations from our patent list of 5K+ patents only
citations_cross = citations.loc[citations['patent_id'].isin(patent_list)]

#===============================================================================
'''
Below function to look up unique patents in the citation file
'''

## create empty dataframe
total_records = len(patent_list)

df = pd.DataFrame()

for i in range(0,total_records):
        df_temp = citations_cross.loc[citations_cross['citation_id'].astype(str)
    ↪  == str(patent_list[i])]
        df = pd.concat([df, df_temp], ignore_index=True)
        sys.stdout.write('\rUpdated record: ' + str(i) + ' of ' + str(
    ↪ total_records))
        sys.stdout.flush()

df.to_csv(working + path + 'immo_cit_no_date2.csv', index=False)
df['cit_pat_date'] = ""

for i in range(len(df)):
        print ('\r'),  # \r no line increment and overwrite of previous print
        sys.stdout.write("Cross matching citations to application dates: {0}%
    ↪ complete".format(np.float64(float(i)/len(df)*100).round(2)))
        sys.stdout.flush()
        df_temp = applications.loc[applications['patent_id'].astype(str) == df['
    ↪ patent_id'].iloc[i]]
        if df_temp.empty == False:
        df['cit_pat_date'].iloc[i] = df_temp['date'].iloc[0]

internal_citations = df
internal_citations.rename(columns={'patent_id':'citing_patent',
                                   'citation_id':'patent_id',
                                   'cit_pat_date':'cit_pat_year'},inplace=True)
internal_citations['cit_pat_year'] = internal_citations['cit_pat_year'].dt.year
internal_citations.drop(['date'])

list(internal_citations) # list column headers
internal_citations.to_csv(working + path + 'internal_citations.csv',index=False)

# ============================================================================
## why are the lists different?
## turns out some patents were never cited!
unique_patents = set(patent_list)
unique_patents_count = len(unique_patents)
unique_citations = set(df['citation_id'])
unique_citations_count = len(unique_citations)
set(unique_patents).intersection(unique_citations)
set(unique_patents)!=set(unique_citations)
no_citations = list(set(unique_patents)-set(unique_citations))
df.groupby('citation_id').count()
#==============================================================================

# load internal citations
internal_citations = pd.read_csv(working + path + 'internal_citations.csv',
                                 header=0, # set header columns row 0
                                 usecols = ['citing_patent',
```

Appendix D (Continued)

```
                                            'patent_id',
                                            'cit_pat_year'], # selects desired
    ↪ columns
                                 dtype = {'patent_id':object,
                                          'citing_patent':object}, # sets dtype
                                 na_values = ['no info', '.'],
                                 encoding = "iso-8859-1")

## join merged data with cited data, use the disambiguated input_data2
input_data4 = pd.merge(left=input_data4,
                       right=internal_citations,
                       how='left',
                       left_on='patent_id',
                       right_on='patent_id',
                       sort=True,
                       left_index=True)

## save the merged data with cited by data
input_data4.to_csv(working + path + 'input_data4.csv', index=False)
```

Appendix E

PATENT DATA TABLE CREATION

Listing E.1: Patent data table creation python code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#===============================================================================
'''
title                :5_20190220_patent_data_table.py
description          :Patent Analysis Machine Learning
author               :Salvatore Immordino
date created         :Sun May 01 15:32:06 2016
date last modified   :Wed Feb 20 19:12:17 2019
version              :0.1
python_version       :3.7.1
'''
#===============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions.

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.

This file uses 'input_data3.csv' and 'input_data4.csv' from the previous steps
    ↪ and merges patent data with CPC (Cooperative Patent Classification
    ↪ system) using patent number as key, and saves it to 'input_data5.csv'."""
    ↪
#===============================================================================
# IMPORT STATEMENTS
#===============================================================================
# Libraries needed to run the tool; analysis:ignore
import pandas as pd

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# drop the warnings
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

# extensions; conda install -c conda-forge ipython-autotime
# '%load_ext autotime' in header to show time

#===============================================================================
# METHODS
#===============================================================================
# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'

# ==============================================================================
```

Appendix E (Continued)

```
# load the patent data files
org_lookup = pd.read_csv(working + path + 'input_data3.csv',
                         header=0, # set header columns row 0
                         usecols = ['patent_id',
                                    'organization'], # selects desired columns
                         dtype = {'patent_id':object}, # sets dtype
                         na_values = ['no info', '.'],
                         converters={'organization': str},
                         encoding = "iso-8859-1")

# note: remember that cit_pat_date is the application year cited
org_lookup = org_lookup.rename(index=str,
                               columns={"patent_id": "citing_patent",
                                        "organization":"citing_org"})

input_data5 = pd.read_csv(working + path + 'input_data4.csv',
                          dtype = {'patent_id':object,
                                   'citing_patent':object},
                          header=0,
                          encoding = "iso-8859-1")

# join selected patent data with cpc data using patent number as key
input_data5 = pd.merge(left=input_data5,
                       right=org_lookup,
                       how='left',
                       left_on='citing_patent',
                       right_on='citing_patent',
                       sort=True,
                       left_index=True)

input_data5.dtypes
input_data5['organization'].nunique()

## save the merged data with cited by data
input_data5.to_csv(working + path + 'input_data5.csv', index=False)
```

Appendix F

PATENT DATA LABEL ENCODING

Listing F.1: Patent data label encoding python code

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title                :5_20190220_patent_data_encoding.py
description          :Patent Analysis Machine Learning
author               :Salvatore Immordino
date created         :Thu Nov 08 14:00:06 2018
date last modified   :Wed Feb 20 19:12:17 2019
version              :0.1
python_version       :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions.

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.

This file uses 'input_data5.csv' and creates an encoding for the 'organization'
    ↪ and 'section_id' column using the LabelEncoder from scikit-learn. It
    ↪ creates another dataframe with the CPC definition list and encoded
    ↪ labels. Three files are saved at the end of this, 'input_data6.csv', '
    ↪ labels_cpc.csv' and 'labels_org.csv'"""
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
# Libraries needed to run the tool; analysis:ignore
import pandas as pd

# switch categorical letters to numbers
from sklearn.preprocessing import LabelEncoder

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# drop the warnings
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

# extensions; conda install -c conda-forge ipython-autotime
# '%load_ext autotime' in header to show time

#==============================================================================
# METHODS
#==============================================================================
```

Appendix F (Continued)

```
# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'

#===============================================================================
# load the patent data files
input_data6 = pd.read_csv(working + path + 'input_data5' + '.csv',
                          header=0, # set header columns row 0
                          dtype = {'patent_id':object,
                                   'citing_patent':object}, # sets dtype
                          na_values = ['no info', '.'],
                          converters={'organization': str},
                          encoding = "iso-8859-1")

# ==============================================================================
# encode organizations for y axis
le = LabelEncoder() #used to turn categorical letters to numbers
le.fit(input_data6.organization)
input_data6['org_number'] = (le.transform(input_data6.organization))

# create organization encoded table
labels_org = pd.DataFrame(list(le.classes_))
labels_org.columns = ['organization']
labels_org # there should be 20 (0-19) sans 3M

# replace the NaN with None in CPC
input_data6['section_id'] = input_data6['section_id'].fillna('0')

# encode section_id for colors
le = LabelEncoder() #used to turn categorical letters to numbers
le.fit(input_data6.section_id)
input_data6['class_number'] = (le.transform(input_data6.section_id))*1.0
input_data6['class_number'].unique()

# create class encoded dataframe
labels_cpc = pd.DataFrame(list(le.classes_))
labels_cpc.columns = ['patent_class']

cpc_definition_list = pd.DataFrame({'patent_class':['0','A','B','C','D','E',
                                                    'F','G','H','Y'],
                                    'definition':['X = Not Marked',
                                                  'A = Human Necessitites',
                                                  'B = Performing Operations',
                                                  'C = Chemistry; Metallurgy',
                                                  'D = Textiles; Paper',
                                                  'E = Fixed Constructions',
                                                  'F = Mechanical Engineering',
                                                  'G = Physics',
                                                  'H = Electricity',
                                                  'Y = New Technology']})

labels_cpc = pd.merge(labels_cpc, cpc_definition_list, on=['patent_class'])

input_data6.info(verbose=True) # tell me all the things

input_data6 = input_data6[['patent_id', # organize the table
                           'title',
                           'abstract',
                           'grant_date_year',
                           'organization',
                           'org_number',
                           'section_id',
                           'class_number',
```

Appendix F (Continued)

```
                           'citing_patent',
                           'cit_pat_year',
                           'citing_org']]

input_data6.to_csv(working + path + 'input_data6.csv', index=False)
labels_cpc.to_csv(working + path + 'labels_cpc.csv', index=False)
labels_org.to_csv(working + path + 'labels_org.csv', index=False)
```

Appendix G

PATENT DATA TF-IDF & CSS MATRIX

Listing G.1: Patent data tf-idf and CSS matrix creation python code

```
 1  #!/usr/bin/env python3
 2  # -*- coding: utf-8 -*-
 3  #==============================================================================
 4  '''
 5  title               :7_20190311_TFIDF_Cosine_Sim.py
 6  description         :Patent Analysis Machine Learning
 7  author              :Salvatore Immordino
 8  date created        :Sun Nov 13 11:26:06 2018
 9  date last modified  :Fri Mar 22 19:12:17 2019
10  version             :0.1
11  python_version      :3.7.1
12  '''
13  #==============================================================================
14  """Big data analysis of patent documents on python. The aim of this research
15  was to determine if machine learning could be used to study inventiveness and
16  inventive knowledge flow independent of historical methods like citation
17  analysis. Using natural language processing and network analysis, this research
18  aimed to identify relatedness between patents based on the inventive language
19  used by patent seekers to describe their inventions.
20
21  Data files were downloaded from the United States Patent Office PatensView Data
        ↪ Download located at www.patentsview.org.
22
23  This file uses input_data6.csv and adds which claims were made to this dataframe.
        ↪  It also adds when the patent was granted, and saves the new csv as '
        ↪ input_data7.csv'. Lines 145 to 167 aims to fix data and encoding errors
        ↪ and 168 on-wards performs some prepossessing such as removing
        ↪ punctuation, lemmatization and defines a function to clean the corpus to
        ↪  create a jargon free doc. Line 317 on-wards performs some
        ↪ visualizations using seaborn and wordclouds. 372 on-wards is the TFIDF
        ↪ Vectorizer and Cosine similarity analysis, with some more visualizations
        ↪  towards the end"""
24  #==============================================================================
25  # IMPORT STATEMENTS
26  #==============================================================================
27  # basic libraries needed to run the tool
28  import pandas as pd
29  import numpy as np
30
31  # time stamping
32  import time
33
34  # create strings and fix encoding errors
35  import string
36
37  # graphing functions
38  import matplotlib.pyplot as plt
39  from wordcloud import WordCloud
40
41  # drop the warnings
42  import warnings
43  warnings.filterwarnings("ignore") # supress all warnings
44
```

Appendix G (Continued)

```
# natural language tools - clean and preprocess
from nltk.corpus import stopwords
from nltk import FreqDist

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# needed for this code
import seaborn as sns

# need to remove multiple letter nonsense words like 'aaa'
import re

# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'
filename = '7_20190311_TFIDF_cosine_sim'


# =============================================================================
#  Data importation
# =============================================================================
# load the patent data files sans textual data
input_data7 = pd.read_csv(working + path + 'input_data6' + '.csv', header=0,
                          usecols = ['patent_id',
                                     'title',
                                     'abstract',
                                     'grant_date_year',
                                     'organization',
                                     'org_number',
                                     'section_id',
                                     'class_number'],
                                     dtype = {'patent_id':object})

# drop duplicates (5651)
input_data7 = input_data7.drop_duplicates()
input_data7.info(verbose=True) # tell me all the things

# =============================================================================
# Import full claims text from patentsview
# =============================================================================
# iterate through claims data to avoid memory error
# iter_csv = pd.read_csv('claim.tsv',
#                          sep='\t',
#                          header=0,
#                          usecols = ['patent_id',
#                                     'text',
#                                     'sequence'],
#                          dtype = {'patent_id':object,
#                                   'text':object,
#                                   'sequence':object},
#                          na_values = ['no info', '.'],
#                          iterator = True,
#                          chunksize = 10000,
#                          encoding = "iso-8859-1")
#
# claims  = pd.concat(chunk for chunk in iter_csv)
#
# patent_local_claims = claims[claims['patent_id'].isin(input_data7['patent_id'].
    ↪ tolist())]
```

Appendix G (Continued)

```
#
# patent_local_claims = patent_local_claims[['patent_id',
#                                            'sequence',
#                                            'text']]
#
# patent_local_claims["sequence"] = pd.to_numeric(claims["sequence"])
#
# patent_local_claims = patent_local_claims.sort_values(['patent_id',
#                                                         'sequence',
#                                                         'text'],
#                                                        ascending=[True,
#                                                                   True,
#                                                                   False])
#
# patent_local_claims = patent_local_claims.rename(columns={'text':'claims'})
#
# patent_local_claims.to_csv(working + path + 'patent_local_claims.csv',
#                            index=False)
#
# patent_local_claims.info(verbose=True)
# ============================================================================
patent_local_claims = pd.read_csv(working + path + 'patent_local_claims.csv',
                                  header=0,
                                  dtype = {'patent_id':object,
                                           'sequence':int,
                                           'claims':object})
# ============================================================================
patent_claims_combined = patent_local_claims.groupby(
        ['patent_id'])['claims'].apply(', '.join).reset_index()

input_data7 =  pd.merge(left=input_data7,
                        right=patent_claims_combined,
                        how='left',
                        left_on='patent_id',
                        right_on='patent_id',
                        sort=True,
                        left_index=True)

input_data7['combined'] = input_data7[['title','abstract','claims']].apply(
        lambda x: ' '.join(x.astype(str)), axis=1)

# ============================================================================
# Import full grant data from patentsview patents.tsv
# ============================================================================
# parser = lambda x : pd.to_datetime(x, format='%Y-%m-%d %H:%M:%S', errors='
    ↪ coerce')

# grant_dates = pd.read_csv(working + path + 'patent' + '.tsv',
#                           #nrows=10000, # this just takes the top 20 rows for
    ↪ speed
#                           sep='\t', # for tab delimited
#                           header=0, # set header columns row 0
#                           usecols = ['id','number','date'], # selects desired
    ↪ columns
#                           dtype = {'id':object,
#                                    'number':object,
#                                    'abstract':object,
#                                    'title':object,
#                                    'num_claims':float}, # sets dtype
#                           index_col=['id'], # sets index column
#                           na_values = ['no info', '.'],
#                           parse_dates = ['date'],
```

Appendix G (Continued)

```
#                           date_parser=parser,
#                           encoding = "iso-8859-1")

# grant_dates = grant_dates.rename(columns={'number':'patent_id'})
# grant_dates = grant_dates.rename(columns={'date':'grant_date'})
# grant_dates = grant_dates[grant_dates['patent_id'].isin(input_data7['patent_id'].tolist())]
# grant_dates.to_csv(working + path + 'grant_dates.csv', index=False)
# grant_dates.info(verbose=True)
# ==============================================================================
grant_dates = pd.read_csv(working + path + 'grant_dates.csv',
                                            header=0,
                                            dtype = {'patent_id':object},
                                            parse_dates = ['grant_date'])
# ==============================================================================
input_data7 = pd.merge(left=input_data7,
                       right=grant_dates,
                       how='left',
                       left_on='patent_id',
                       right_on='patent_id',
                       sort=True,
                       left_index=True)

# ------------------------------------------------------------------------------
#  Fix data and encoding errors
# ------------------------------------------------------------------------------
'''
https://stackoverflow.com/questions/16467479/normalizing-unicode
https://stackoverflow.com/questions/49891778/conversion-utf-to-ascii-in-python-with-pandas-dataframe
'''

import unicodedata # to fix encoding errors
input_data7['combined'] = input_data7['combined'].apply(
        lambda val: unicodedata.normalize('NFKD', val).encode(
                'ascii', 'ignore').decode())

# remove encoding replacements for subscript, superscript, and degrees
input_data7['combined'] = input_data7['combined'].str.replace(".sub.", "") # removes subset
input_data7['combined'] = input_data7['combined'].str.replace(".sup.", "") # removes superscript
input_data7['combined'] = input_data7['combined'].str.replace(".degree",
           " degrees")

## get unique list of patent #'s from patent_text and parse the list
# input_data7.patent_id.nunique() # count number of patents (5651)
my_columns = input_data7['patent_id'].tolist() # create patent list
corpus = input_data7['combined'].tolist()
patent_numbers = '|'.join(my_columns)

# ==============================================================================
#  Preprocessing
# ==============================================================================
'''
https://stackoverflow.com/questions/40568948/load-local-resources-with-nltk
https://pythonprogramming.net/nltk-corpus-corpora-tutorial/
https://stackoverflow.com/questions/10467024/how-do-i-create-my-own-nltk-text-from-a-text-file
https://stackoverflow.com/questions/22350879/removing-single-quotation-marks-while-preserving-apostrophes-python-nltk
'''
```

Appendix G (Continued)

```
# ------------------------------------------------------------------------------
#  Create jargon lexical
# ------------------------------------------------------------------------------
# jargon removes word "containing", we want to weigh lignosulfonate that
# follows, this proximity is important for future research'''

# Created a list of common patent terms

#jargon == {} # for pre-jargon bar plot
jargon = {'according', 'also', 'apparatus', 'assembly', 'body', 'claim',
          'claimed', 'component', 'composition', 'comprise', 'comprises',
          'comprising', 'consisting', 'containing', 'device', 'disclosed',
          'element', 'embodying', 'end', 'face', 'first', 'form', 'formed',
          'forming', 'forms', 'group', 'include', 'includes', 'including',
          'invention', 'layer', 'le', 'least', 'made', 'making', 'material',
          'may', 'mean', 'means', 'member', 'method', 'mixture', 'one',
          'patent', 'plurality', 'portion', 'preferably', 'present',
          'process', 'product', 'provided', 'provides', 'providing', 'relates',
          'resulting', 'said', 'second', 'selected', 'substantially',
          'substrate', 'support', 'surface', 'system', 'technology', 'thereof',
          'third', 'two', 'web', 'weight', 'wherein', 'within', 'wt'}

# ------------------------------------------------------------------------------
#  Stop words, punctuation, lemmatization, and word length steps
# ------------------------------------------------------------------------------

# natural language tool kit
#import nltk

# stop words - a list of low value words
#nltk.download("stopwords")  # download list'
stop_words = set(stopwords.words('english'))

# ------------------------------------------------------------------------------
# we may wish to remove the word 'to' from stop_words because it is
# commonly used to define the best mode invention boundaries
# ------------------------------------------------------------------------------

# Punctuation
#nltk.download('punkt')
punctuations = set(string.punctuation)
punctuations.remove('-') # maybe remove hyphens
#punctuations.remove('/')

# Lemmatization
from nltk.stem.wordnet import WordNetLemmatizer
lemma = WordNetLemmatizer()

# ------------------------------------------------------------------------------
# attempt to find closest noun, this seemed to reduce the unique valued added
# nouns and makes me question whether we should use parts of speech (POS) at all
# ------------------------------------------------------------------------------

# set minimum word length to include
'''word_len = ?  to set length of words, undesired, removes chemical elements
like fe for 'iron or' c for 'carbon'; mgo 'magnesium oxide', ca = calcium'''
word_len = 2

# ==============================================================================
#  Spelling Auto-correct
# ==============================================================================
'''
```

Appendix G (Continued)

```
https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-
    ↪ data-predictive-python/
https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-
    ↪ beginners-using-textblob/
conda install -c conda-forge textblob
'''
# ==============================================================================
# from textblob import TextBlob
# from textblob import Word
#
# def spell_correct(word_list):
#     try:
#         corrected = []
#         for word in word_list:
#             w = Word(word)
#             corrected.append(w.correct())
#         return corrected
#     except UnicodeDecodeError:
#         return None
#
# test = [spell_correct(x) for x in corpus]
#
# lines = input_data7['combined']
#
# line1 = TextBlob(lines[0])
# print(line1.correct())
#
# for line in lines:
#     # TextBlob is providing correct method
#     print(TextBlob(line).correct())
# ==============================================================================

