Opinion Mining for Decision Making

BY

PAOLO POLIMENO CAMASTRA
B.S., Politecnico di Milano, Milan, Italy, 2016

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2019

Chicago, Illinois

ACKNOWLEDGMENTS

I would like to thank my Advisor, Professor Barbara Di Eugenio, for her support and guidance. I would like to thank Professor Pier Luca Lanzi from Politecnico di Milano, my colleagues Mohammed Elshendy and Andrea Lui for their help and feedback on this work.
I would like to thank my family, my friends, all my course mates and colleagues.

PPC

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
SVD	Singular Value Decomposition
BSS	Between Clusters Sum of Squares
WSS	Within Clusters Sum of Squares
OM	Opinion Mining
SA	Sentiment Analysis
NN	Neural Network
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
BOW	Bag of Words
W2V	Word2Vec
LDA	Latent Dirichlet Allocation
SVD	Singular Value Decomposition
AMI	Adjusted Mutual Information

SUMMARY

This work is concerned with Text Clustering, Sentiment Classification and Transfer Learning in Sentiment Classification. We choose to work in the domain of online reviews because they are suitable for our purposes: the dataset comprises reviews' texts and labels as well as various categories of products. Labels are needed in order to perform the Supervised Learning part of our work, i.e. Sentiment Classification, while the presence of different product categories enable us to have different domains, such as Electronics, Movies, Clothing, etc., which is a requirement for Transfer Learning. Our main contribution in Text Clustering is in terms of comparison between clusterings obtained after different text vectorizations: tf-idf+Singular Value Decomposition (SVD), LDA and word2vec.

CHAPTER 1

INTRODUCTION

Whether we are speaking about products, businesses, professionals or any entity which can be subject to evaluation, others' opinions are always an important factor in making decisions on what to choose or buy. During the dot-com bubble of 1999 various companies started to appear which allowed their users to upload reviews on anything ranging from movies to sports teams. The significant role played by online reviews nowadays, as well as the fact that they come labeled with sentiment score, prompted us to choose them as the main tool for our work.

1.1 Purpose

In this work we set ourselves to deal with the problem of Transfer Learning in Sentiment Analysis (SA). The problem is of particular interest given that SA can be applied on a variety of sources (e.g. social media, articles, blogs etc.), the majority of which come unlabeled and cannot be exploited without costly and/or time consuming hand-labeling process.

The problem has been proposed before, nonetheless we believe our attempt deserves attention for multiple reasons: first of all, the dimension of the dataset used for pre-training the model in terms of number of instances. The dataset we use is far larger than most datasets used in the context of sentiment analysis (10^5 vs 10^4).

We consider 5-class classification, instead of transforming our target to binary form (positive-negative): although in some cases the original labels are kept, in most papers they divide reviews in positive and negative, by removing 3-stars labelled reviews, such as in [1]. We are then left with an unbalanced distribution of the target variable: 5-star reviews are the most present, followed by 1-star ones, while 2nd, 3rd and 4th class are substantially underrepresented.

Given the nature of the task (multiclass), we take particular care in selecting a measure which is representative not only of overall performances but also of performances on every single class. Therefore we will not choose accuracy or a weighted version of F1, but we will consider F1 calculated for every single class.

Last, our approach is wide both in the choice of the data, in the sense that we consider more than one target domain, where by domain we intend a product category, and in the choice of the models, in that we compare standard ML and DL models to assess their performances both on Single-Domain and on Transfer Learning Sentiment Analysis.

While working on sentiment Analysis on product reviews, we came across the idea of removing noise from reviews, where with noise we mean those reviews which are intended to alter the rating of a product for the purpose of either increasing (hyper spam) or decreasing (defaming spam) its value to the eyes of a potential customer.

As a starting point we decided to cluster product names, in order to obtain smaller groups of products and, consequently, of reviews on which to apply spam detection algorithms. This step is based on the assumption that a spammer/a group of spammers operates on a certain group of similar products, a sensible assumption when dealing with hyper spam and defaming spam, both of which are intended to respectively unduly praise or discredit a certain product with respect to main competitors.

We soon reckoned that the problem of detecting spam reviews is a very difficult one, the main obstacle being that if a spam review is well crafted it can look exactly like a legitimate one and also for a human it is difficult to build a dataset of fake and legitimate reviews [2]. For the above reason, we decided to set aside the problem of recognizing spam reviews and to continue focusing on the Text Clustering field. The focus of this part will be the comparison of various vectorization flows in terms of clusters of documents obtained.

Outline

In the first chapter we introduce our work, we discuss its purpose and the various tasks we are going to work on.

In the second chapter we will present the NLP and Machine Learning technologies which we are going to use in our work.

In the third chapter we will discuss previous work in the fields of Document Clustering and Opinion Mining.

In the fourth chapter we will describe the dataset we will work with.

In the fifth chapter we will deal with Document Clustering.

In the sixth chapter we will deal with Opinion Polarity Analysis.

In the seventh chapter we will discuss Conclusions and Future Works.

CHAPTER 2

TEXT PROCESSING TECHNOLOGIES

In the previous chapter we discussed the problems we are going to address in our work. In both problems we will be dealing with plain English text and we will be using the tools which have been developed in the field of NLP since its beginnings in the 50s to these days with the advent of Deep Learning.

2.1 Vectorial representation of text

In order for algorithms to be able to process text, various representations have been developed. The ones we take into consideration map either the whole document or single words or n-grams to a Vectorial Space. Below we illustrate the representations which will be used in our work.

2.1.1 Bag of words model

This model can be found as early as 1954 in [3]. If we are working on a set of documents D, we will be able, after a preprocessing phase, to extract N unique tokens. Each document is represented as a vector, whose length is the same as the number of words of the vocabulary extracted from the corpus, that is to say: N. The entries of the vector represent the frequency of each vocabulary term in the document.

For example, if we assume the following ordered vocabulary:

$$\{'I','you','