#-----------------------------------------------------------------------------
#  Tokenize words
# ------------------------------------------------------------------------------
'''
https://machinelearningmastery.com/clean-text-machine-learning-python/
https://stackoverflow.com/questions/24695092/how-to-not-remove-apostrophe-only-
    ↪ for-some-words-in-text-file-in-python
https://machinelearningmastery.com/clean-text-machine-learning-python/
http://dsgeek.com/2018/02/19/tfidf_vectors.html
https://stackoverflow.com/questions/8897593/similarity-between-two-text-
    ↪ documents?rq=1
https://stackoverflow.com/questions/43451906/load-column-in-csv-file-into-spacy
https://spacy.io/usage/linguistic-features#tokenization
https://stackoverflow.com/questions/45547568/how-can-i-prevent-tfidfvectorizer-
    ↪ to-get-numbers-as-vocabulary
https://www.oreilly.com/learning/how-can-i-tokenize-a-sentence-with-python
https://spacy.io/usage/linguistic-features#tokenization
https://stackoverflow.com/questions/43451906/load-column-in-csv-file-into-spacy
https://spacy.io/api/tokenizer
https://stackoverflow.com/questions/46981137/tokenizing-using-pandas-and-spacy
https://spacy.io/usage/spacy-101
https://www.kaggle.com/zackakil/nlp-using-word-vectors-with-spacy-cldspn
https://explosion.ai/blog/sense2vec-with-spacy
https://towardsdatascience.com/machine-learning-for-text-classification-using-
    ↪ spacy-in-python-b276b4051a49
'''
def clean(doc):
    number_free = ''.join([c for c in doc if c not in "1234567890"])
    words = [word.strip(string.punctuation) for word in number_free.split(" ")]
    filtered = [f for f in words if f and f.lower() not in stop_words]
```

Appendix G (Continued)

```
    undo = "".join([" "+i if not i.startswith("'") and i not in string.
    ↪ punctuation else i for i in filtered]).strip()
    punc_free = ''.join(ch for ch in undo if ch not in punctuations)
    smallword_free = ' '.join([w for w in punc_free.split() if len(w)>word_len])
    lemmatized = " ".join(lemma.lemmatize(word) for word in smallword_free.split
    ↪ ())
    jargon_free = " " .join([j for j in lemmatized.lower().split() if j not in
    ↪ jargon])
    for i in jargon_free:
        jargon_free = re.sub((i+i+i), ' ', jargon_free)
        #jargon_free = jargon_free.replace('the ',' ') # not need with proper
    ↪ space inserted on merge
    nonsense = ' '.join([w for w in jargon_free.split() if len(w)>1])
    return nonsense

corpus_clean = [clean(doc) for doc in corpus] # list of sentance strings
corpus_tokenize = [clean(doc).split() for doc in corpus]  # list of string words

# bool(set(corpus_clean).intersection(corpus_clean1)) # compare same = true
# need to remove 'the' & 'a' from end of words - fixed via claim merge step with
    ↪  space
# dust_control_after = dust_control_after.replace('the ',' ')

# ============================================================================
#  Word Frequency analysis
# ============================================================================
'''https://stackoverflow.com/questions/46486157/how-to-remove-every-word-with-
    ↪ non-alphabetic-characters'''
'''https://stackoverflow.com/questions/44810269/the-wordcloud-formed-is-showing-
    ↪ apostrophe-sign?noredirect=1&lq=1'''
'''http://www.dzhaworks.com/blog/visualizing-a-cloud-of-skills-with-python/'''
'''https://stackoverflow.com/questions/32313206/force-wordcloud-python-module-to
    ↪ -include-all-words'''

documents = corpus_clean
documents_words = ' '.join(corpus_clean).split()

from collections import Counter
info_counts = Counter(documents_words)

info_common_words = [word[0] for word in info_counts.most_common(30)]
info_common_counts = [word[1] for word in info_counts.most_common(30)]

# use to find words
# input_data7['combined'].str.contains('le')

# ----------------------------------------------------------------------------
# Bar plot of wards prior to jargon removal - use empty set
# ----------------------------------------------------------------------------
# ============================================================================
# plt.figure(figsize=(15,8))
# sns.barplot(x=info_common_words, y=info_common_counts)
# sns.set(font_scale=.90)
# plt.xticks(rotation=60, ha="right")
# plt.title('Most common words used in patents for building industry without
    ↪ jargon removal')
# plt.savefig(working + path + 'img/common_words pre-jargon',bbox_inches='tight
    ↪ ',dpi=300)
# plt.show()
# ============================================================================

# ----------------------------------------------------------------------------
# Bar plot of words after jargon removal
# ----------------------------------------------------------------------------
''''https://www.kdnuggets.com/2018/09/machine-learning-text-classification-using
    ↪ -spacy-python.html'''

plt.figure(figsize=(15,8))
sns.barplot(x=info_common_words, y=info_common_counts)
sns.set(font_scale=.90)
plt.xticks(rotation=60, ha="right")
plt.title('Most common words used in patents for building industry after jargon
    ↪ removal')
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename + '_' + 'common_words_post_jargon', bbox_inches='tight', dpi
    ↪ =300)
plt.show()

# ----------------------------------------------------------------------------
# Frequency distribution
# ----------------------------------------------------------------------------
tokens = documents_words

fd = FreqDist(tokens)
fd.most_common(100)

# frequency of top 100 words
fd.plot(35)
#plt.savefig(working + path + 'img/word frequency plot',bbox_inches='tight',dpi
    ↪ =300)

# adds cummulative count from left to right
fd.plot(35, cumulative=True)
#plt.savefig(working + path + 'img/word frequency cummulative',bbox_inches='
    ↪ tight',dpi=300)

# ----------------------------------------------------------------------------
# Wordcloud visualization
# ----------------------------------------------------------------------------
wordcloud = WordCloud(width=800,
                      height=400,
                      max_font_size=100,
                      #max_words=100,
                      ranks_only = True,
                      background_color="white",
                      stopwords=set()).generate_from_frequencies(fd)

plt.figure(figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.tight_layout(pad=0)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename + '_' + 'wordcloud.png', facecolor='k', bbox_inches='tight')

# ----------------------------------------------------------------------------
# Patent abstract before and after
# ----------------------------------------------------------------------------
'''https://text-compare.com/''' # used this to compare text output

#dust_control_before = input_data7.loc[input_data7['patent_id'] == '6673144' ] #
    ↪  index 2780
#print (corpus_clean[2780])
#dust_control_before = input_data7.loc[input_data7['patent_id'] == '8323429' ] #
    ↪  index 2780
```

Appendix G (Continued)

```

# need to remove multiple of letters like 'aaaaaaaaaaaaaaa'

#dust_control_after = corpus_clean[2780]

#for i in dust_control_after:
#    dust_control_after = re.sub((i+i+i), ' ', dust_control_after)
#    dust_control_after = ' '.join([w for w in dust_control_after.split() if len
    ↪ (w)>1])

# need to remove 'the' & 'a' from end of words - fixed via claim merge step with
    ↪  space
#dust_control_after = dust_control_after.replace('the ',' ')


#===============================================================================
#  Term Frequency * Inverse Document Frequency, Tf-Idf
#===============================================================================
'''https://stackoverflow.com/questions/12118720/python-tf-idf-cosine-to-find-
    ↪ document-similarity?rq=1'''
'''https://radimrehurek.com/gensim/tut2.html'''

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vect = TfidfVectorizer(max_features=None,      # use all the word terms
                             max_df=0.65,            # ignore used everywhere
                             min_df=0.0,             # ignore no rares words
                             ngram_range=(1, 3),     # unigram thru trigram
                             analyzer=lambda x:[w for w in x if w not in
    ↪ stop_words],
                             strip_accents='unicode') # remove accents

tfidf = tfidf_vect.fit_transform(corpus_tokenize)
tfidf

feature_names = tfidf_vect.get_feature_names()

#===============================================================================
#  Document Term Matrix (dtm)
#===============================================================================
# Convert sparse matrix (DTM) to dataframe to see word frequencies.
dtm = tfidf.todense() # doc_term_matrix (dtm)
df_dtm = pd.DataFrame(dtm, columns=feature_names, index=[input_data7.patent_id])
df_dtm.head(5)

#===============================================================================
#  Cosine Similarity Matrix
#===============================================================================
'''https://www.machinelearningplus.com/nlp/cosine-similarity/'''

from sklearn.metrics.pairwise import linear_kernel
cosine_similarities_row_1 = linear_kernel(tfidf[0:1], tfidf).flatten()
cosine_similarities_row_1 # this is just the first row patent

# Compute CSS (Cosine Similarity Scores)
from sklearn.metrics.pairwise import cosine_similarity
print(cosine_similarity(tfidf, tfidf))
cosine_similarities = cosine_similarity(tfidf, tfidf)

# show top 5 (-6) related patents (-1 = index #) to first patent based on CSS
related_docs_indices = cosine_similarities_row_1.argsort()[:-20:-1]
related_docs_indices
```

Appendix G (Continued)

```

print (documents[0])
print (documents[25])
print (documents[226])
print (documents[3761])
print (documents[3693])
print (documents[28])
print (documents[1235])
print (documents[5635])
print (documents[1485])
print (documents[3995])
print (documents[281])

# ==============================================================================
#  Cosine Similarity Score Statistics & Histrogram
# ==============================================================================
'''https://stackoverflow.com/questions/33203645/how-to-plot-a-histogram-using-
    ↪ matplotlib-in-python-with-a-list-of-data'''

# ------------------------------------------------------------------------------
# remove identity matrix
# ------------------------------------------------------------------------------
def identity(n):
    m=[[0 for x in range(n)] for y in range(n)]
    for i in range(0,n):
        m[i][i] = 1
    return m

n = np.size(cosine_similarities,0) # count size

z = identity(n) # create identity
np.mean(z) # verify

# decided to remove self pairs instead
y = cosine_similarities - z # subtract identity'''
#y = cosine_similarities # no longer subtract identity

data = y.flatten() # flatten the array to the set of cosine similarities
'''is_zero = np.absolute(data) < np.finfo(float).eps # change floating point'''
'''data[is_zero] = np.nan # replace zero's with nans'''

# ------------------------------------------------------------------------------
# mu is the mean
# median is the middle value
# mode is the number that is repeated more often than any other
# ------------------------------------------------------------------------------
# ------------------------------------------------------------------------------
# histogram(s) of css for patent data
# ------------------------------------------------------------------------------
plt.figure(figsize=(20,10))
#plt.hist(data[~np.isnan(data)], bins='auto')
plt.hist(data[~np.isnan(data)], 1000)
plt.xlim(0, 0.2)
#plt.ylim(0, 3000000)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename + '_' + 'css_hist_plt.png', facecolor='k', bbox_inches='tight')

#------------------------------------------------------------------------------
plt.figure(figsize=(20,10))
sns.set_style('darkgrid')
sns.distplot(data[~np.isnan(data)])
#sns.countplot(data[~np.isnan(data)])
```

Appendix G (Continued)

```
plt.set_xlim(0, 0.1)
#plt.set_ylim(0, 100)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename + '_' + 'css_hist_sea.png', facecolor='k', bbox_inches='tight')

#-------------------------------------------------------------------------------
# best fit of data
from scipy.stats import norm
import matplotlib.mlab as mlab

(mu, sigma) = norm.fit(data[~np.isnan(data)]) # np.isnana to ignore nans stats

mean = np.mean(data[~np.isnan(data)])
median = np.median(data[~np.isnan(data)])
minimum = np.min(data[~np.isnan(data)])
maximum = np.max(data[~np.isnan(data)])

fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111)

#the histogram of the data
n, bins, patches = ax.hist(
        data[~np.isnan(data)], 500, density=1, facecolor='green', alpha=0.75)

bincenters = 0.5*(bins[1:]+bins[:-1])

# add a 'best fit' line for the normal PDF
z = mlab.normpdf( bincenters, mu, sigma)
l = ax.plot(bincenters, z, 'r--', linewidth=1)

# plot
ax.set_xlabel('Cosine Similarities')
ax.set_ylabel('Probability')

#title = r'$\mathrm{Histogram\ of\ Patents:}\ \mu=%.2f,\ \sigma=%.2f' % (mu,
    ↪ sigma)
title = "Fit results: mu = %.3f,  std = %.3f" % (mu, sigma)
ax.set_title(title)
ax.set_xlim(0, 0.1)
ax.set_ylim(0, 100)
ax.grid(True)
plt.axvline(mu, color='b', linestyle='dashed', linewidth=2)
plt.axvline(median, color='orange', linestyle='dashed', linewidth=2)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename + '_' + 'css_histogram.png', facecolor='k', bbox_inches='tight')
    ↪

# -----------------------------------------------------------------------------
#  Create Cosine Similarity Score (CSS) Dataframe
# ----------------------------------------------------------------------------
df_css = pd.DataFrame(cosine_similarities) # convert css matrix to datafram
df_css.columns = my_columns # change columns names to patent numbers
df_css.index = my_columns # set index to patent_id before stack
df_css_stack = df_css
df_css = df_css.stack().reset_index() # stack & reset index
df_css.columns = ['patent_id','css_patent_id','css'] # rename columns
df_css = df_css[df_css['patent_id'] != df_css['css_patent_id']] # remove self
df_css = df_css.sort_values(
        ['patent_id','css',],ascending=[True, False]).reset_index(drop=True)
df_css.to_csv(working + path + 'df_css.csv', index=False) # save css df
#-------------------------------------------------------------------------------
```

```
df_css = pd.read_csv(working + path + 'df_css.csv', header=0,
                                  dtype = {'patent_id':object,
                                           'css_patent_id':object})

# working data frame
# keeping zero's now too
df_css_zero = df_css
#df_css = df_css[df_css['css'] != 0] # remove css zero
df_css.head(20) # sans zero

#==============================================================================
#  Minimum Cosine Similarity Score (CSS) Threshold
#==============================================================================
'''
https://www.researchgate.net/post/
    ↪ Determination_of_threshold_for_cosine_similarity_score
https://stackoverflow.com/questions/30089675/clustering-cosine-similarity-matrix
    ↪ ?rq=1
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.9729&rep=rep1&type=
    ↪ pdf
https://stackoverflow.com/questions/21374375/semantic-relatedness-algorithms-
    ↪ python/21377351#21377351
'''
# ----------------------------------------------------------------------------
# Cosine Similarity Score (CSS)
# when working on a corpus of documents, we use a threshold formula:
# average(cosine_similarities)+alpha*standard_deviation(cosine_similarities)
# where lpha is a parameter and sigma is the standard deviation.
# ----------------------------------------------------------------------------

np.mean(df_css['css']) # matches mu
np.std(df_css['css'])
np.mean(df_css_zero['css']) # matches mu
np.std(df_css_zero['css'])
ave_css = mu # average(cos_similarity_matrix)
std_css = sigma  # standard_deviation(cos_similarity_matrix)
med_css = median # is the middle of the list
alpha = 6 # alpha parameter

# CSS minimum threshold
t = (ave_css + (alpha * std_css)) #

# ----------------------------------------------------------------------------
# calculate alpha and threshold value plot
# ----------------------------------------------------------------------------

def xfrange(start, stop, step):
    i = 0
    while start + i * step < stop:
        yield start + i * step
        i += 1

threshold = []
alphas = []

for i in xfrange(0, 25, 0.1):
    alpha = ("{0:.2f}".format(round((i),2)))
    css_t = ave_css + (i * std_css)
    css_t = round(css_t,3)
    alphas.append(alpha)
    threshold.append(css_t)
```

Appendix G (Continued)

```
alphas_df = pd.DataFrame(np.column_stack([alphas, threshold]),
                        columns=['alphas','threshold'])

alphas_df = alphas_df.apply(pd.to_numeric)

sns.set_style("darkgrid")

xticks = np.arange(0,26,1).tolist()

yticks_list = np.arange(0,1.1,0.05).tolist()
yticks = [ round(elem, 2) for elem in yticks_list ]

g = sns.lmplot(x='alphas', # Horizontal axis
               y='threshold',   # Vertical axis
               data=alphas_df,  # Data source
               fit_reg=True,    # Regression line
               size=10)
               #aspect=2)        # Size and dimension

g = (g.set_axis_labels("Alpha values", "Threshold").set(xlim=(0, 25),
                       ylim=(0, 1),
                       xticks=xticks,
                       yticks=yticks))

plt.title("Alpha values versus threshold")

#plt.savefig(working + path + 'img/alpha_threshold_plot',  bbox_inches='tight',
    ↪ dpi=300)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename + '_' + 'alpha_threshold_plot.png', facecolor='k', bbox_inches='
    ↪ tight')

# ----------------------------------------------------------------------------
# create a datafram to compare css counts over different threshold values
# ----------------------------------------------------------------------------
# Opportunity to use loop function here to generate table
# ----------------------------------------------------------------------------
df_css_0 = df_css
df_css_count_0_threshold = df_css_0.groupby(
        'patent_id')['css_patent_id'].nunique()

t = ave_css + (0.75*std_css)
df_css_75 = df_css[df_css['css'] >= t] #0.75 alpha
df_css_count_75_threshold = df_css_75.groupby(
        'patent_id')['css_patent_id'].nunique()

t = ave_css + (0.85*std_css)
df_css_85 = df_css[df_css['css'] >= t] #0.85 alpha
df_css_count_85_threshold = df_css_85.groupby(
        'patent_id')['css_patent_id'].nunique()

t = ave_css + (0.95*std_css)
df_css_95 = df_css[df_css['css'] >= t] #0.95 alpha
df_css_count_95_threshold = df_css_95.groupby(
        'patent_id')['css_patent_id'].nunique()

t = ave_css + (0.99*std_css)
df_css_99 = df_css[df_css['css'] >= t] #0.99 alpha
df_css_count_99_threshold = df_css_99.groupby(
        'patent_id')['css_patent_id'].nunique()

t = ave_css + (2*std_css)
```

Appendix G (Continued)

```
df_css_2 = df_css[df_css['css'] >= t] #2 alpha
df_css_count_2_threshold = df_css_2.groupby(
        'patent_id')['css_patent_id'].nunique()

t = ave_css + (3*std_css)
df_css_3 = df_css[df_css['css'] >= t] #3 alpha
df_css_count_3_threshold = df_css_3.groupby(
        'patent_id')['css_patent_id'].nunique()

t = ave_css + (4*std_css)
df_css_4 = df_css[df_css['css'] >= t] #4 alpha
df_css_count_4_threshold = df_css_4.groupby(
        'patent_id')['css_patent_id'].nunique()

t = ave_css + (5*std_css)
df_css_5 = df_css[df_css['css'] >= t] #5 alpha
df_css_count_5_threshold = df_css_5.groupby(
        'patent_id')['css_patent_id'].nunique()

t = ave_css + (6*std_css)
df_css_6 = df_css[df_css['css'] >= t] #6 alpha
df_css_count_6_threshold = df_css_6.groupby(
        'patent_id')['css_patent_id'].nunique()

#==============================================================================
#  Minimum CSS Threshold Alpha Table
#==============================================================================
'''https://stackoverflow.com/questions/21374375/semantic-relatedness-algorithms-
    ↪ python/21377351#21377351'''
'''https://stackoverflow.com/questions/21374375/semantic-relatedness-algorithms-
    ↪ python'''
# ----------------------------------------------------------------------------

# these are all css to each other however it has not been filtered by foward css
    ↪  dates
df_css_counts = pd.DataFrame({'alpha.0':df_css_count_0_threshold,
                              'alpha.75':df_css_count_75_threshold,
                              'alpha.85':df_css_count_85_threshold,
                              'alpha.95':df_css_count_95_threshold,
                              'alpha.99':df_css_count_99_threshold,
                              'alpha.2X':df_css_count_2_threshold,
                              'alpha.3X':df_css_count_3_threshold,
                              'alpha.4X':df_css_count_4_threshold,
                              'alpha.5X':df_css_count_5_threshold,
                              'alpha.6X':df_css_count_6_threshold})

df_css_counts = df_css_counts.fillna(0) # fill NaNs with zero
df_css_counts['patent_id'] = df_css_counts.index
df_css_counts.to_csv(working + path + 'df_css_counts.csv', index=False)
df_css_counts.info(verbose=True)

#==============================================================================
#  CSS Cited Application Dates
#==============================================================================
# note cite_pat_year is the application filing year
parser = lambda x : pd.to_datetime(x, format='%Y-%m-%d %H:%M:%S',
                                   errors='coerce')

applications = pd.read_csv(working + path + 'application' + '.tsv', sep='\t',
                           header=0,
                           usecols = ['patent_id','date'],
                           dtype = {'patent_id':object},
```

Appendix G (Continued)

```
                           na_values = ['no info', '.'],
                           parse_dates = ['date'],
                           date_parser=parser,
                           encoding = "iso-8859-1")

application_dates = applications[applications['patent_id'].isin(
        input_data7['patent_id'].tolist())]

application_dates = application_dates.rename(columns={'date':'app_date'})
application_dates['app_year'] = application_dates['app_date'].dt.year
application_dates.to_csv(working + path + 'application_dates.csv', index=False)
application_dates.info(verbose=True)
#------------------------------------------------------------------------------
application_dates = pd.read_csv(working + path + 'application_dates.csv',
                                header=0,
                                dtype = {'patent_id':object,
                                         'css_patent_id':object,
                                         'css':int})

# =============================================================================
# Creation of final table
# ==============================================================================
input_data7 = input_data7[['patent_id',
                           'title',
                           'abstract',
                           'claims',
                           'combined',
                           'grant_date',
                           'grant_date_year',
                           'organization',
                           'org_number',
                           'section_id',
                           'class_number']]

input_data7.to_csv(working + path + 'input_data7.csv', index=False)
#==============================================================================
```

Appendix H

PATENT DATA BUBBLE CITATION VISUALIZATION

Listing H.1: Patent data bubble citation visualization python code

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#===============================================================================
'''
title               :8_20190310_patent_data_bubble_cit3.py
description         :Patent Analysis Machine Learning
author              :Salvatore Immordino
date created        :Sun Dec 09 09:20:00 2018
date last modified  :Fri Mar 22 19:12:17 2019
version             :0.1
python_version      :3.7.1
'''
#===============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions.

Data files were downloaded from the United States Patent Office PatensView Data Download located at www.patentsview.org.

Files used: input_data6.csv, df_css.csv (cosine similarity scores from step 7), labels_cpc.csv, labels_org.csv, application_dates.csv and grant_dates.csv

- Patent application time data and patent grant time data is merged with citing patents and saved as 'input_data8.csv'. Line 150 on-wards uses seaborn and matplotlib to create the following charts:
    1) Change in average cosine similarity score  different minimum citation counts
    2) Histogram of mean cosine similarity scores for cited patents within the building industry
    3) Probability distribution function of the mean cosine similarity scores
    4) Scatter plot of U.S. building materials companies intra-citing patents"""
#===============================================================================
# IMPORT STATEMENTS
#===============================================================================

# basic libraries needed to run the tool
import pandas as pd
import numpy as np

# graphing functions
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib import rcParams

# drop the warnings
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

# time stamping
```

Appendix H (Continued)

```
import time

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'
filename = '8_20190310_patent_data_bubble_cit3'

# ==============================================================================
#  Data importation
# ==============================================================================
input_data8 = pd.read_csv(working + path + 'input_data6.csv', header=0,
                          dtype = {'patent_id':object,
                                   'citing_patent':object},
                          na_values = ['no info', '.'],
                          converters={'organization': str},
                          encoding = "iso-8859-1")

df_css = pd.read_csv(working + path + 'df_css.csv', header=0,
                     dtype = {'patent_id':object, 'css_patent_id':object})

labels_cpc = pd.read_csv(working + path + 'labels_cpc.csv', header=0)
labels_org = pd.read_csv(working + path + 'labels_org.csv', header=0)

application_dates = pd.read_csv(working + path + 'application_dates.csv',
                                header=0,
                                dtype = {'patent_id':object},
                                parse_dates = ['app_date'])

grant_dates = pd.read_csv(working + path + 'grant_dates.csv', header=0,
                          dtype = {'patent_id':object},
                          parse_dates = ['grant_date'])

# ------------------------------------------------------------------------------
# Merge patent grant time data with patents
# ------------------------------------------------------------------------------
input_data8 = pd.merge(left=input_data8,
                             right=grant_dates,
                             how='left',
                             left_on=['patent_id'],
                             right_on=['patent_id',],
                             sort=True,
                             left_index=True)

input_data8 = input_data8.rename(columns = {'grant_date_year':'grant_year'})

# ------------------------------------------------------------------------------
# Merge patent application time data with patents
# ------------------------------------------------------------------------------
input_data8 = pd.merge(left=input_data8,
                             right=application_dates,
                             how='left',
                             left_on=['patent_id'],
                             right_on=['patent_id'],
                             sort=True,
                             left_index=True)

input_data8 = input_data8.rename(columns = {'app_year':'pat_app_year'})
```

Appendix H (Continued)

```
input_data8 = input_data8.rename(columns = {'app_date':'pat_app_date'})

# ----------------------------------------------------------------------------
# Merge patent application time data with citing patents
# ----------------------------------------------------------------------------
application_dates = application_dates.rename(columns = \
                                             {'patent_id':'citing_patent'})

input_data8 = pd.merge(left=input_data8,
                             right=application_dates,
                             how='left',
                             left_on=['citing_patent'],
                             right_on=['citing_patent'],
                             sort=True,
                             left_index=True)

# ----------------------------------------------------------------------------
# Merge css data onto patent & citing patent pairs
# ----------------------------------------------------------------------------
df_css = df_css.rename(columns = {'css_patent_id':'citing_patent'})

input_data8 = pd.merge(left=input_data8,
                             right=df_css,
                             how='left',
                             left_on=['patent_id','citing_patent'],
                             right_on=['patent_id', 'citing_patent'],
                             sort=True,
                             left_index=True)

# ----------------------------------------------------------------------------
# Sort, rename,and save clean citation data table
# ----------------------------------------------------------------------------
input_data8.info(verbose=True) # tell me all the things
input_data8 = input_data8.rename(columns = {'cit_pat_year':'cit_app_year'})
input_data8 = input_data8.rename(columns = {'app_date':'cit_app_date'})
input_data8 = input_data8.drop('app_year', 1) # redundant

input_data8 = input_data8[['patent_id',
                           'title',
                           'abstract',
                           'grant_year',
                           'grant_date',
                           'pat_app_year',
                           'pat_app_date',
                           'organization',
                           'org_number',
                           'section_id',
                           'class_number',
                           'citing_patent',
                           'cit_app_year',
                           'cit_app_date',
                           'css']]

#input_data8.to_csv(working + path + 'input_data8.csv', index=False)

#============================================================================
# Load previously saved table
#============================================================================
'''
input_data8 = pd.read_csv(working + path + 'input_data8.csv', header=0,
                          dtype = {'patent_id':object,
                                   'citing_patent':object},
                          parse_dates = ['grant_date','cit_app_date'],
                          na_values = ['no info', '.'],
                          encoding = "iso-8859-1")

input_data8.info(verbose=True) # tell me all the things
'''
#==============================================================================
#  Create citation dataframe
#==============================================================================
X0 = input_data8.groupby('patent_id')['pat_app_date'].unique()
X1 = input_data8.groupby('patent_id')['cit_app_year'].unique().apply(list)
X2 = input_data8.groupby('patent_id')['organization'].unique().apply(list)
X3 = input_data8.groupby('patent_id')['section_id'].unique()
X4 = input_data8.groupby('patent_id')['citing_patent'].nunique() # count cited
X5 = input_data8.groupby('patent_id')['grant_year'].unique()
X6 = input_data8.groupby('patent_id')['org_number'].unique().str[0]
X7 = input_data8.groupby('patent_id')['class_number'].unique().str[0]
X8 = input_data8.groupby('patent_id')['css'].mean() # mean citation css
X9 = input_data8.groupby('patent_id')['grant_date'].unique()
X10 = input_data8.groupby('patent_id')['css'].std()
# ----------------------------------------------------------------------------
# marginal mean css versus patent citation counts versus average css
# ----------------------------------------------------------------------------
import seaborn as sns
from scipy import stats

stats.describe(X4) # count of cited patents per patent
stats.describe(X8[~np.isnan(X8)]) # average css for cited patents

X8.std()

np.mean(X8)
np.mean(X8[~np.isnan(X8)])
np.std(X8[~np.isnan(X8)])
np.std(X8[np.isnan(X8)])
np.min(X8[~np.isnan(X8)])
np.max(X8[~np.isnan(X8)])

df_counts_css = pd.DataFrame({'citation_count':X4,'mean_css':X8,'std_css':X10})

# Don't drop. Just take rows where css is finite
df_counts_css = df_counts_css[np.isfinite(df_counts_css['mean_css'])]

# Don't drop. Just take css for cited counts looking at just 10
df_counts_css_gt = df_counts_css[df_counts_css['citation_count'] >= 1]
df_counts_css_gt.info(verbose=True) # tell me all the things
df_counts_css_gt['mean_css'].mean() # 0.20 vs 0.26

# ----------------------------------------------------------------------------
# calculate the average css for different minimum required citation counts
# ----------------------------------------------------------------------------
def xfrange(start, stop, step):
    i = 0
    while start + i * step < stop:
        yield start + i * step
        i += 1

average_css = []
citation_count = []
size = []
std_css = []

```

```
for i in xfrange(1, 85, 1):
    df_counts_css_gt = df_counts_css[df_counts_css['citation_count'] >= [i]]
    average_css.append(df_counts_css_gt['mean_css'].mean())
    std_css.append(df_counts_css_gt['std_css'].std())
    citation_count.append(i)
    count = len(df_counts_css_gt.index)
    size.append(count)

citation_count_df = pd.DataFrame(np.column_stack([citation_count, average_css,
    ↪ size, std_css]),
                        columns=['citation_count','average_css', 'size', '
    ↪ std_css'])

citation_count_df = citation_count_df.apply(pd.to_numeric)

(citation_count_df['average_css'] * citation_count_df['citation_count']).sum()
    ↪ / citation_count_df['citation_count'].sum()

# -------------------------------------------
# plot
# -------------------------------------------

plt.rcParams['figure.figsize']=(10,10)
sns.set_style("darkgrid")

cmap = sns.diverging_palette(240, 10, l=65, center="dark", as_cmap=True)

g = plt.scatter(citation_count_df["citation_count"],
                citation_count_df["average_css"],
                c=citation_count_df["size"],
                cmap = 'jet',
                s=30)

cbar = plt.colorbar(g, pad=0.025)
cbar.set_label('# CSS Cited Pairs', rotation=270, labelpad=15)

cbar_labels = np.arange(0, int(cbar.vmax) + 198, 100)
loc = cbar_labels + .5
cbar.set_ticks(loc)
cbar.set_ticklabels(cbar_labels)

#sns.scatterplot(x ='citation_count',
#                    y ='average_css',   # vertical axis
#                    data = citation_count_df,  # data source
#                    #size = "size",
#                    #sizes = (10, 250),
#                    palette= 'jet',
#                    hue = "size",
#                    legend = False)

g = sns.regplot("citation_count",
                "average_css",
                data = citation_count_df,
                scatter=False,
                color=".1")

xticks = np.arange(0,95,5).tolist()
yticks = np.arange(0.15,0.35, 0.01).tolist()

g.set(xlim=(0,90), xticks = xticks)
g.set(ylim=(0.15,0.30), yticks = yticks)
```

Appendix H (Continued)

```
plt.title("Average CSS per Minimum Citation Count", fontsize=14)
plt.xlabel("Minimum Citation Count", fontsize=14) #Adding axis labels
plt.ylabel("Average CSS", fontsize=14)

plt.savefig(working + path + (
        'img/' + time.strftime("%Y%m%d-%H%M%S")+ '_') + filename + '
    ↪ _mean_css_by_cit_count',
        bbox_inches='tight',dpi=300)

plt.show()
plt.clf()


# ----------------------------------------------------------------------------
# histogram and KDE of mean css of all patent/citing patent pairs
# ----------------------------------------------------------------------------
'''https://realpython.com/python-histograms/'''
# A kernel density estimation (KDE) is a way to estimate the probability
# density function (PDF) of the random variable that underlies our sample.
# KDE is a means of data smoothing.
fig = plt.figure(figsize=(18,6))
plt.title('Mean CSS for cited patents within building industry')
sns.set_style('darkgrid')
d = X8[~np.isnan(X8)]
sns.distplot(d)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S")+ '_') +
    ↪ filename + '_mean_css_for_cited_patents_within_building_industry_kde',
    ↪ bbox_inches='tight',dpi=300)
plt.show()

fig = plt.figure(figsize=(18,6))
plt.title('Mean CSS for cited patents within building industry')
sns.set_style('darkgrid')
d = X8[~np.isnan(X8)]
sns.distplot(d, fit=stats.laplace, kde=False)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S")+ '_') +
    ↪ filename + '_mean_css_for_cited_patents_within_building_industry_laplace'
    ↪ ,bbox_inches='tight',dpi=300)
plt.show()
# ----------------------------------------------------------------------------
# sns joint plots
# ----------------------------------------------------------------------------
x = df_counts_css['citation_count']
y = df_counts_css['mean_css']

sns.set() # default settings

# figure size in inches
g = sns.jointplot(x=x, y=y, kind='scatter', height=15)
g = sns.set(font_scale=2.5)
#g = g.annotate(stats.pearsonr, fontsize=18)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename +  '_cit_counts_mean_css_scatter_scatter', bbox_inches='tight',
    ↪ dpi=300)
sns.jointplot(x=x, y=y, kind='hex', height=15, xlim=(0, 90))
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename +  '_cit_counts_mean_css_plot_hex', bbox_inches='tight',dpi=300)
    ↪
sns.jointplot(x=x, y=y, kind='kde', height=15, xlim=(0, 90), ylim=(0, 1.2))
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename +  '_cit_counts_mean_css_kde', bbox_inches='tight',dpi=300)
```

```
sns.jointplot(x=x, y=y, kind='scatter', s=200, color='m', edgecolor="skyblue",
    ↪ linewidth=2, height=15)
#plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ 'cit_counts_mean_css_plot_edge', bbox_inches='tight',dpi=300)


sns.set(style="white", color_codes=True)
sns.set(font_scale=2.5)
sns.jointplot(x=x, y=y, kind='kde', color="skyblue", height=15, xlim=(0, 40),
    ↪ ylim=(0, None))
#sns.jointplot(x=x, y=y, kind='kde', color="skyblue", height=15, xlim=(0, 100),
    ↪ ylim=(0, None))
#plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ 'cit_counts_mean_css_plot', bbox_inches='tight',dpi=300)

# -----------------------------------------------------------------------------
dist = stats.norm()

x = np.linspace(start=stats.norm.ppf(0.01),
                stop=stats.norm.ppf(0.99), num=250)

gkde = stats.gaussian_kde(dataset=y)

# `gkde.evaluate()` estimates the PDF itself.
fig, ax = plt.subplots(figsize=(18, 10))
ax.plot(x, dist.pdf(x), linestyle='solid', c='red', lw=3,
        alpha=0.8, label='Analytical (True) PDF')
ax.plot(x, gkde.evaluate(x), linestyle='dashed', c='black', lw=2,
        label='PDF Estimated via KDE')
ax.legend(loc='best', frameon=False)
ax.set_title('Analytical vs. Estimated PDF')
ax.set_ylabel('Probability')
ax.text(-2., 0.35, r'$f(x) = \frac{\exp(-x^2/2)}{\sqrt{2*\pi}}$',
        fontsize=12)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename + '_probability_density_function_mean_css_plot', bbox_inches='
    ↪ tight',dpi=300)

# -----------------------------------------------------------------------------

df_citation_data = pd.DataFrame({'cit_app_year':X1,
                                 'organization':X2,
                                 'section_id':X3,
                                 'citation_count':X4,
                                 'grant_year':X5,
                                 'org_number':X6,
                                 'class_number':X7,
                                 'mean_css':X8})

#df_citation_data = df_citation_data.fillna(0) # fill NaNs with zero
#df_citation_data.to_csv(working + path + (time.strftime("%Y%m%d-%H%M%S") + '_')
    ↪  + 'df_cit_data.csv', index=False)
df_citation_data.to_csv(working + path + 'df_cit_data.csv', index=False)
#==============================================================================
#  Create the citation analysis visualization
#==============================================================================

import datetime
import matplotlib.dates as mdates

warnings.filterwarnings(
    action='ignore', module='matplotlib.figure', category=UserWarning,
    message=('This figure includes Axes that are not compatible with
    ↪ tight_layout, '
             'so results might be incorrect.'))

years = mdates.YearLocator()   # every year
months = mdates.MonthLocator()  # every month
yearsFmt = mdates.DateFormatter('%Y')
mpl.style.use('classic')

# define the data
s = X4.astype(float).values**1.35 # citing patent
#x = X5.astype(float).values**1.0 # grant date year
x = X0 # X9 is grant date; X0 is application date
y = X6.astype(float).values**1.0 # organization number
z = X7.astype(float).values**1.0 # CPC number

rcParams.update({'figure.autolayout': True})
#plt.rcParams['axes.facecolor'] = 'white'

# setup the plot
fig, ax = plt.subplots(1,1, figsize=(15,10))

# setup limits
# plt.xlim([datetime.date(1975, 1, 1), datetime.date(2017, 1, 1)])
plt.ylim([-1,21])

# define the colormap
cmap = plt.cm.jet
cmap.set_under('gray')

# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]

# force the first color entry to be grey
cmaplist[0] = (.5,.5,.5,1.0)

# create the new map
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)

# define the bins and normalize
bounds = np.linspace(0,10,11)
norm = mpl.colors.BoundaryNorm(bounds, cmap.N)
loc = bounds + .5

# make the scatter
scat = ax.scatter(x, y, c=z, s=s, cmap=cmap, norm=norm, lw = 0)

# create a second axes for the colorbar
ax2 = fig.add_axes([1.02, 0.04, 0.04, 0.92])
cb = mpl.colorbar.ColorbarBase(ax2,
                               cmap=cmap,
                               norm=norm,
                               spacing='proportional',
                               ticks=bounds,
                               boundaries=bounds,
                               format='%1i')
cb.set_ticks(loc)
ax.set_title('U.S. Building Materials Companies Intra-Citing Patents', size=18)

# major & minor ticks
y_ticks_major = np.arange(0, 21, 1)
ax.set_yticks(y_ticks_major)
```

```

# format the ticks
#ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(yearsFmt)
ax.xaxis.set_minor_locator(years)

datemin = datetime.date(1970, 1, 1)
datemax = datetime.date(2017, 1, 1)
ax.set_xlim(datemin, datemax)

# and a corresponding grid
#ax.grid(which='both', color='white')

# or if you want differnet settings for the grids:
ax.grid(which='minor', alpha=0.5)
ax.grid(which='major', alpha=0.6)

# set tick labels
ax.set_yticklabels(labels_org['organization'])

ax2.yaxis.set_label_coords(-0.40, 0.50)
ax2.set_ylabel('Cooperative Patent Classification (CPC) [-]',
               size=12,
               labelpad=-20)

ax2.set_yticklabels(labels_cpc['definition'])
ax2.tick_params(axis=u'both', which=u'both',length=0)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename + '_visualization_intra_cit.png', bbox_inches='tight',dpi=300)

# beautify the x-labels
plt.gcf().autofmt_xdate()
plt.show()
```

Appendix I

PATENT DATA BUBBLE CSS VISUALIZATION

Listing I.1: Patent data bubble CSS visualization python code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title                :9_20190310_patent_data_bubble_css.py
description          :Patent Analysis Machine Learning
author               :Salvatore Immordino
date created         :Sun Dec 09 09:20:00 2018
date last modified   :Fri Feb 22 19:12:17 2019
version              :0.1
python_version       :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions. Creates a scatter plot of U.
    ↪ S. building materials companies intra-css of patents. Data files were
    ↪ downloaded from the United States Patent Office PatensView Data Download
    ↪  located at www.patentsview.org."""
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
# basic libraries needed to run the tool
import pandas as pd
import numpy as np

# graphing functions
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib import rcParams

# drop the warnings
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

# time stamping
import time

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'
filename = '9_20190310_patent_data_bubble_css'
# =============================================================================
#  Data importation
# =============================================================================
```

Appendix I (Continued)

```
input_data9 = pd.read_csv(working + path + 'input_data8.csv', header=0,
                          usecols = ['patent_id',
                                     'title',
                                     'abstract',
                                     'grant_year',
                                     'grant_date',
                                     'pat_app_year',
                                     'pat_app_date',
                                     'organization',
                                     'org_number',
                                     'section_id',
                                     'class_number'],
                          dtype = {'patent_id':object},
                          parse_dates = ['grant_date','pat_app_date'],
                          na_values = ['no info', '.'],
                          converters={'organization': str},
                          encoding = "iso-8859-1")

input_data9.info(verbose=True) # tell me all the things

labels_cpc = pd.read_csv(working + path + 'labels_cpc.csv', header=0)
labels_org = pd.read_csv(working + path + 'labels_org.csv', header=0)

df_css = pd.read_csv(working + path + 'df_css.csv', header=0,
                     dtype = {'patent_id':object, 'css_patent_id':object})

application_dates = pd.read_csv(working + path + 'application_dates.csv',
                                header=0,
                                dtype = {'patent_id':object},
                                parse_dates = ['app_date'])
# ------------------------------------------------------------------------------
# Merge css data onto patent data
# ------------------------------------------------------------------------------
ave_css = np.mean(df_css['css']) # average(cos_similarity_matrix)
std_css = np.std(df_css['css'])  # standard_deviation(cos_similarity_matrix)
alpha = 6 # alpha parameter

# CSS minimum threshold
t = (ave_css + (alpha * std_css)) #

intra_css = df_css[df_css['css'] >= t] # select only intra_css > t

input_data9 = pd.merge(left=input_data9,
                       right=intra_css,
                       how='left',
                       left_on=['patent_id'],
                       right_on=['patent_id'],
                       sort=True,
                       left_index=True)

input_data9.info(verbose=True) # tell me all the things
input_data9.head(20)

# ------------------------------------------------------------------------------
# Merge patent application time data with citing patents
# ------------------------------------------------------------------------------
application_dates = application_dates.rename(columns = \
                                             {'patent_id':'css_patent_id'})

input_data9 = pd.merge(left=input_data9,
                       right=application_dates,
                       how='left',
```

Appendix I (Continued)

```
                        left_on=['css_patent_id'],
                        right_on=['css_patent_id'],
                        sort=True,
                        left_index=True)

# sort by patent_id
input_data9 = input_data9.sort_values(by=['patent_id'])

# css patent app_date must be after orginal patent_id grant date
#input_data9x = input_data9[input_data9['app_date'] >= input_data9['grant_date
    ↪ ']]

# same as above except switching to filing date
input_data9x = input_data9[input_data9['app_date'] >= input_data9['pat_app_date'
    ↪ ]]
input_data9x = input_data9x[['patent_id','css_patent_id','css','app_date','
    ↪ app_year']]

# kind of a hack but we lose patent_id's so need to merge it back
input_data9 = input_data9.drop(['css_patent_id','css','app_date','app_year'],
    ↪ axis=1).drop_duplicates()

input_data9 = pd.merge(left=input_data9,
                        right=input_data9x,
                        how='left',
                        left_on=['patent_id'],
                        right_on=['patent_id'],
                        sort=True,
                        left_index=True)

# ------------------------------------------------------------------------------
# Sort, rename,and save clean citation data table
# ------------------------------------------------------------------------------
input_data9 = input_data9.rename(columns = {'app_date':'css_app_date'})
input_data9 = input_data9.rename(columns = {'app_year':'css_app_year'})
input_data9.to_csv(working + path + 'input_data9.csv', index=False)
input_data9.to_csv(working + path + (time.strftime("%Y%m%d-%H%M%S") + '_') + '
    ↪ input_data9.csv', index=False)
input_data9.info(verbose=True) # tell me all the things

#===============================================================================
# Load previously saved table
#===============================================================================
#input_data9 = pd.read_csv(working + path + 'input_data9.csv', header=0,
#                           dtype = {'patent_id':object,
#                                    'css_patent_id':object},
#                           parse_dates = ['grant_date','css_app_date'],
#                           na_values = ['no info', '.'],
#                           encoding = "iso-8859-1")
#
#input_data9.info(verbose=True) # tell me all the things
#===============================================================================
#  Create citation dataframe
#===============================================================================
Y0 = input_data9.groupby('patent_id')['pat_app_date'].unique()
Y1 = input_data9.groupby('patent_id')['css_app_year'].unique().apply(list)
Y2 = input_data9.groupby('patent_id')['organization'].unique().apply(list)
Y3 = input_data9.groupby('patent_id')['section_id'].unique()
Y4 = input_data9.groupby('patent_id')['css_patent_id'].nunique() # count css
Y5 = input_data9.groupby('patent_id')['grant_year'].unique()
Y6 = input_data9.groupby('patent_id')['org_number'].unique().str[0]
Y7 = input_data9.groupby('patent_id')['class_number'].unique().str[0]
```

```
Y8 = input_data9.groupby('patent_id')['css'].mean() # mean citation css
Y9 = input_data9.groupby('patent_id')['grant_date'].unique()
#================================================================================
#  Create CSS dataframe
#================================================================================
df_css_data = pd.DataFrame({'css_app_year':Y1,
                            'organization':Y2,
                            'section_id':Y3,
                            'css_count':Y4,
                            'grant_year':Y5,
                            'org_number':Y6,
                            'class_number':Y7,
                            'mean_css':Y8})

#df_citation_data = df_citation_data.fillna(0) # fill NaNs with zero
#df_css_data.to_csv(working + path + (time.strftime("%Y%m%d-%H%M%S")+ '_') + '
    ↪ df_css_data.csv', index=False)
df_css_data.to_csv(working + path + 'df_css_data.csv', index=False)
#================================================================================
#  Create the citation analysis visualization
#================================================================================
import datetime
import matplotlib.dates as mdates

warnings.filterwarnings(
    action='ignore', module='matplotlib.figure', category=UserWarning,
    message=('This figure includes Axes that are not compatible with
    ↪ tight_layout, '
             'so results might be incorrect.'))

mpl.style.use('classic')

years = mdates.YearLocator()   # every year
months = mdates.MonthLocator()  # every month
yearsFmt = mdates.DateFormatter('%Y')

# define the data
s = Y4.astype(float).values**1.2 # citing patent
x = Y0 # X9 is grant date; X0 is application date
y = Y6.astype(float).values**1.0 # organization number
z = Y7.astype(float).values**1.0 # CPC number

rcParams.update({'figure.autolayout': True})
#plt.rcParams['axes.facecolor'] = 'white'

# setup the plot
fig, ax = plt.subplots(1,1, figsize=(15,10))

# setup limits
plt.ylim([-1,21])

# define the colormap
cmap = plt.cm.jet
cmap.set_under('gray')

# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]

# force the first color entry to be grey
cmaplist[0] = (.5,.5,.5,1.0)

# create the new map
```

Appendix I (Continued)

```
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)

# define the bins and normalize
bounds = np.linspace(0,10,11)
norm = mpl.colors.BoundaryNorm(bounds, cmap.N)
loc = bounds + .5

# make the scatter
scat = ax.scatter(x, y, c=z, s=s, cmap=cmap, norm=norm, lw = 0)

# create a second axes for the colorbar
ax2 = fig.add_axes([1.02, 0.04, 0.04, 0.92])
cb = mpl.colorbar.ColorbarBase(ax2,
                               cmap=cmap,
                               norm=norm,
                               spacing='proportional',
                               ticks=bounds,
                               boundaries=bounds,
                               format='%1i')
cb.set_ticks(loc)
ax.set_title('U.S. Building Materials Companies Intra-CSS Patents', size=18)

# major & minor ticks
y_ticks_major = np.arange(0, 21, 1)
ax.set_yticks(y_ticks_major)

# format the ticks
#ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(yearsFmt)
ax.xaxis.set_minor_locator(years)

datemin = datetime.date(1970, 1, 1)
datemax = datetime.date(2017, 1, 1)
ax.set_xlim(datemin, datemax)

# and a corresponding grid
#ax.grid(which='both', color='white')

# or if you want differnet settings for the grids:
ax.grid(which='minor', alpha=0.5)
ax.grid(which='major', alpha=0.6)

# set tick labels
ax.set_yticklabels(labels_org['organization'])

ax2.yaxis.set_label_coords(-0.40, 0.50)
ax2.set_ylabel('Cooperative Patent Classification (CPC) [-]',
               size=12,
               labelpad=-20)

ax2.set_yticklabels(labels_cpc['definition'])
ax2.tick_params(axis=u'both', which=u'both',length=0)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') +
    ↪ filename + '_visualization_intra_css.png',
            bbox_inches='tight', dpi=300)

# beautify the x-labels
plt.gcf().autofmt_xdate()
plt.show()
```

Appendix J

PATENT DATA INTERSECTION & KNOWLEDGE FLOW

Listing J.1: Patent data citation CSS intersection python code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title                  :10_20190110_patent_data_bubble_css.py
description            :Patent Analysis Machine Learning
author                 :Salvatore Immordino
date created           :Wed Jan 02 17:37:51 2019
date last modified     :Wed Feb 20 19:12:17 2019
version                :0.1
python_version         :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions.

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.

This file uses input_data8.csv, input_data9.csv and application_dates.csv mainly
    ↪  as well as the '3935021_Google_Sim_Docs.csv' and aims to track
    ↪ similarity between patent number 3935021 (info acquired via google
    ↪ search). After some cleaning and restructuring, it merges a dataframe
    ↪ containing patent_id, css_patent_id and citing_patent and plots the
    ↪ intersection between CSS, CIT and results from google and proceedes to
    ↪ create a histogram counting the citation over time for selected patents.
    ↪ """
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
# basic libraries needed to run the tool
import pandas as pd
import numpy as np

# graphing functions
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib.ticker as ticker

# drop the warnings
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

# time stamping
import time

# find out your current working directory
import os
print(os.getcwd())
```

Appendix J (Continued)

```
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# needed for this code
import seaborn as sns
from scipy import stats
from scipy.stats import norm

# legend
import matplotlib.patches as mpatches

# set filename and path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'
filename = '10_20190311_patent_cit_css_intersection'

# set palette  orange, green, red, purple, brown, pink, grey, tan, lblue, blue
color_list = sns.set_palette(palette=sns.color_palette("muted"))
muted_colors = palette=sns.color_palette("muted")
muted_list = (muted_colors.as_hex())
print(muted_colors.as_hex())

# ============================================================================
#  Data importation
# ============================================================================
application_dates = pd.read_csv(working + path + 'application_dates.csv',
                                header=0,
                                dtype = {'patent_id':object},
                                parse_dates = ['app_date'])

input_data8 = pd.read_csv(working + path + 'input_data8.csv', header=0,
                          dtype = {'patent_id':object,
                                   'citing_patent':object},
                          parse_dates = ['grant_date','cit_app_date', '
    ↪ pat_app_date'],
                          na_values = ['no info', '.'],
                          encoding = "iso-8859-1")

input_data9 = pd.read_csv(working + path + 'input_data9.csv', header=0,
                          dtype = {'patent_id':object,
                                   'css_patent_id':object},
                          parse_dates = ['grant_date','css_app_date', '
    ↪ pat_app_date'],
                          na_values = ['no info', '.'],
                          encoding = "iso-8859-1")

input_data9.info(verbose=True) # tell me all the things
input_data9.head(10)

# ============================================================================
#  Create Intersection Tables
# ============================================================================
df_intra_css = input_data9[['patent_id','css_patent_id']].copy()
df_intra_css = df_intra_css.rename(columns = {'css_patent_id':'citing_patent'})
df_intra_css = df_intra_css.drop_duplicates()
df_intra_css.info(verbose=True) # tell me all the things
df_intra_css.head(20)

df_intra_cit = input_data8[['patent_id','citing_patent']].copy()
df_intra_cit.info(verbose=True) # tell me all the things
df_intra_cit = df_intra_cit.drop_duplicates()
df_intra_cit.head(20)
```

```
df_intersection = pd.merge(df_intra_cit,
                           df_intra_css,
                           how='inner',
                           on=['patent_id', 'citing_patent'])

df_intersection = df_intersection.drop_duplicates()
df_intersection.head(10)

# remember to reload input tables 8 & 9
Z0 = input_data8.groupby('patent_id')['citing_patent'].nunique().nlargest(12).
    ↪ reset_index() # count cited
Z1 = df_intersection.groupby('patent_id')['citing_patent'].nunique() #
    ↪ intersection counts
Z2 = df_intra_css[df_intra_css['patent_id'] == '3935021'] # list intra css
Z3 = df_intra_cit[df_intra_cit['patent_id'] == '3935021'] # list intra cit

# grap 12 random patents
Z4 = input_data8.groupby('patent_id')['citing_patent'].nunique().sample(12).
    ↪ reset_index() # count cited

# range function that allows non-integer steps
def xfrange(start, stop, step):
    i = 0
    while start + i * step < stop:
        yield start + i * step
        i += 1

## loop function to compare patents by method
new_index = pd.Series(xfrange(0, 26, 1))
#patents = ['3935021', '6673144', '6432267','7585388']
patents = Z0[('patent_id')].tolist()
patent = 0
data = pd.DataFrame([])

for i, patent in enumerate(patents):
    # css related patents
    css_data_cited = input_data9.loc[input_data9['patent_id'] == patent].
    ↪ drop_duplicates()
    css_data_cited = css_data_cited[['patent_id', 'css_patent_id', 'pat_app_year
    ↪ ', 'css_app_year']]
    css_data_cited['css_years_past'] = css_data_cited['css_app_year'] -
    ↪ css_data_cited['pat_app_year']
    css_data_cited = css_data_cited[['patent_id','css_years_past']]

    # cit related patents
    cit_data_cited = input_data8.loc[input_data8['patent_id'] == patent].
    ↪ drop_duplicates()
    cit_data_cited = cit_data_cited[['patent_id', 'citing_patent', 'pat_app_year
    ↪ ', 'cit_app_year']]
    cit_data_cited['cit_years_past'] = cit_data_cited['cit_app_year'] -
    ↪ cit_data_cited['pat_app_year']
    cit_data_cited = cit_data_cited[['patent_id','cit_years_past']]

    # inputs
    css = css_data_cited['css_years_past'].groupby(css_data_cited['
    ↪ css_years_past']).count()
    cit = cit_data_cited['cit_years_past'].groupby(cit_data_cited['
    ↪ cit_years_past']).count()

    combined = (pd.concat([css, cit], axis = 1)).reindex(new_index, fill_value
    ↪ =0).fillna(0)
```

Appendix J (Continued)

```
    combined = combined.rename(columns={combined.columns[0]: patent + '_' + "css
    ↪ ",
                                    combined.columns[1]: patent + '_' + "cit
    ↪ "})

    data = pd.concat([data, combined], axis=1)

    print ("patent {} = {}".format(i, patent))

    if i == len(patents) - 1:
        print ("stacking and rename patent columns")
        data_wide = data
        data = data.stack().reset_index()
        data.columns = ['years_past','related_method','related_count']

# ==============================================================================
# Bar charts, make sure to run all three as they are interdependent
# ==============================================================================
import matplotlib as mpl #set defaults
mpl.rcdefaults()

desired_patent_bar = '6500493'
filename = 'code_10_'

# css related bar chart pick a patent
css_data_cited = input_data9.loc[input_data9['patent_id'] == desired_patent_bar].
    ↪ drop_duplicates()
css_data_cited = css_data_cited[['patent_id','css_app_date']]
plt.figure(figsize=(8,6), dpi=100)
plt.ylim([0,25])
plt.title('US'+ desired_patent_bar + ' Intra-CSS', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('CSS-related Count', fontsize=12)
css_data_cited['css_app_date'].groupby(
        css_data_cited['css_app_date'].dt.year).count().plot(
                kind="bar",
                color='C0')

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              ) + filename + '_' + 'css_related_bar_us' + str(
    ↪ desired_patent_bar) + '.png',
                              bbox_inches='tight', dpi=300)

# citation bar plot, pick a patent
cit_data_cited = input_data8.loc[input_data8['patent_id'] == desired_patent_bar].
    ↪ drop_duplicates()
cit_data_cited = cit_data_cited[['patent_id','cit_app_date']]
plt.figure(figsize=(8,6), dpi=100)
plt.ylim([0,25])
plt.title('US'+ desired_patent_bar + ' Intra-CIT', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Citation Count', fontsize=12)
cit_data_cited['cit_app_date'].groupby(
        cit_data_cited['cit_app_date'].dt.year).count().plot(
                kind="bar",
                color='C1')

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              ) + filename + '_' + 'cit_related_bar_us' + str(
    ↪ desired_patent_bar) + '.png',
                              bbox_inches='tight', dpi=300)
```

Appendix J (Continued)

```
css_data_cited.info(verbose=True) # tell me all the things
css_data_cited.head(10)

# Combined bar

P0 = css_data_cited['css_app_date'].groupby(css_data_cited['css_app_date'].dt.
    ↪ year).count()
P1 = cit_data_cited['cit_app_date'].groupby(cit_data_cited['cit_app_date'].dt.
    ↪ year).count()
P2 = pd.concat([P0, P1], axis=1).reset_index()
P2 = P2.rename(columns={P2.columns[0]: "year" })

P2max = max(P2.year) + 1
P2min = min(P2.year)

def xfrange(start, stop, step):
    i = 0
    while start + i * step < stop:
        yield start + i * step
        i += 1

P3 = pd.DataFrame(xfrange(P2min, P2max, 1))
P3 = P3.rename(columns={P3.columns[0]: "year" })
P4 = pd.merge(P3, P2, how ='left', on=['year']).fillna(0)
P4.set_index('year', inplace=True)

P4.plot.bar(figsize=(8, 6), color=muted_colors, width=0.9)
plt.ylim([0,25])
plt.title(' Method Comparison:' + ' US' + desired_patent_bar, fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Method Count', fontsize=12)
plt.legend(title="Method")
plt.grid(False)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              ) + filename + '_' + 'methods_year_bar_us' + str(
    ↪ desired_patent_bar) + '.png',
                              bbox_inches='tight', dpi=300)

# ===============================================================================
# heat map
# ===============================================================================
fig = plt.figure(figsize=(8,6), dpi=100)
sns.heatmap(data_wide, annot=True)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              ) + filename + '_' + 'heat_map_top_12_cited' + '.
    ↪ png',
                              bbox_inches='tight', dpi=300)

# ===============================================================================
# distribution plot KDE estimation
# ===============================================================================

desired_patent_kde = '4647496'

css_kde = data.loc[data.related_method == str(desired_patent_kde) + "_css"]
cit_kde = data.loc[data.related_method == str(desired_patent_kde) + "_cit"]

xmin = 0  # m1.min()
xmax = 35 # m1.max()
ymin = 0  # m2.min()
ymax = 25 # m2.max()
```

```

# ----------------------------------------------------------------------------
# method 1: distribution plot KDE estimation
# ----------------------------------------------------------------------------
# ----------------------------------
# css kde
# ----------------------------------
plt.figure(figsize=(8,6), dpi=100)
ax = sns.kdeplot(css_kde.years_past,
                 css_kde.related_count,
                 cmap="Blues",
                 shade=True,
                 shade_lowest=False,
                 clip=(0.0, 35.0))

ax.set_xlim([xmin, xmax])
ax.set_ylim([ymin, ymax])

plt.suptitle('Kernel Density Estimation', y=.95, fontsize=14)
plt.title('Method: Intra-CSS' + ' US' + desired_patent_kde, fontsize=12, x
    ↪ =0.47)
plt.xlabel('Years', fontsize=12)
plt.ylabel('Relatedness Method Counts', fontsize=12)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              )  + filename + '_' +  'css_kde_hist_us' + str(
    ↪ desired_patent_kde) + '.png',
                              bbox_inches='tight', dpi=300)
# ----------------------------------
# cit kde
# ----------------------------------
plt.figure(figsize=(8,6), dpi=100)
ax = sns.kdeplot(cit_kde.years_past,
                 cit_kde.related_count,
                 cmap="Oranges",
                 shade=True,
                 shade_lowest=False,
                 clip=(0.0, 35.0))

ax.set_xlim([xmin, xmax])
ax.set_ylim([ymin, ymax])

plt.suptitle('Kernel Density Estimation', y=.95, fontsize=14)
plt.title('Method: Intra-Citation' + ' US' + desired_patent_kde, fontsize=12, x
    ↪ =0.47)
plt.xlabel('Years', fontsize=12)
plt.ylabel('Relatedness Method Counts', fontsize=12)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              )  + filename + '_' +  'cit_kde_hist_us' + str(
    ↪ desired_patent_kde) + '.png',
                              bbox_inches='tight', dpi=300)

# ----------------------------------------------------------------------------
# method 2: distribution plot guassian kde estimation
# ----------------------------------------------------------------------------

# ----------------------------------
# css kde
# ----------------------------------
css1 = css_kde.years_past
css2 = css_kde.related_count
```

Appendix J (Continued)

```

# Perform a kernel density estimate on the data:
X, Y = np.mgrid[xmin:xmax:100j, ymin:ymax:100j]
positions = np.vstack([X.ravel(), Y.ravel()])
values = np.vstack([css1, css2])
kernel = stats.gaussian_kde(values)
Z = np.reshape(kernel(positions).T, X.shape)

# Plot the results:
fig, ax = plt.subplots(figsize=(8,6), dpi=100)
ax.imshow(np.rot90(Z), cmap=plt.cm.gist_earth_r,
          extent=[xmin, xmax, ymin, ymax])
ax.plot(css1, css2, 'k.', markersize=2)
ax.set_xlim([xmin, xmax])
ax.set_ylim([ymin, ymax])
plt.suptitle('Kernel Density Estimation', y=0.95, fontsize=14,)
plt.title('Method: Intra-CSS' + ' US' + desired_patent_kde, fontsize=12, x
    ↪ =0.47)
plt.xlabel('Years', fontsize=12)
plt.ylabel('Relatedness Method Counts', fontsize=12)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              )  + filename + '_' +  'color_css_kde_hist_us' +
    ↪ str(desired_patent_kde) + '.png',
                              bbox_inches='tight', dpi=300)

# ----------------------------------
# cit kde
# ----------------------------------
cit1 = cit_kde.years_past
cit2 = cit_kde.related_count

xmin = 0  # m1.min()
xmax = 35 # m1.max()
ymin = 0  # m2.min()
ymax = 25 # m2.max()

# Perform a kernel density estimate on the data:
X, Y = np.mgrid[xmin:xmax:100j, ymin:ymax:100j]
positions = np.vstack([X.ravel(), Y.ravel()])
values = np.vstack([cit1, cit2])
kernel = stats.gaussian_kde(values)
Z = np.reshape(kernel(positions).T, X.shape)

# Plot the results:
fig, ax = plt.subplots(figsize=(8,6), dpi=100)
ax.imshow(np.rot90(Z), cmap=plt.cm.gist_earth_r,
          extent=[xmin, xmax, ymin, ymax])
ax.plot(cit1, cit2, 'k.', markersize=2)
ax.set_xlim([xmin, xmax])
ax.set_ylim([ymin, ymax])
plt.suptitle('Kernel Density Estimation', y=0.95, fontsize=14)
plt.title('Method: Intra-Citation' + ' US' + desired_patent_kde, fontsize=12, x
    ↪ =0.47)
plt.xlabel('Years', fontsize=12)
plt.ylabel('Relatedness Method Counts', fontsize=12)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              )  + filename + '_' +  'color_cit_kde_hist_us' +
    ↪ str(desired_patent_kde) + '.png',
                              bbox_inches='tight', dpi=300)

```

Appendix J (Continued)

```
# ==============================================================================
# histogram
# ==============================================================================
# ----------------------------------
# css matplotlib histogram method 1
# ----------------------------------
(mu, sigma) = norm.fit(css_kde.related_count.values)

plt.cla() # clear the axis
plt.figure(figsize=(8,6), dpi=100)

# the histogram of the NBERdata
n, bins, patches = plt.hist(css_kde.related_count, 10, normed=1, facecolor=
    ↪ muted_list[0], alpha=0.75)

bincenters = 0.5*(bins[1:]+bins[:-1])
bin_width = bins[1]-bins[0]

# add a 'best fit' line
y = mlab.normpdf(bins, mu, sigma)
l = plt.plot(bins, y, 'r--', linewidth=2)

#plot
plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.xlabel('# CSS-Related')
plt.ylabel('Probability Density')
plt.suptitle('PDF & Histogram: Intra-CSS' + ' US' + desired_patent_kde, y=1.00,
    ↪ fontsize=16)
plt.title(r'$\mathrm{Statistics:}\ \mu=%.0f,\ \sigma=%.0f$' %(mu, sigma),
    ↪ fontsize=14)

mean = plt.axvline(css_kde.related_count.mean(), color='C2', linestyle='dashed',
    ↪  linewidth=2)
median = plt.axvline(css_kde.related_count.median(), color='C5', linestyle='
    ↪ dashed', linewidth=2)

red_line = mpatches.Patch(color='red', label='Normal Distribution')
green_line = mpatches.Patch(color='C2', label='Mean')
brown_line = mpatches.Patch(color='C5', label='Median')

plt.xlim(0, 11)
plt.ylim(0, 0.5)
plt.legend(handles=[red_line, green_line, brown_line])
plt.grid(True)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
            )  + filename + '_' +  'css_pdf_hist_us' + str(desired_patent_kde) +
    ↪  '.png',
            bbox_inches='tight', dpi=300)

# ----------------------------------
# cit matplotlib histogram method 1
#
(mu, sigma) = norm.fit(cit_kde.related_count.values)

plt.cla() # clear the axis
plt.figure(figsize=(8,6), dpi=100)

# the histogram of the NBERdata
n, bins, patches = plt.hist(cit_kde.related_count, 10, normed=1, facecolor='C1',
    ↪  alpha=0.75)
```

```

bincenters = 0.5*(bins[1:]+bins[:-1])
bin_width = bins[1]-bins[0]

# add a 'best fit' line
y = mlab.normpdf(bins, mu, sigma)
l = plt.plot(bins, y, 'r--', linewidth=2)

#plot
plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.xlabel('# CIT-Related')
plt.ylabel('Probability Density')
plt.suptitle('PDF & Histogram: Intra-CIT' + ' US' + desired_patent_kde, y=1.00,
    ↪ fontsize=16)
plt.title(r'$\mathrm{Stats}\ \mu=%.0f,\ \sigma=%.0f$' %(mu, sigma), fontsize=14)

mean = plt.axvline(css_kde.related_count.mean(), color='C2', linestyle='dashed',
    ↪  linewidth=2)
median = plt.axvline(css_kde.related_count.median(), color='C5', linestyle='
    ↪ dashed', linewidth=2)

red_line = mpatches.Patch(color='red', label='Normal Distribution')
green_line = mpatches.Patch(color='C2', label='Mean')
brown_line = mpatches.Patch(color='C5', label='Median')

plt.xlim(0, 11)
plt.ylim(0, 0.5)
plt.legend(handles=[red_line, green_line, brown_line])
plt.grid(True)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
           )  + filename + '_' +  'cit_pdf_hist_us' + str(desired_patent_kde) +
    ↪ '.png',
           bbox_inches='tight', dpi=300)

# ---------------------------------
# css seaborn histogram method 2
# ---------------------------------
#plt.cla() # clear the axis
plt.figure(figsize=(8,6), dpi=100)
sns.distplot(css_kde.related_count, hist=True, kde=True, bins=int(10), color = '
    ↪ C0', hist_kws={'edgecolor':'black'})
plt.title('PDF: Intra-CSS' + ' US' + desired_patent_kde, fontsize=12, x =0.47)
plt.xlabel('# CSS-Related', fontsize=12)
plt.ylabel('Probability', fontsize=12)
plt.ylim(0, 0.5)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
           )  + filename + '_' +  'css_pdf_us' + str(desired_patent_kde) + '.png
    ↪ ',
           bbox_inches='tight', dpi=300)

# ---------------------------------
# cit seaborn histogram method 2
# ---------------------------------
#plt.cla() # clear the axis
plt.figure(figsize=(8,6), dpi=100)
sns.distplot(cit_kde.related_count, hist=True, kde=True, bins=int(10), color = '
    ↪ C1', hist_kws={'edgecolor':'black'})
plt.title('PDF: Intra-Citation' + ' US' + desired_patent_kde, fontsize=12, x
    ↪ =0.47)
plt.xlabel('# Citations', fontsize=12)
```

Appendix J (Continued)

```
plt.ylabel('Probability', fontsize=12)
plt.ylim(0, 0.5)
plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
           )  + filename + '_' +  'cit_pdf_us' + str(desired_patent_kde) + '.png
    ↪ ',
           bbox_inches='tight', dpi=300)

# ==============================================================================
# find peaks
# ==============================================================================
'''
This function takes a one-dimensional array and finds all local maxima
by simple comparison of neighbouring values.
'''

from scipy.signal import find_peaks

fig = plt.figure(figsize=(8,6), dpi=100)

x1 = cit_kde.related_count.reset_index(drop=True).apply(pd.to_numeric)
peaks, _ = find_peaks(x1, height=0)

plt.plot(x1, color = 'C1')
plt.plot(peaks, x1[peaks], "x", color = 'C1')
plt.plot(np.zeros_like(x1), "--", color="gray")

x2 = css_kde.related_count.reset_index(drop=True).apply(pd.to_numeric)
peaks, _ = find_peaks(x2, height=0)

plt.plot(x2, color = 'C0')
plt.plot(peaks, x2[peaks], "x", color = 'C0')
plt.plot(np.zeros_like(x2), "--", color="gray")

plt.xlim([0,35])
plt.ylim([0,25])
plt.title('Find Local Maxima:' + ' Patent US' + desired_patent_kde, fontsize=14,
    ↪  x =0.47)
plt.xlabel('Years', fontsize=12)
plt.ylabel('Relatedness Method Counts', fontsize=12)
L = plt.legend(fontsize=10)
L.get_texts()[0].set_text('Citation Count')
L.get_texts()[1].set_text('Citation Peak')
L.get_texts()[2].set_text('CSS Count')
L.get_texts()[3].set_text('CSS Peak')

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              )  + filename + '_' +  'method_count_peaks__us' +
    ↪ str(desired_patent_kde) + '.png',
                              bbox_inches='tight', dpi=300)

# ==============================================================================
# intra-relatedness over time
# ==============================================================================
# plotting resources used
'''
https://stackoverflow.com/questions/48204780/how-to-plot-multiple-figures-in-a-
    ↪ row-using-seaborn
https://seaborn.pydata.org/generated/seaborn.catplot.html
http://seaborn.pydata.org/tutorial/categorical.html?highlight=bar%20plot
https://stackoverflow.com/questions/33049884/how-to-plot-2-seaborn-lmplots-side-
    ↪ by-side
```

```
https://stackoverflow.com/questions/41659188/how-to-adjust-subplot-size-in-
    ↪ seaborn
https://stackoverflow.com/questions/41329789/populating-seaborn-subplots-using-
    ↪ an-array
'''
# set fig parameters
num_cols = 3
num_rows = 4
num_plots = len(patents)

# set figsize here
fig, axs = plt.subplots(figsize=(13,10),
                        sharey=True,
                        sharex=True,
                        ncols=num_cols,
                        nrows=num_rows)

# iterate through all axes and create a plot
for i, ax in enumerate(axs.flatten()):
    sns.set_context("paper", rc={"font.size":5,
                                 "axes.titlesize":8,
                                 "axes.labelsize":10})
    sns.catplot(x='years_past',
                y='related_count',
                hue='related_method',
                data = data[data.related_method.str.contains(patents[i])],
                kind='bar',
                palette="muted",
                legend=False,
                ax=ax)
    ax.xaxis.set_major_locator(ticker.MultipleLocator(5))
    ax.xaxis.set_major_formatter(ticker.ScalarFormatter())
    ax.xaxis.set_minor_locator(ticker.MultipleLocator())
    ax.yaxis.set_minor_locator(ticker.MultipleLocator())
    plt.close(2) # need to close empty sns plots

plt.tight_layout()

plt.savefig(
        working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                          )  + filename + '_' +  'patent_relationships_over_time
    ↪ ' + '.png',
                          bbox_inches='tight', dpi=300)
plt.show()
```

Appendix K

PATENT DATA DIMENSION REDUCTION

Listing K.1: Patent data dimension reduction

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title                :11_20190311_dimension_reduction.py
description          :Patent Analysis Machine Learning
author               :Salvatore Immordino
date created         :Wed Jan  13 16:00:36 2019
date last modified   :Fri Mar 22 19:12:17 2019
version              :0.1
python_version       :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions.

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.

This file uses input_data6.csv and patent_local_claims.csv and merges with
    ↪ claims.csv and grant_dates.csv. Fixes data and encoding errors, and
    ↪ preprocessing to clean the data. Since tSNE is computationally expensive,
    ↪  a simpler decomposition method, PCA is used first and then performs
    ↪ tSNE on its output.
- The next section performs a tSNE perplexity study at four different levels and
    ↪  plots for each level are saved.
- The next section performs a visualization of t-SNE followed by normal PCA
    ↪ reduction followed by visualization of it, scale adjusted, for
    ↪ comparison."""
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
# basic libraries needed to run the tool
import pandas as pd
import numpy as np

# graphing functionsxxx
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import rcParams

# time stamping
import time

# drop the warnings
import warnings
warnings.filterwarnings("ignore") # supress all warnings

# natural language tools - clean and preprocess
```

Appendix K (Continued)

```
from nltk.corpus import stopwords

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
os.chdir(working)
working = working.replace('\\', '/') # replaced all instances of \ with \\

# needed for this code
#import seaborn as sns
import string

# need to remove multiple letter words like 'aaaaaaaaaaaaaaa'
import re

# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'
filename = '11_20190311_dimension_reduction'

# ============================================================================
#  Data importation
# ============================================================================
# load the patent data files sans textual data
input_data7 = pd.read_csv(working + path + 'input_data7' + '.csv', header=0,
#                           usecols = ['patent_id',
#                                      'title',
#                                      'abstract',
#                                      'grant_date_year',
#                                      'organization',
#                                      'org_number',
#                                      'section_id',
#                                      'class_number'],
                                     dtype = {'patent_id':object})

# drop duplicates (5651)
input_data7 = input_data7.drop_duplicates()
input_data7.info(verbose=True) # tell me all the things

# ============================================================================
# Import full claims text from patentsview
# ============================================================================
patent_local_claims = pd.read_csv(working + path + 'patent_local_claims.csv',
                                  header=0,
                                  dtype = {'patent_id':object,
                                           'sequence':int,
                                           'claims':object})
# ============================================================================
patent_claims_combined = patent_local_claims.groupby(
        ['patent_id'])['claims'].apply(','.join).reset_index()

input_data7 =  pd.merge(left=input_data7,
                        right=patent_claims_combined,
                        how='left',
                        left_on='patent_id',
                        right_on='patent_id',
                        sort=True,
                        left_index=True)

input_data7['combined'] = input_data7[['title','abstract','claims']].apply(
        lambda x: ''.join(x.astype(str)), axis=1)

```

Appendix K (Continued)

```
# ===========================================================================
# Import full grant data from patentsview patents.tsv
# ===========================================================================
grant_dates = pd.read_csv(working + path + 'grant_dates.csv',
                                            header=0,
                                            dtype = {'patent_id':object},
                                            parse_dates = ['grant_date'])
# ===========================================================================
input_data7 = pd.merge(left=input_data7,
                       right=grant_dates,
                       how='left',
                       left_on='patent_id',
                       right_on='patent_id',
                       sort=True,
                       left_index=True)

# ---------------------------------------------------------------------------
#  Fix data and encoding errors
# ---------------------------------------------------------------------------
import unicodedata # to fix encoding errors

input_data7['combined'] = input_data7['combined'].apply(
        lambda val: unicodedata.normalize('NFKD', val).encode(
                'ascii', 'ignore').decode())

# remove encoding replacements for subscript, superscript, and degrees
input_data7['combined'] = input_data7['combined'].str.replace(".sub.", "") #
    ↪ removes subset
input_data7['combined'] = input_data7['combined'].str.replace(".sup.", "") #
    ↪ removes superscript
input_data7['combined'] = input_data7['combined'].str.replace(".degree", "
    ↪ degrees")

## get unique list of patent #'s from patent_text and parse the list
# input_data7.patent_id.nunique() # count number of patents (5651)
# ===========================================================================
#  Split the data to train and test set
# ===========================================================================
#from sklearn.model_selection import train_test_split
#train, test = train_test_split(input_data7, test_size=0.33, random_state=42)

train = input_data7

# ===========================================================================
# Create lists
# ===========================================================================
'''https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.
    ↪ html'''

corpus = train.combined                     # title, abstract, claims list
labels_cpc = pd.read_csv(working + path + 'labels_cpc.csv', header=0)
#my_columns = train['patent_id'].tolist()    # patent numbers
#patent_numbers = '|'.join(my_columns)       # patent number list


# ===========================================================================
#  Create jargon lexical
# ===========================================================================
# Created a list of common patent terms
#patent_terms = pd.read_csv(working + path + 'patent_terms.csv')
jargon = {'according', 'also', 'apparatus', 'assembly', 'body', 'claim',
          'claimed', 'component', 'composition', 'comprise', 'comprises',
```

Appendix K (Continued)

```
          'comprising', 'consisting', 'containing', 'device', 'disclosed',
          'element', 'embodying', 'end', 'face', 'first', 'form', 'formed',
          'forming', 'forms', 'group', 'include', 'includes', 'including',
          'invention', 'layer', 'le', 'least', 'made', 'making', 'material',
          'may', 'mean', 'means', 'member', 'method', 'mixture', 'one',
          'patent', 'plurality', 'portion', 'preferably', 'present',
          'process', 'product', 'provided', 'provides', 'providing', 'relates',
          'resulting', 'said', 'second', 'selected', 'substantially',
          'substrate', 'support', 'surface', 'system', 'technology', 'thereof',
          'third', 'two', 'web', 'weight', 'wherein', 'within', 'wt'}

# -----------------------------------------------------------------------------
#  Stop words, punctuation, lemmatization, and word length steps
# -----------------------------------------------------------------------------
# create stop words
stop_words = set(stopwords.words('english'))

# punctuation
punctuations = set(string.punctuation)
punctuations.remove('-') # remove hyphens
#punctuations.remove('/')

# lemmatization
from nltk.stem.wordnet import WordNetLemmatizer
lemma = WordNetLemmatizer()

# set minimum word length
word_len = 2

# ----------------------------------------------------------------------------
#  Tokenize word function
# -----------------------------------------------------------------------------

def clean(doc):
    number_free = ''.join([c for c in doc if c not in "1234567890"])
    words = [word.strip(string.punctuation) for word in number_free.split(" ")]
    filtered = [f for f in words if f and f.lower() not in stop_words]
    undo = "".join([" "+i if not i.startswith("'") and i not in string.
    ↪ punctuation else i for i in filtered]).strip()
    punc_free = ''.join(ch for ch in undo if ch not in punctuations)
    smallword_free = ' '.join([w for w in punc_free.split() if len(w)>word_len])
    lemmatized = " ".join(lemma.lemmatize(word) for word in smallword_free.split
    ↪ ())
    jargon_free = " " .join([j for j in lemmatized.lower().split() if j not in
    ↪ jargon])
    for i in jargon_free:
        jargon_free = re.sub((i+i+i), ' ', jargon_free)
        #jargon_free = jargon_free.replace('the ',' ') # not need with proper
    ↪ space inserted on merge
    nonsense = ' '.join([w for w in jargon_free.split() if len(w)>1])
    return nonsense

corpus_clean = [clean(doc) for doc in corpus] # list of sentance strings
corpus_tokenize = [clean(doc).split() for doc in corpus]  # list of string words

#==============================================================================
#  Term Frequency * Inverse Document Frequency, Tf-Idf
#==============================================================================
'''https://stackoverflow.com/questions/12118720/python-tf-idf-cosine-to-find-
    ↪ document-similarity?rq=1'''
'''https://radimrehurek.com/gensim/tut2.html'''
'''http://dsgeek.com/2018/02/19/tfidf_vectors.html'''
```

Appendix K (Continued)

```

#-------------------------------------------------------------------------------
# create a dictionary ('conda install -c anaconda gensim')
#-------------------------------------------------------------------------------
from gensim.corpora import Dictionary
from gensim.models.tfidfmodel import TfidfModel
from gensim.matutils import sparse2full

#documents_words = ' '.join(corpus_clean).split()
docs_dict = Dictionary(corpus_tokenize)

#docs_dict.filter_extremes(no_below=20, no_above=0.2)
docs_dict.compactify()

docs_corpus = [docs_dict.doc2bow(doc) for doc in corpus_tokenize]
model_tfidf = TfidfModel(docs_corpus, id2word=docs_dict)
docs_tfidf  = model_tfidf[docs_corpus]
docs_vecs   = np.vstack([sparse2full(c, len(docs_dict)) for c in docs_tfidf])

#===============================================================================
# Dimension reduction
#===============================================================================
'''https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.
    ↪ html'''
'''https://github.com/aviolante/sas-python-work/blob/master/tSneExampleBlogPost.
    ↪ ipynb'''

from sklearn.decomposition import PCA
docs_pca = PCA(n_components=50).fit_transform(docs_vecs)

'''http://www.scikit-yb.org/en/latest/api/text/tsne.html'''
'''https://distill.pub/2016/misread-tsne/'''
'''https://stats.stackexchange.com/questions/263539/clustering-on-the-output-of-
    ↪ t-sne'''

# Visualize 'http://dsgeek.com/2018/02/19/tfidf_vectors.html'
'''https://www.kdnuggets.com/2018/08/introduction-t-sne-python.html'''
# above ^ has some papers to reference...
'''https://www.datacamp.com/community/tutorials/introduction-t-sne'''

#===============================================================================
# t-SNE perplexity study at differing levels
#===============================================================================
#import seaborn as sns
from sklearn import manifold

warnings.filterwarnings(
action='ignore', module='matplotlib.figure', category=UserWarning,
message=('This figure includes Axes that are not compatible with tight_layout, '
         'so results might be incorrect.'))

perplexities = [i for i in np.arange(0,101,1)]
#perplexities = [0,5,10,15,100]
#perplexities = [30]

#===============================================================================
#  Create the t-SNE visualization
#===============================================================================
for i, perplexity in enumerate(perplexities):

    tsne = manifold.TSNE(n_components=2,
                         init='random',
```

Appendix K (Continued)

```
                    random_state=0,
                    perplexity=perplexity)

viz = tsne.fit_transform(docs_pca)
# create new 'data' df using tsne values and orginal data
tsne_corpus_out = pd.DataFrame(viz, columns=['x','y'])
data = tsne_corpus_out.join(train)

a4_dims = (11.7, 8.27)

# define the data
s = 30 # point size
x = data.x
y = data.y
z = data.class_number

rcParams.update({'figure.autolayout': True})
plt.rcParams['axes.facecolor'] = 'white'

# setup the plot
fig, ax = plt.subplots(1,1, figsize=a4_dims)

# define the colormap
cmap = plt.cm.jet
cmap.set_under('gray')

# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]

# force the first color entry to be grey
cmaplist[0] = (.5,.5,.5,1.0)

# create the new map
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)

# define the bins and normalize
bounds = np.linspace(0,10,11)
norm = mpl.colors.BoundaryNorm(bounds, cmap.N)
loc = bounds + .5

# make the scatter
scat = ax.scatter(x,
                  y,
                  c=z,
                  s=s,
                  cmap=cmap,
                  norm=norm,
                  lw = 0,
                  alpha=0.20)

# create a second axes for the colorbar
ax2 = fig.add_axes([1.02, 0.04, 0.04, 0.92])
cb = mpl.colorbar.ColorbarBase(ax2,
                               cmap=cmap,
                               norm=norm,
                               spacing='proportional',
                               ticks=bounds,
                               boundaries=bounds,
                               format='%1i')
cb.set_ticks(loc)
```

Appendix K (Continued)

```
    ax.set_title('t-SNE Results: U.S. Building Material Companies Intra-CSS
    ↪ Patent Corpus', weight='bold', size=18)
    ax.set_xlabel('Dimension 1', weight='bold').set_fontsize('14')
    ax.set_ylabel('Dimension 2', weight='bold').set_fontsize('14')

    # axis scale
    ax.set_xlim(-100, 100)
    ax.set_ylim(-100, 100)

    # major & minor ticks
    ticks_major = np.arange(-100, 125, 25)
    ax.set_yticks(ticks_major)
    ax.set_xticks(ticks_major)

    # annotate
    ax.annotate('Perplexity = ' + str(perplexity) ,
                xy=(0.02, 0.95),
                fontsize = 14,
                xycoords='axes fraction')

    ax2.yaxis.set_label_coords(-0.40, 0.50)
    ax2.set_ylabel('Cooperative Patent Classification (CPC) [-]',
                   size=12,
                   labelpad=-20)

    ax2.set_yticklabels(labels_cpc['definition'])
    ax2.tick_params(axis=u'both', which=u'both',length=0)

    plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              ) + filename + '
    ↪ _visualization_tnse_perplexity_' + str(
                                      perplexity) + '.png',
                bbox_inches='tight', dpi=300)

    plt.show()

#==============================================================================
# Create perplexity animation of dimension reduction
#==============================================================================
# grab file_names
import glob, os

# create gifs
import imageio

png_dir = working + path + 'img/'
save = (time.strftime("%Y%m%d-%H%M%S") + '_') + filename + '_animation_'

images = []
os.chdir(png_dir)
for file_name in glob.glob("*313*_perplexity_*"): # find pics to animate
    file_path = os.path.join(png_dir, file_name)  # set file path
    images.append(imageio.imread(file_path))      # append pics
imageio.mimsave(save + 'tnse_perplexity.gif', images, duration = 1.0)
#print (images)

# ------------------------------------------------------------------------------
# re-sizing gif functions (change default to 1/4 size, also duration
# ------------------------------------------------------------------------------
'''https://stackoverflow.com/questions/41718892/pillow-resizing-a-gif'''
# ------------------------------------------------------------------------------
from PIL import Image
```

Appendix K (Continued)

```python


def resize_gif(path, save_as=None, resize_to=None):
    """
    Resizes the GIF to a given length:

    Args:
        path: the path to the GIF file
        save_as (optional): Path of the resized gif. If not set, the original
↪ gif will be overwritten.
        resize_to (optional): new size of the gif. Format: (int, int). If not
↪ set, the original GIF will be resized to
                              half of its size.
    """
    all_frames = extract_and_resize_frames(path, resize_to)

    if not save_as:
        save_as = path

    if len(all_frames) == 1:
        print("Warning: only 1 frame found")
        all_frames[0].save(save_as, optimize=True)
    else:
        all_frames[0].save(save_as,
                    optimize=True,
                    save_all=True,
                    append_images=all_frames[1:],
                    duration=100,
                    loop=1)


def analyseImage(path):
    """
    Pre-process pass over the image to determine the mode (full or additive).
    Necessary as assessing single frames isn't reliable. Need to know the mode
    before processing all frames.
    """
    im = Image.open(path)
    results = {
        'size': im.size,
        'mode': 'full',
    }
    try:
        while True:
            if im.tile:
                tile = im.tile[0]
                update_region = tile[1]
                update_region_dimensions = update_region[2:]
                if update_region_dimensions != im.size:
                    results['mode'] = 'partial'
                    break
            im.seek(im.tell() + 1)
    except EOFError:
        pass
    return results


def extract_and_resize_frames(path, resize_to=None):
    """
    Iterate the GIF, extracting each frame and resizing them

    Returns:
```

Appendix K (Continued)

```
            An array of all frames
        """
        mode = analyseImage(path)['mode']

        im = Image.open(path)

        if not resize_to:
            resize_to = (im.size[0] // 5, im.size[1] // 5)

        i = 0
        p = im.getpalette()
        last_frame = im.convert('RGBA')

        all_frames = []

        try:
            while True:
                # print("saving %s (%s) frame %d, %s %s" % (path, mode, i, im.size,
        ↪ im.tile))

                '''
                If the GIF uses local colour tables, each frame will have its own
        ↪ palette.
                If not, we need to apply the global palette to the new frame.
                '''
                if not im.getpalette():
                    im.putpalette(p)

                new_frame = Image.new('RGBA', im.size)

                '''
                Is this file a "partial"-mode GIF where frames update a region of a
        ↪ different size to the entire image?
                If so, we need to construct the new frame by pasting it on top of
        ↪ the preceding frames.
                '''
                if mode == 'partial':
                    new_frame.paste(last_frame)

                new_frame.paste(im, (0, 0), im.convert('RGBA'))

                new_frame.thumbnail(resize_to, Image.ANTIALIAS)
                all_frames.append(new_frame)

                i += 1
                last_frame = new_frame
                im.seek(im.tell() + 1)
        except EOFError:
            pass

        return all_frames

    # ------------------------------------------------------------------------------

    resize_gif(png_dir + '20190313-222456
        ↪ _11_20190311_dimension_reduction_animation_tnse_perplexity.gif',
               save_as = png_dir + save + '.gif')

    #===============================================================================
    # PCA Dimension reduction
    #===============================================================================
```

Appendix K (Continued)

```
'''https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.
    ↪ html'''
'''https://github.com/aviolante/sas-python-work/blob/master/tSneExampleBlogPost.
    ↪ ipynb'''

from sklearn.decomposition import PCA
docs_pca = PCA(n_components=2).fit_transform(docs_vecs)

# create new 'data' df using tsne values and orginal data
pca_corpus_out = pd.DataFrame(docs_pca, columns=['x','y'])
data = pca_corpus_out.join(train)

# Visualize 'http://dsgeek.com/2018/02/19/tfidf_vectors.html'
'''https://www.kdnuggets.com/2018/08/introduction-t-sne-python.html'''
# above ^ has some papers to reference...
'''https://www.datacamp.com/community/tutorials/introduction-t-sne'''

#===============================================================================
#  Create the PCA visualization
#===============================================================================

warnings.filterwarnings(
    action='ignore', module='matplotlib.figure', category=UserWarning,
    message=('This figure includes Axes that are not compatible with
    ↪ tight_layout, '
             'so results might be incorrect.'))

a4_dims = (11.7, 8.27)

# define the data
s = 25 # point size
x = ((data.x)*250)
y = ((data.y)*250)
z = data.class_number

max(x)
min(x)

rcParams.update({'figure.autolayout': True})
plt.rcParams['axes.facecolor'] = 'white'

# setup the plot
fig, ax = plt.subplots(1,1, figsize=a4_dims)

# define the colormap
cmap = plt.cm.jet
cmap.set_under('gray')

# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]

# force the first color entry to be grey
cmaplist[0] = (.5,.5,.5,1.0)

# create the new map
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)

# define the bins and normalize
bounds = np.linspace(0,10,11)
norm = mpl.colors.BoundaryNorm(bounds, cmap.N)
loc = bounds + .5

```

Appendix K (Continued)

```
# make the scatter
scat = ax.scatter(x,
                  y,
                  c=z,
                  s=s,
                  cmap=cmap,
                  norm=norm,
                  lw = 0,
                  alpha=0.25)

# create a second axes for the colorbar
ax2 = fig.add_axes([1.02, 0.04, 0.04, 0.92])
cb = mpl.colorbar.ColorbarBase(ax2,
                               cmap=cmap,
                               norm=norm,
                               spacing='proportional',
                               ticks=bounds,
                               boundaries=bounds,
                               format='%1i')
cb.set_ticks(loc)

ax.set_title('PCA Results: U.S. Building Material Companies Intra-CSS Patent
    ↪ Corpus', weight='bold', size=18)
ax.set_xlabel('Dimension 1', weight='bold').set_fontsize('14')
ax.set_ylabel('Dimension 2', weight='bold').set_fontsize('14')

# axis scale
ax.set_xlim(-50, 150)
ax.set_ylim(-75, 125)

# major & minor ticks
ticks_major_x = np.arange(-50, 175, 25)
ticks_major_y = np.arange(-75, 150, 25)
ax.set_xticks(ticks_major_x)
ax.set_yticks(ticks_major_y)

ax2.yaxis.set_label_coords(-0.40, 0.50)
ax2.set_ylabel('Cooperative Patent Classification (CPC) [-]',
               size=12,
               labelpad=-20)

ax2.set_yticklabels(labels_cpc['definition'])
ax2.tick_params(axis=u'both', which=u'both',length=0)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
                              ) + filename + '_visualization_pca.png',
            bbox_inches='tight', dpi=300)

plt.show()
```

Appendix L

NBER DATA STATISTICAL ANALYSIS

Listing L.1: Statistical analysis of NBER data

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title                :20190208_Paper_Immordino_Updated.py
description          :Patent Analysis Machine Learning
author               :Salvatore Immordino
date created         :Fri Feb 08 19:36:00 2019
date last modified   :Wed Feb 20 19:12:17 2019
version              :0.1
python_version       :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions. """

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
## Libraries needed to run the tool
import numpy as np
import pandas as pd
import math
import statsmodels.formula.api as sm
import matplotlib as mp
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy import stats

# ============================================================================
#  Data importation
# ============================================================================
file_name = 'C:\\Users\\DAD\\Box\\SAM_IMMORDINO_THESIS_WORK\\
    ↪ CME594_MACHINE_LEARNING\\Midterm\\apat63_99.csv'

## Read the csv file accounting for two-row header and three-column index values
input_data = pd.read_csv(file_name, header=0)
#input_data = pd.read_csv(file_name + '.csv', header=0)
data = input_data.apply(pd.to_numeric, errors='coerce')
size = float(len(data.index))

## Print number of rows and colums read
print("{0} rows and {1} columns".format(len(data.index), len(data.columns)))

data.head(n=10)

```

Appendix L (Continued)

```
#number_patents = pd.DataFrame(input_data.groupby('GYEAR').PATENT.count())
#number_patents = input_data[['PATENT', 'GYEAR', 'COUNTRY']].groupby(['GYEAR','
    ↪ COUNTRY']).agg(['count'])
#number_patents = input_data[['PATENT', 'GYEAR']].groupby(['GYEAR']).agg(['count
    ↪ '])
number_patents = input_data[['PATENT', 'GYEAR']].groupby(['GYEAR']).count()
number_patents = number_patents.rename(columns = {'PATENT':'PATENTS'})
number_patents.plot()

min(number_patents.PATENTS)
max(number_patents.PATENTS)

x = number_patents.index
y = number_patents.PATENTS
fit = np.polyfit(x,y,3)

plt.scatter(x,y)
plt.show()

# time stamping
import time

# fit_fn is now a function which takes in x and returns an estimate for y
fit_fn = np.poly1d(fit)
plt.plot(x,y, 'yo', x, fit_fn(x), '--k')
plt.ylim(40000, 160000)
plt.xlabel('Grant Year')
plt.ylabel('Number Patents')

save_loc = 'C:/Users/simmordino/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/
    ↪ Immordino Paper 1/fig/'

plt.savefig(save_loc + (time.strftime("%Y%m%d-%H%M%S") + '_') + '
    ↪ USPTO_issued_patents_Count.png',
bbox_inches='tight',dpi=600)

print("Average number of total patents per year: {0}".format(int(np.nanmean(
    ↪ number_patents))))
print("Median number of total patents per year: {0}".format(int(np.nanmedian(
    ↪ number_patents))))
print("The standard deviation of total patents per year: {0}".format(int(np.std(
    ↪ number_patents, ddof=1))))

#slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)
#print("r-squared:", r_value**2)
#===============================================================================
# best fit of data
# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# set the file path
#path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'
path = '/Users/DAD/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/
    ↪ bin/'
# time stamping
import time

# legend
import matplotlib.patches as mpatches
```

Appendix L (Continued)

```

datos = number_patents['PATENTS'].values

(mu, sigma) = norm.fit(datos)

# the histogram of the data
n, bins, patches = plt.hist(data, 36, normed=1, facecolor='green', alpha=0.75)

bincenters = 0.5*(bins[1:]+bins[:-1])
bin_width = bins[1]-bins[0]

# add a 'best fit' line
y = mlab.normpdf(bins, mu, sigma)
l = plt.plot(bins, y, 'r--', linewidth=2)

#plot
plt.xlabel('Patents Issued Yearly')
plt.ylabel('Probability Density')
plt.suptitle('Density Plot & Histogram of Patents Issued', y=1.0, fontsize=16)
plt.title(r'$\mathrm{Histogram\ of\ IQ:}\ \mu=%.0f,\ \sigma=%.0f$' %(mu, sigma),
    ↪  fontsize=12)

mean = plt.axvline(number_patents['PATENTS'].mean(), color='b', linestyle='
    ↪ dashed', linewidth=2)
median = plt.axvline(number_patents['PATENTS'].median(), color='orange',
    ↪ linestyle='dashed', linewidth=2)

red_line = mpatches.Patch(color='red', label='Normal Distribution')
blue_line = mpatches.Patch(color='blue', label='Mean')
orange_line = mpatches.Patch(color='orange', label='Median')

plt.xlim(40000, 160000)
plt.ylim(0, 0.00005)
plt.legend(handles=[red_line, blue_line, orange_line])
plt.grid(True)

plt.savefig(path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_') + '
    ↪ NBER_density_histrogram.png', bbox_inches='tight', dpi=300)

plt.show()

#==============================================================================
import matplotlib.pyplot as plt
import seaborn as sns

# matplotlib histogram
plt.hist(data, color = 'green', edgecolor = 'black', bins = int(180/5))

# seaborn histogram
sns.distplot(data, hist=True, kde=False,
bins=int(180/5), color = 'blue',
hist_kws={'edgecolor':'black'})
# Add labels
plt.title('Density Plot \& Histogram of Patents Issued')
plt.xlabel('Patents')
plt.ylabel('Probability')



#==============================================================================

A = (data.GYEAR)            # Year patent granted
```

Appendix L (Continued)

```
B = (data.APPYEAR)           # Year patent applied
C = (data.CAT)               # Patent Technological Categories
D = (data.ASSCODE)           # Types of Assignees
P = 2                        # Perecent threshold for counting

cat_1 = 'Chemical'
cat_2 = 'Computers & Communications'
cat_3 = 'Drugs & Medical'
cat_4 = 'Electrical & Electronic'
cat_5 = 'Mechanical'
cat_6 = 'Others'
cat_7 = 'patents'

ass_1 = 'Unassigned'
ass_2 = 'U.S. non-government organizations (mostly corporations)'
ass_3 = 'None U.S. non-government organizations (mostly corporations)'
ass_4 = 'U.S. Indivduals'
ass_5 = 'Non U.S. Indviduals'
ass_6 = 'Feds'
ass_7 = 'Non U.S. Government'

## Print the proportion
print("Proportion of {1} who filed patents from 1963 through 1999: {0}%".format(
    ↪ np.float64(100.0*(list(D).count(1) / (size))).round(1), ass_1))
print("Proportion of {1} who filed patents from 1963 through 1999: {0}%".format(
    ↪ np.float64(100.0*(list(D).count(2) / (size))).round(1), ass_2))
print("Proportion of {1} who filed patents from 1963 through 1999: {0}%".format(
    ↪ np.float64(100.0*(list(D).count(3) / (size))).round(1), ass_3))
print("Proportion of {1} who filed patents from 1963 through 1999: {0}%".format(
    ↪ np.float64(100.0*(list(D).count(4) / (size))).round(1), ass_4))
print("Proportion of {1} who filed patents from 1963 through 1999: {0}%".format(
    ↪ np.float64(100.0*(list(D).count(5) / (size))).round(1), ass_5))
print("Proportion of {1} who filed patents from 1963 through 1999: {0}%".format(
    ↪ np.float64(100.0*(list(D).count(6) / (size))).round(1), ass_6))
print("Proportion of {1} who filed patents from 1963 through 1999: {0}%".format(
    ↪ np.float64(100.0*(list(D).count(7) / (size))).round(1), ass_7))

np.float64(100.0*(list(C).count(6) / (size)))

max(D)
min(D)

# Lag between application & granted
L = (data['GYEAR'] - data['APPYEAR'].shift())
L = ((A - B).shift()).values

plt.hist(((A-B) >=0).values, 10, normed=1, facecolor='green', alpha=0.75)


#==============================================================================
# list(L).count(1)
# list(L).count(2)
# list(L).count(3)
# list(L).count(4)
#
# elements, repeats = np.unique(L, return_counts=True)
# index = repeats.argmax()
# elem = elements[index]
#
# print "Max elem is " + str(elem) + " with " + str(repeats[index]) + "
    ↪ repetitions."
#==============================================================================
```

Appendix L (Continued)

```python


# Average lag between application & granted
lag = (np.nanmean(L))
np.float64(np.std(L))


print("Average lag between patent file and granted is {} years".format(np.
    ↪ float64(lag).round(2)))

((data['GYEAR'] - data['APPYEAR']).values)


## Count the sets
count_all_instances = ((A)>=0).sum()*1.0
a = count_over_2_years = ((data['GYEAR'] - data['APPYEAR'].shift()) > 2).sum()
b = count_drugs_medical = (C.shift() == 3).sum()
a_b = count_over_2_years_cat_3 = (((data['GYEAR'] - data['APPYEAR'].shift()) >
    ↪ 2) & (C.shift() == 3)).sum()

## Print Support: {0}% of all instances were both A & B
support = ((100.0*(a_b) / (count_all_instances)))
print("Support: {0}% of all {1} took over {2} years and were classified as {3}."
.format(np.float64(support).round(1), cat_7, P, cat_3))

##Print Confidence: {0}% of A were also of B
confidence = ((100.0*(a_b) / (a)))
print("Confidence: {0}% of {1} that took over {2} years were also classified as
    ↪ {3}."
.format(np.float64(confidence).round(1), cat_7, P, cat_3))

## Print Lift: Given A it is {0} times more likely to have B
lift = ((confidence) / (100.0*(b) / (count_all_instances)))
# lift = (100.0*sup_AUB) / (sup_A * sup_B)    # alternate lift calculation
print("Lift: Given {1} that take over {2} years it is {0} times more likely they
    ↪  are classified as {3}."
.format(np.float64(lift).round(2), cat_7, P, cat_3))

## lift(A,B) = sup(A U B) / sup(A) * sup(B) = P(A|B) / P(A) * P(B)
sup_A = (100.0*(a) / (count_all_instances))                #P(A)
sup_B = (100.0*(b) / (count_all_instances))                #P(B)
sup_AUB = ((100.0*(a_b) / (count_all_instances))) #P(A U B)

## The probability that event A occurs, given that event B has already occurred
## is P_A_given_B = P(A|B) = P(A and B) / P(B)
P_A_given_B = (100.0*(sup_AUB / sup_B)) #P(A|B) or All_Conf
## P_B_given_A = P(B|A) = P(A and B) / P(A)
P_B_given_A = (100.0*(sup_AUB / sup_A)) #P(B|A) or Confidence

## all_conf(A,B) = sup(A U B) / max{sup(A),sup(B)}
all_conf = ((100.0*sup_AUB) / max(sup_A,sup_B))/100
print("all_conf: The smallest confidence of the two assocation rules: {0}"
.format(np.float64(all_conf).round(2)))

##  max_conf(A,B) = max{P(A|B),P(B|A)}
max_conf = (max(P_A_given_B,P_B_given_A))/100.0
print("max_conf: The maximum confidence of the two assocation rules: {0}"
.format(np.float64(max_conf).round(2)))

## Kulc(A,B) = 1/2(P(A|B)+P(B|A))
kulc = 0.5*((P_A_given_B + P_B_given_A)/100)
print("kulc: The average confidence between the two assocation rules: {0}"
```

Appendix L (Continued)

```
.format(np.float64(kulc).round(2)))

## cosine(A,B) = P(A U B) /SQRT(P(A)*P(B)) = sup(AUB) / SQRT(sup(A)*sup(B))
cosine = sup_AUB / math.sqrt(sup_A*sup_B)
print("cosine: The cosine value between the two rules {0}"
.format(np.float64(cosine).round(2)))
```

Appendix M

NBER DATA NEURAL NETWORK

Listing M.1: Neural network prediction of patent technical category using NBER data

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#===============================================================================
'''
title                :NBER_Neural_Networks_and_Deep_Learning.py
description          :Patent Analysis Machine Learning
author               :Salvatore Immordino
date created         :Mon Apr 04 12:40:00 2016
date last modified   :Wed Feb 20 19:12:17 2019
version              :0.1
python_version       :3.7.1
'''
#===============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions. """

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.
#===============================================================================
# IMPORT STATEMENTS
#===============================================================================
#Libraries needed to run the tool
import numpy as np
import pandas as pd
from sklearn.neural_network import MLPRegressor
from sklearn.neural_network import MLPClassifier
from sklearn import preprocessing #to normalize the values
from sklearn.model_selection import train_test_split #new for version 0.18 but
    ↪ seems to be out soon
import matplotlib.pyplot as plt
from sklearn.preprocessing import Imputer

# ==============================================================================
#  Data importation
# ==============================================================================
#Ask for file name and read the file
#file_name = raw_input("Name of file:")
#file_name = 'C:/Users/DAD/Desktop/CME_594/HW9/test_apat63_99'
file_name = 'C:/Users/simmordino/Desktop/CME_594/HW9/test_apat63_99'
input_data = pd.read_csv(file_name + '.csv', header=0, index_col=0)
input_data = input_data.sample(n=5000)

#analysis_type = raw_input("Analysis Type 'R' or 'C': ")
analysis_type = 'C'

#Print number of rows and colums read
print("{0} rows and {1} columns".format(len(input_data.index), len(input_data.
    ↪ columns)))
print("")
```

```python

#Defining X1, X2, and all the data X
#Defining X1 thru X7, and all the data X
X1 = input_data.GYEAR.values
X2 = input_data.GDATE.values
X3 = input_data.APPYEAR.values
X4 = input_data.SUBCAT.values
X5 = input_data.ASSCODE.values
X6 = input_data.NCLASS.values
X7 = pd.get_dummies(input_data["COUNTRY"])

X_raw = np.column_stack((X1, X2, X3, X4, X5, X6, X7))

#Normalizing or not the data
#X =preprocessing.normalize(X_raw) #does not seem to improve the accuracy
X = X_raw

imp = Imputer(missing_values='NaN', strategy='most_frequent', axis=0)
X = imp.fit_transform(X)

if analysis_type == 'R':
Y = input_data.area.values.astype(float)
else:
Y = input_data.CAT.values

#Using Built in train test split function in sklearn
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)

if analysis_type == 'R':
#Fit the neural network for Regression purposes (i.e., you expect a continuous variable out)
#Note that 'sgd' and 'adam' require a batch_size and the function is not as clear
# http://scikit-learn.org/dev/modules/generated/sklearn.neural_network.MLPRegressor.html
acti = ['logistic', 'tanh', 'relu'] #activation..sklearn.neural_network.MLPRegressor
algo = ['l-bfgs', 'sgd', 'adam'] # preffers l-bfgs
learn = ['constant', 'invscaling', 'adaptive']
# All selections from above
neural = MLPRegressor(activation=acti[0], algorithm=algo[0], batch_size = 1, learning_rate = learn[0], hidden_layer_sizes=(7,))
neural.fit(X_train, Y_train) # THIS IS YOUR FIT SET IT EQUAL TO SOMETHING AND APPLY TO A DATA SET IT WILL GIVE YOUR y'S
neural_score = neural.score(X_test, Y_test)
#List parameters settings
#neural.get_params(deep=TRUE)
#print list_params
print("Analysis Type: {0}".format(analysis_type))
print("Algorithm Type: {0}".format(neural.algorithm))
print("Learning Type: {0}".format(neural.learning_rate))
print("Activation Type: {0}".format(neural.activation))
print("Shape of neural network: {0}".format([coef.shape for coef in neural.coefs_]))
print("")
print("Coefs: ")
print(neural.coefs_[0].round(2))
print(neural.coefs_[1].round(2))
print("Intercepts: {0}".format(neural.intercepts_))
print("Iteration: {0}".format(neural.n_iter_))
print("Layers: {0} - always 3 seems wrong".format(neural.n_layers_))
print("Outputs: {0}".format(neural.n_outputs_))
```

Appendix M (Continued)

```
print("Activation: {0}".format(neural.out_activation_))

#Assess the fitted Neural Network
print("Y test and predicted")
print(Y_test.round(1))
print(neural.predict(X_test).round(1))

print("Accuracy as Pearson's R2: {0}".format(neural_score.round(4)))

else:

#Setting up neorode fitting loop the model with training data
accuracy = []
k = range(1,51)
for i in range(0, len(k)):
#Fit the neural network for Classification purposes (i.e., you expect a
    ↪ continuous variable out)
#Note that 'sgd' and 'adam' require a batch_size and the function is not as
    ↪ clear
acti = ['logistic', 'tanh', 'relu']
algo = ['l-bfgs', 'sgd', 'adam']
learn = ['constant', 'invscaling', 'adaptive']
# increasing the hidden layers has a big impact    hidden_layer_sizes=(7,,3,) 7
    ↪ is input, hidden 3...shows (7,7), (7,3), (3,1)
neural = MLPClassifier(activation=acti[0], algorithm=algo[0], batch_size = 1,
    ↪ learning_rate = learn[2], hidden_layer_sizes=(k[i],))
# the above seven needs to go from 7 - 50 with for loop and show accuracy chart.
neural.fit(X_train, Y_train)
neural_score = neural.score(X_test, Y_test)
#List parameters settings
#neural.get_params(deep=TRUE)
#print list_params
#==============================================================================
#        print("Analysis Type: {0}".format(analysis_type))
#        print("Activation Type: {0}".format(neural.activation))
#        print("Algorithm Type: {0}".format(neural.algorithm))
#        print("Learning Type: {0}".format(neural.learning_rate))
#        print("Classes: {0}".format(neural.classes_))
#        print("Shape of neural network: {0}".format([coef.shape for coef in
    ↪ neural.coefs_]))
#        print("")
#        print("Coefs: ")
#        print(neural.coefs_[0].round(2))
#        print(neural.coefs_[1].round(2))
#        print("Intercepts: {0}".format(neural.intercepts_))
#        print("Iteration: {0}".format(neural.n_iter_))
#        print("Layers: {0} - always 3 seems wrong".format(neural.n_layers_))
#        print("Outputs: {0}".format(neural.n_outputs_))
#        print("Activation: {0}".format(neural.out_activation_))
#
#        #Assess the fitted Neural Network
#        print("Y test and predicted")
#        print(Y_test)
#        print(neural.predict(X_test))
#        print("")
#        print("Mean Accuracy: {0}".format(neural_score.round(4)))
#==============================================================================

accuracy.append(neural_score.round(4))

best_accuracy = accuracy.index(max(accuracy)) + 1 #this gives the neurode count
    ↪ that gave the highest accuracy
```

Appendix M (Continued)

```

plt.figure(figsize=(15,10))
plt.plot(k, accuracy)
plt.title("Value of k is " + str(best_accuracy), fontsize=16)
plt.xlabel("neurode count (k)", fontsize=16) #Adding axis labels
plt.ylabel("neural score - accuracy", fontsize=16)
plt.xlim(0, 50) #Setting limits of axes
plt.ylim(0, 1)
plt.annotate("Analysis Type = " + str(analysis_type), xy=(0.01, 0.99), xycoords=
    ↪ 'axes fraction',
fontsize=12, horizontalalignment='left', verticalalignment='top')
plt.annotate("Activation Type = " + str(neural.activation), xy=(0.01, 0.96),
    ↪ xycoords='axes fraction',
fontsize=12, horizontalalignment='left', verticalalignment='top')
plt.annotate("Algorithm Type = " + str(neural.algorithm), xy=(0.01, 0.93),
    ↪ xycoords='axes fraction',
fontsize=12, horizontalalignment='left', verticalalignment='top')
plt.annotate("Learning Type = " + str(neural.learning_rate), xy=(0.01, 0.90),
    ↪ xycoords='axes fraction',
fontsize=12, horizontalalignment='left', verticalalignment='top')
plt.annotate("Maximum Accuracy = " + str(max(accuracy).round(2)), xy=(0.01,
    ↪ 0.87), xycoords='axes fraction',
fontsize=12, horizontalalignment='left', verticalalignment='top')
#==============================================================================
#Saving plots
plt.savefig(file_name +'_'+ str(neural.activation) +'_'+ str(neural.algorithm) +
    ↪ '_'+ str(neural.learning_rate) + '_ANN_loop.png', dpi=300)
plt.savefig(file_name +'_'+ str(neural.activation) +'_'+ str(neural.algorithm) +
    ↪ '_'+ str(neural.learning_rate) + '_ANN_loop.pdf', format='pdf', dpi=300)
#plt.savefig(file_name + '_plot.png') #Saving the plot
plt.show()
#==============================================================================

plt.clf()
```

Appendix N

NBER DATA RANDOM FOREST

Listing N.1: Random forest prediction of patent technical category using NBER data

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title               :NBER_Random_Forest.py
description         :Patent Analysis Machine Learning
author              :Salvatore Immordino
date created        :Sun Mar 13 18:49:20 2016
date last modified  :Wed Feb 20 19:12:17 2019
version             :0.1
python_version      :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions. """

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
#Libraries needed to run the tool
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.cross_validation import train_test_split
import matplotlib.pyplot as plt

from os import system
import matplotlib.image as mpimg
from sklearn import preprocessing
from sklearn.preprocessing import Imputer
from sklearn.preprocessing import LabelEncoder

# ============================================================================
#  Data importation
# ============================================================================
#file_name = raw_input("Name of file:")
file_name = 'test_apat63_99'
input_data = pd.read_csv(file_name + '.csv', header=0, index_col=0)
input_data = input_data.sample(n=1000)

#Print number of rows and colums read
print("{0} rows and {1} columns".format(len(input_data.index), len(input_data.
    ↪ columns)))
print("")

```

Appendix N (Continued)

```
#Defining X1 thru X7, and all the data X
X1 = input_data.GYEAR.values
X2 = input_data.GDATE.values
X3 = input_data.APPYEAR.values
X4 = input_data.SUBCAT.values
X5 = input_data.ASSCODE.values
X6 = input_data.NCLASS.values
X7 = input_data.COUNTRY

le = preprocessing.LabelEncoder()
le.fit(input_data.COUNTRY)
# list(le.classes_) #show the unique classes
X7 = le.transform(X7)
# list(le.inverse_transform([22])) #check to figure out what 22 = U.S.

X = np.column_stack((X1, X2, X3, X4, X5, X6, X7))

imp = Imputer(missing_values='NaN', strategy='most_frequent', axis=0)
X = imp.fit_transform(X)

Y = input_data.CAT.values

#Define model parameters
crit_choice = ['gini', 'entropy']
crit = crit_choice[0]
n_estimators = 5
test_size = 0.15

#Using Built in train test split function in sklearn
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = test_size)

x_min, x_max = X[:,0].min() - 1, X[:,0].max() + 1 #Defines min and max on the x-
    ↪ axis
y_min, y_max = X[:,1].min() - 1, X[:,1].max() + 1 #Defines min and max on the y-
    ↪ axis

plot_step = (x_max - x_min)/300 #step size in the mesh to plot entire areas

xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
np.arange(y_min, y_max, plot_step)) #Defines meshgrid

decitree = tree.DecisionTreeClassifier(criterion=crit).fit(X_train, Y_train) #
    ↪ Decision Tree
decitree_predict = decitree.predict(X_test)
decitree_score = metrics.accuracy_score(decitree_predict, Y_test)

randfor = RandomForestClassifier(n_estimators = n_estimators, criterion=crit).
    ↪ fit(X_train, Y_train) #Random Forest
randfor_predict = randfor.predict(X_test)
randfor_score = metrics.accuracy_score(randfor_predict, Y_test)

#Export tree properties in graph format
#http://stackoverflow.com/questions/27817994/visualizing-decision-tree-in-scikit
    ↪ -learn
#http://stackoverflow.com/questions/19613239/make-a-graph-in-pydot-from-decision
    ↪ -tree-in-sklearn-python?rq=1

dotfile = open("HW9_bonus.dot", 'w')
tree.export_graphviz(decitree, out_file = dotfile)
dotfile.close()

```

Appendix N (Continued)

```
#to see the graph need to use 'dot -Tpng HW9_data.dot -o HW9_data.png' in
    ↪ command prompt
system("dot -Tpng HW9_bonus.dot -o HW9_bonus.png")
plt.figure(figsize=(15,10))
img = mpimg.imread('HW9_bonus.png')
plt.xticks([])
plt.yticks([])
plt.imshow(img)
plt.show()

print (Y_test)
print("Decision Tree")
print (decitree_predict)
print (decitree_score)
print("Random Forest")
print (randfor_predict)
print (randfor_score)
```

Appendix O

PATENT DATA PREDICTION

Listing O.1: Patent data predicition

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title                :12_20190311_prediction.py
description          :Patent Analysis Machine Learning
author               :Salvatore Immordino
date created         :Wed Jan  13 16:00:36 2019
date last modified   :Fri Mar 22 19:12:17 2019
version              :0.1
python_version       :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions.

Data files were downloaded from the United States Patent Office PatensView Data
    ↪ Download located at www.patentsview.org.
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
# basic libraries needed to run the tool
import pandas as pd
import numpy as np

# graphing functions
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib import rcParams

# drop the warnings
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

# time stamping
import time

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# set the file path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'

# ============================================================================
#  Data importation
# ============================================================================
```

Appendix O (Continued)

```
input_data12 = pd.read_csv(working + path + 'input_data7.csv', header=0,
                          usecols = ['patent_id',
                                     'title',
                                     'abstract',
                                     'claims',
                                     'combined',
                                     'grant_date',
                                     'grant_date_year',
                                     'organization',
                                     'org_number',
                                     'section_id',
                                     'class_number'],
                          dtype = {'patent_id':object},
                          parse_dates = ['grant_date'],
                          na_values = ['no info', '.'],
                          converters={'organization': str},
                          encoding = "iso-8859-1")

#==============================================================================
#  Group statistics
#==============================================================================
prediction = pd.DataFrame({'cit_count':X4, 'css_count':Y4, 'mean_css':Y8})
prediction.reset_index()

input_data12 = pd.merge(left=input_data12,
                        right=prediction,
                        how='left',
                        left_on=['patent_id'],
                        right_on=['patent_id'],
                        sort=True,
                        left_index=True)

input_data12.to_csv(working + path + 'input_data12.csv', index=False)

input_data12 = input_data12.drop(['title','abstract','organization','section_id
    ↪ ','claims','combined'], axis=1)

#==============================================================================
#  Document Term Matrix (dtm)
#==============================================================================
# Convert sparse matrix (DTM) to dataframe to see word frequencies.
dtm = tfidf.todense() # doc_term_matrix (dtm)
#patent_vectors = pd.DataFrame(dtm, columns=feature_names, index=[input_data12.
    ↪ patent_id])
patent_vectors = pd.DataFrame(dtm, index=[input_data12.patent_id])
patent_vectors.reset_index()
patent_vectors.head(5)
#==============================================================================
input_data12 = pd.merge(left=input_data12,
                        right=patent_vectors,
                        how='left',
                        left_on=['patent_id'],
                        right_on=['patent_id'],
                        sort=True,
                        left_index=True)


#input_data12.info(verbose=True) # tell me all the things

'''https://stackoverflow.com/questions/13187778/convert-pandas-dataframe-to-
    ↪ numpy-array-preserving-index'''

```

Appendix O (Continued)

```
numpy_matrix = input_data12.as_matrix()
X = numpy_matrix


#Print number of rows and colums read
print("{0} rows and {1} columns".format(len(input_data12.index), len(
    ↪ input_data12.columns)))
print("")

#Defining X1, X2, and all the data X
#Defining X1 thru X7, and all the data X
X1 = data.x # tnse x0
X2 = data.y # tnse x2
X3 = data.patent_id
X4 = data.grant_date
X5 = data.grant_date_year
X6 = data.org_number
X7 = input_data12.cit_count
X8 = input_data12.css_count
X9 = input_data12.mean_css
X10 = input_data12.class_number

'''https://stackoverflow.com/questions/13187778/convert-pandas-dataframe-to-
    ↪ numpy-array-preserving-index'''

#X_raw = np.column_stack((X1, X2, X3, X4, X5, X6, X8, X9, X10))
#
#from sklearn.preprocessing import Imputer
#imp = Imputer(missing_values='NaN', strategy='most_frequent', axis=0)
#
#X = imp.fit_transform(X_raw)
#X = X_raw

#y = X10 # predict cpc class
#y = ?? # predict mean time to peak citation
y = X7 # predict citation

#==============================================================================
#  Plot the ANN dataset for CPC Code
#==============================================================================
'''https://medium.com/ml-algorithms/neural-networks-for-decision-boundary-in-
    ↪ python-b243440fb7d1'''
plt.scatter(X[:,0], X[:,1], s=40, c=y, cmap=cmap)
plt.show()

# -----------------------------------------------------------------------------
#labels_cpc = pd.read_csv(working + path + 'labels_cpc.csv', header=0)
#
#warnings.filterwarnings(
#    action='ignore', module='matplotlib.figure', category=UserWarning,
#    message=('This figure includes Axes that are not compatible with
    ↪ tight_layout, '
#             'so results might be incorrect.'))
#
#rcParams.update({'figure.autolayout': True})
#plt.rcParams['axes.facecolor'] = 'white'
#
## setup the plot
#fig, ax = plt.subplots(1,1, figsize=(15,10))
#
# define the colormap
cmap = plt.cm.jet
```

Appendix O (Continued)

```
cmap.set_under('gray')

# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]

# force the first color entry to be grey
cmaplist[0] = (.5,.5,.5,1.0)

# create the new map
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)
#
## define the bins and normalize
#bounds = np.linspace(0,10,11)
#norm = mpl.colors.BoundaryNorm(bounds, cmap.N)
#loc = bounds + .5
#
## make the scatter
#scat = ax.scatter(X[:,0], X[:,1], s=40, c=y, cmap=cmap)
#
## create a second axes for the colorbar
#ax2 = fig.add_axes([1.02, 0.04, 0.04, 0.92])
#cb = mpl.colorbar.ColorbarBase(ax2,
#                                cmap=cmap,
#                                norm=norm,
#                                spacing='proportional',
#                                ticks=bounds,
#                                boundaries=bounds,
#                                format='%1i')
#cb.set_ticks(loc)
#
#ax.set_title('ANN Scatter: U.S. Building Material Companies Intra-CSS Patent
    ↪ Corpus', weight='bold', size=18)
#ax.set_xlabel('Dimension 1', weight='bold').set_fontsize('14')
#ax.set_ylabel('Dimension 2', weight='bold').set_fontsize('14')
#
### axis scale
#ax.set_xlim(-100, 100)
#ax.set_ylim(-100, 100)
#
## major & minor ticks
#ticks_major = np.arange(-100, 125, 25)
#ax.set_yticks(ticks_major)
#ax.set_xticks(ticks_major)
#
#ax2.yaxis.set_label_coords(-0.40, 0.50)
#ax2.set_ylabel('Cooperative Patent Classification (CPC) [-]',
#                size=12,
#                labelpad=-20)
#
#ax2.set_yticklabels(labels_cpc['definition'])
#ax2.tick_params(axis=u'both', which=u'both',length=0)
#
##plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
##                              ) + 'visualization_tnse.png',
##            bbox_inches='tight', dpi=300)
#
#plt.show()
################################################################################

# Train the logistic regression classifier
import sklearn.linear_model
clf = sklearn.linear_model.LogisticRegressionCV()
```

Appendix O (Continued)

```
clf.fit(X, y)

# using Built in train test split function in sklearn
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)
mlp.fit(X_train, y_train.values.ravel())

predictions = mlp.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
```

Appendix P

PATENT CIATATION CSS VENN

Listing P.1: Patent CIatation CSS Venn

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#==============================================================================
'''
title               :13_2019023_patent_cit_css_venn.py
description         :Patent Analysis Machine Learning
author              :Salvatore Immordino
date created        :Sun May 01 15:32:06 2018
date last modified  :Wed Feb 20 19:12:17 2019
version             :0.1
python_version      :3.7.1
'''
#==============================================================================
"""Big data analysis of patent documents on python. The aim of this research
was to determine if machine learning could be used to study inventiveness and
inventive knowledge flow independent of historical methods like citation
analysis. Using natural language processing and network analysis, this research
aimed to identify relatedness between patents based on the inventive language
used by patent seekers to describe their inventions. """
#==============================================================================
# IMPORT STATEMENTS
#==============================================================================
# basic libraries needed to run the tool
import pandas as pd
import numpy as np
import re

# graphing functions
import matplotlib.pyplot as plt
#import matplotlib.mlab as mlab
#import matplotlib.ticker as ticker

# drop the warnings
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

# time stamping
import time

# find out your current working directory
import os
print(os.getcwd())
working = (os.getcwd())
working = working.replace('\\', '/') # replaced all instances of \ with \\

# needed for this code
import seaborn as sns
#from scipy import stats
#from scipy.stats import norm

# venn imports
from matplotlib_venn import venn3 #venn3_circles
from matplotlib_venn import venn2
```

Appendix P (Continued)

```

# legend
#import matplotlib.patches as mpatches

# set filename and path
path = '/Box/SAM_IMMORDINO_THESIS_WORK/LaTeX_THESIS/Immordino Paper 1/bin/'
filename = '13_20190311_patent_cit_css_venn'

# set palette  orange, green, red, purple, brown, pink, grey, tan, lblue, blue
color_list = sns.set_palette(palette=sns.color_palette("muted"))
muted_colors = palette=sns.color_palette("muted")
muted_list = (muted_colors.as_hex())
print(muted_colors.as_hex())

# ============================================================================
#  Data importation
# ============================================================================

application_dates = pd.read_csv(working + path + 'application_dates.csv',
                                header=0,
                                dtype = {'patent_id':object},
                                parse_dates = ['app_date'])

input_data8 = pd.read_csv(working + path + 'input_data8.csv', header=0,
                          dtype = {'patent_id':object,
                                   'citing_patent':object},
                          parse_dates = ['grant_date','cit_app_date', '
    ↪ pat_app_date'],
                          na_values = ['no info', '.'],
                          encoding = "iso-8859-1")

input_data9 = pd.read_csv(working + path + 'input_data9.csv', header=0,
                          dtype = {'patent_id':object,
                                   'css_patent_id':object},
                          parse_dates = ['grant_date','css_app_date', '
    ↪ pat_app_date'],
                          na_values = ['no info', '.'],
                          encoding = "iso-8859-1")

input_data9.info(verbose=True) # tell me all the things
input_data9.head(10)

# ============================================================================
#  Create Intersection Tables
# ============================================================================

df_intra_css = input_data9[['patent_id','css_patent_id']].copy()
df_intra_css = df_intra_css.rename(columns = {'css_patent_id':'citing_patent'})
df_intra_css = df_intra_css.drop_duplicates()
df_intra_css.info(verbose=True) # tell me all the things
df_intra_css.head(20)

df_intra_cit = input_data8[['patent_id','citing_patent']].copy()
df_intra_cit.info(verbose=True) # tell me all the things
df_intra_cit = df_intra_cit.drop_duplicates()
df_intra_cit.head(20)

df_intersection = pd.merge(df_intra_cit,
                           df_intra_css,
                           how='inner',
                           on=['patent_id', 'citing_patent'])

```

```
df_intersection = df_intersection.drop_duplicates()
df_intersection.head(10)

# remeber to reload inpot tables 8 & 9
Z0 = input_data8.groupby('patent_id')['citing_patent'].nunique().nlargest(12).
    ↪ reset_index() # count cited
Z1 = df_intersection.groupby('patent_id')['citing_patent'].nunique() #
    ↪ intersection counts
Z2 = df_intra_css[df_intra_css['patent_id'] == '3935021'] # list intra css
Z3 = df_intra_cit[df_intra_cit['patent_id'] == '3935021'] # list intra cit

# grap 12 random patents
Z4 = input_data8.groupby('patent_id')['citing_patent'].nunique().sample(12).
    ↪ reset_index() # count cited

# range function that allows non-integer steps
def xfrange(start, stop, step):
    i = 0
    while start + i * step < stop:
        yield start + i * step
        i += 1

## loop function to compare patents by method
new_index = pd.Series(xfrange(0, 26, 1))
#patents = ['3935021', '6673144', '6432267','7585388']
patents = Z0[('patent_id')].tolist()
patent = 0
data = pd.DataFrame([])

for i, patent in enumerate(patents):
    # css related patents
    css_data_cited = input_data9.loc[input_data9['patent_id'] == patent].
    ↪ drop_duplicates()
    css_data_cited = css_data_cited[['patent_id', 'css_patent_id', 'pat_app_year
    ↪ ', 'css_app_year']]
    css_data_cited['css_years_past'] = css_data_cited['css_app_year'] -
    ↪ css_data_cited['pat_app_year']
    css_data_cited = css_data_cited[['patent_id','css_years_past']]

    # cit related patents
    cit_data_cited = input_data8.loc[input_data8['patent_id'] == patent].
    ↪ drop_duplicates()
    cit_data_cited = cit_data_cited[['patent_id', 'citing_patent', 'pat_app_year
    ↪ ', 'cit_app_year']]
    cit_data_cited['cit_years_past'] = cit_data_cited['cit_app_year'] -
    ↪ cit_data_cited['pat_app_year']
    cit_data_cited = cit_data_cited[['patent_id','cit_years_past']]

    # inputs
    css = css_data_cited['css_years_past'].groupby(css_data_cited['
    ↪ css_years_past']).count()
    cit = cit_data_cited['cit_years_past'].groupby(cit_data_cited['
    ↪ cit_years_past']).count()

    combined = (pd.concat([css, cit], axis = 1)).reindex(new_index, fill_value
    ↪ =0).fillna(0)
    combined = combined.rename(columns={combined.columns[0]: patent + '_' + "css
    ↪ ",
                                        combined.columns[1]: patent + '_' + "cit
    ↪ "})

    data = pd.concat([data, combined], axis=1)
```

Appendix P (Continued)

```

    print ("patent {} = {}".format(i, patent))

    if i == len(patents) - 1:
        print ("stacking and rename patent columns")
        data_wide = data
        data = data.stack().reset_index()
        data.columns = ['years_past','related_method','related_count']

# ============================================================================
#  Create Intersection Tables
# ============================================================================

df_intra_css = input_data9[['patent_id','css_patent_id']].copy()
df_intra_css = df_intra_css.rename(columns = {'css_patent_id':'citing_patent'})
df_intra_css = df_intra_css.drop_duplicates()
df_intra_css.info(verbose=True) # tell me all the things
df_intra_css.head(20)

df_intra_cit = input_data8[['patent_id','citing_patent']].copy()
df_intra_cit.info(verbose=True) # tell me all the things
df_intra_cit = df_intra_cit.drop_duplicates()
df_intra_cit.head(20)

df_intersection = pd.merge(df_intra_cit,
                           df_intra_css,
                           how='inner',
                           on=['patent_id', 'citing_patent'])

df_intersection = df_intersection.drop_duplicates()
df_intersection.head(10)

# grab some patents; top 12
Z0 = input_data8.groupby('patent_id')['citing_patent'].nunique().nlargest(12).
    ↪ reset_index() # count cited
Z1 = df_intersection.groupby('patent_id')['citing_patent'].nunique() #
    ↪ intersection counts

# a single patent by number
Z_NUM = '3935021'

Z2 = df_intra_css[df_intra_css['patent_id'] == Z_NUM] # list intra css
Z2.count()

Z3 = df_intra_cit[df_intra_cit['patent_id'] == Z_NUM] # list intra cit
Z3.count()

# 12 random patents
Z4 = input_data8.groupby('patent_id')['citing_patent'].nunique().sample(12).
    ↪ reset_index() # count cited

# ============================================================================
# intra-citations over time create data table
# ============================================================================

# googgle calls them family to family citations

## loop function to compare patents by method
#patents = ['3935021', '6673144', '6432267','7585388']
patents = Z0[('patent_id')].tolist() #from input_data8 which was cited patents
patent = 0
data_venn = pd.DataFrame([])
```

Appendix P (Continued)

```

for i, patent in enumerate(patents):
    # css related patents
    css_venn = df_intra_css.loc[df_intra_css['patent_id'] == patent].
    ↪ drop_duplicates()

    # cit related patents
    cit_venn = df_intra_cit.loc[df_intra_cit['patent_id'] == patent].
    ↪ drop_duplicates()

    combined_venn = (pd.concat([css_venn, cit_venn], axis = 1))
    combined_venn = combined_venn.rename(columns={combined_venn.columns[0]:
    ↪ patent + '_' + "css", combined_venn.columns[1]: patent + '_' + "cit"})

    data_venn = pd.concat([data_venn, combined_venn], axis=1)

    print ("patent {} = {}".format(i, patent))

    if i == len(patents) - 1:
        print ("stacking and rename patent columns")
        data_venn_wide = data_venn
        data_venn = data_venn.stack().reset_index().drop('level_0', 1)
        data_venn.columns = ['related_method','related_patents']
        data_venn = data_venn.sort_values(['related_method'], ascending=[True])

# ============================================================================
# Google similar patent documents to 3935021
# ============================================================================
'''https://patents.google.com/patent/US3935021A/en'''
'''https://www.pcworld.com/article/3049943/software/excel-pro-tips-importing-and
    ↪ -parsing-data.html'''

# Google Similar Documents
google_3935021 = pd.read_csv(working + path + '3935021_Google_Sim_Docs.csv',
                             header=0,
                             dtype = {'patent_id':object},
                             parse_dates = ['publication_date'])

# select just patent numbers with no letters
google_3935021['patent_id'] = google_3935021['patent_id'].apply(
        lambda x: re.search(r'\d+', x).group())

# merge google similar documents with application dates table
google_3935021 = pd.merge(left=google_3935021,
                             right=application_dates,
                             how='left',
                             left_on=['patent_id'],
                             right_on=['patent_id'],
                             sort=True,
                             left_index=True)

# drop any publications greater than 7 numbers long, just real patents
google_3935021 = google_3935021[~(google_3935021['patent_id'].str.len() > 7)]

# select only patents that came after orginal patent_id grant date
google_3935021 = google_3935021[(google_3935021['publication_date'].dt.year >=
    ↪ 1976)]

# patent drop rows outside of organization group
google_3935021 = google_3935021[np.isfinite(google_3935021['app_year'])]

'''
```

Appendix P (Continued)

```
885 = Henry Clark; 150 = Dow; 341, 744, 987 = WR Grace; 995 = CBI; 288 = SKW;
192 = Aqualon; 031, 033 = Celotex; 839 = Rohm & Haas; 550 = Tec Inc;
237 = Walker Industries; 903 = National, cement products
'''

'''prediction go back an train/test on CPC catergory'''

# ============================================================================
#  Patent intersection between CSS, CIT, and Google using VENNN
# ============================================================================
''' https://pypi.org/project/matplotlib-venn/'''
'''https://python-graph-gallery.com/172-custom-venn-diagram/'''

patent_intersection_no = '3935021'

css_intersection = set(df_intra_css.loc[df_intra_css['patent_id'] ==
    ↪ patent_intersection_no].drop_duplicates()['citing_patent'])
cit_intersection = set(df_intra_cit.loc[df_intra_cit['patent_id'] ==
    ↪ patent_intersection_no].drop_duplicates()['citing_patent'])
google = set(google_3935021['patent_id'])

fig = plt.figure(figsize=(20,10))
g = venn3([css_intersection, cit_intersection, google], ('CSS', 'Cited','Google'
    ↪ ))
plt.title('Method Comparison Venn Diagram: Patent US' + str(
    ↪ patent_intersection_no), fontsize=16)

g.get_patch_by_id('100').set_color('C0')
g.get_patch_by_id('110').set_color('C5')
g.get_patch_by_id('010').set_color('C1')
g.get_patch_by_id('001').set_color('C4')
g.get_patch_by_id('100').set_alpha(0.6)
g.get_patch_by_id('110').set_alpha(0.6)
g.get_patch_by_id('010').set_alpha(0.6)
g.get_patch_by_id('100').set_alpha(0.6)

for text in g.set_labels:
    text.set_fontsize(14)
for text in g.subset_labels:
    text.set_fontsize(16)

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
 ) + filename + '_cssvcitvgoo_patent_' + patent_intersection_no + '_venn3.png',
            bbox_inches='tight',dpi=300)

# ============================================================================
# intra-citations over time venn plot
# ============================================================================

# set fig parameters
patents = Z0[('patent_id')].tolist()
num_cols = 3
num_rows = 4
num_plots = len(patents)

# set figsize here
fig, axs = plt.subplots(figsize=(13,10),
                        #sharey=True,
                        #sharex=True,
                        ncols=num_cols,
                        nrows=num_rows)
```

Appendix P (Continued)

```
plt.suptitle('Method Comparison: Venn Diagram of Top 12 Most Cited Patents', y
    ↪ =1.02, fontsize=18)

# iterate through all axes and create a venn plot
for i, ax in enumerate(axs.flatten()):
    css_set = set(df_intra_css.loc[df_intra_css['patent_id'] == patents[i]].
    ↪ drop_duplicates()['citing_patent'])
    cit_set = set(df_intra_cit.loc[df_intra_cit['patent_id'] == patents[i]].
    ↪ drop_duplicates()['citing_patent'])

    c = venn2([css_set, cit_set],
              (str(patents[i]) + '_' + 'css',
               str(patents[i]) + '_' + 'cit'),
               ax=ax)

    c.get_patch_by_id('10').set_color('C0')
    c.get_patch_by_id('01').set_color('C1')
    c.get_patch_by_id('11').set_color('C5')
    c.get_patch_by_id('10').set_edgecolor('none')
    c.get_patch_by_id('01').set_edgecolor('none')
    c.get_patch_by_id('11').set_edgecolor('none')
    c.get_patch_by_id('10').set_alpha(0.6)
    c.get_patch_by_id('01').set_alpha(0.6)
    c.get_patch_by_id('11').set_alpha(0.6)

    for text in c.set_labels:
        text.set_fontsize(10)
    for text in c.subset_labels:
        text.set_fontsize(12)

    plt.close(2)

plt.tight_layout()

plt.savefig(working + path + ('img/' + time.strftime("%Y%m%d-%H%M%S")
+ '_') + filename + '_intersection_venn_relationships.png',
bbox_inches='tight', dpi=300)


# ==============================================================================
# venn2 with labels
# ==============================================================================
#patent_intersection_no = '3935021'
patent_intersection_no = '9303363'

C1 = set(df_intra_css.loc[df_intra_css['patent_id'] == patent_intersection_no].
    ↪ drop_duplicates()['citing_patent'])
C2 = set(df_intra_cit.loc[df_intra_cit['patent_id'] == patent_intersection_no].
    ↪ drop_duplicates()['citing_patent'])

sets = [C1, C2]
setLabels = ['CSS', 'Cited']

fig = plt.figure(figsize=(20,10))
ax = plt.gca()
v = venn2(subsets = sets, set_labels = setLabels, ax = ax)

v.get_patch_by_id('10').set_color('C0')
v.get_patch_by_id('01').set_color('C1')
v.get_patch_by_id('11').set_color('C5')

v.get_patch_by_id('10').set_edgecolor('none')
```

Appendix P (Continued)

```
v.get_patch_by_id('01').set_edgecolor('none')
v.get_patch_by_id('11').set_edgecolor('none')

v.get_patch_by_id('10').set_alpha(0.6)
v.get_patch_by_id('01').set_alpha(0.6)
v.get_patch_by_id('11').set_alpha(0.6)

v.get_label_by_id('10').set_text('\n'.join(sorted(C1-C2)))
v.get_label_by_id('01').set_text('\n'.join(sorted(C2-C1)))
v.get_label_by_id('11').set_text('\n'.join(sorted(C1&C2)))

A = len(C1.difference(C2))
B = len(C1.intersection(C2))
C = len(C2.difference(C1))

h, l = [],[]
h.append(v.get_patch_by_id('10'))
h.append(v.get_patch_by_id('11'))
h.append(v.get_patch_by_id('01'))

l.append(' ' + setLabels[0] + '-related' + ' ' + '(' + str(A) + ')')
l.append(' ' + 'Intersection' + ' ' + '(' + str(B) + ')')
l.append(' ' + setLabels[1] + ' ' + '(' + str(C) + ')')

for text in v.set_labels:
    text.set_fontsize(17)
for text in v.subset_labels:
    text.set_fontsize(9)

#create legend from handles and labels
ax.legend(handles=h, labels=l, fontsize=14, loc=3)

plt.title('Method Comparison Venn Diagram: Patent US' + str(
    ↪ patent_intersection_no), fontsize=20)

plt.tight_layout()

plt.savefig(working + path + (
        'img/' + time.strftime("%Y%m%d-%H%M%S") + '_'
        ) + filename + '_cssvcit_patent_' + patent_intersection_no + '_venn2.png
    ↪ ',
            bbox_inches='tight',dpi=300)

dir(v)
```

Appendix Q

ORIGINAL CITATION VISUALIZATION

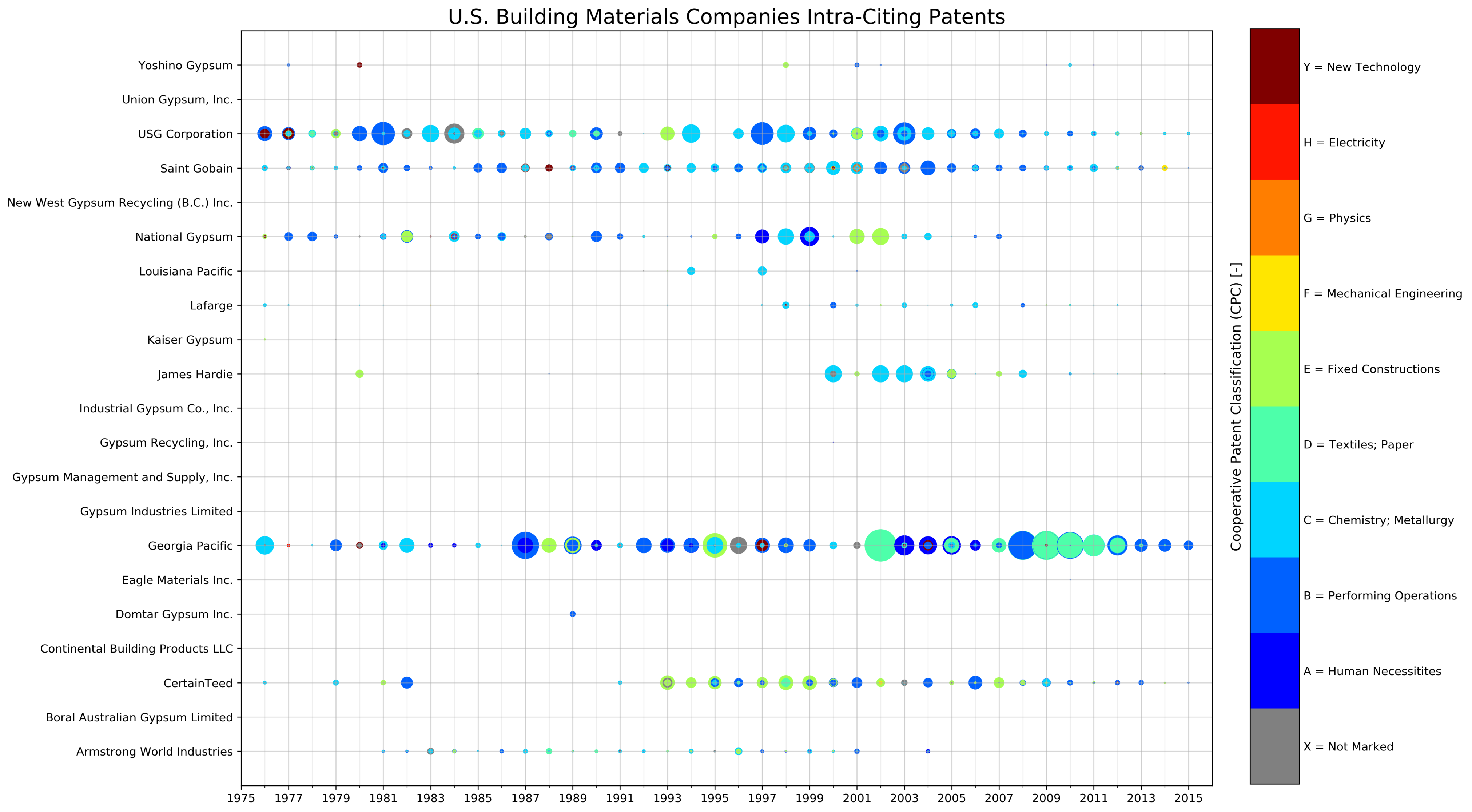


Figure 31: U.S. Building Material Patents: Invention Impact Using Intra-CIT Similarity Threshold (1975 - 2015)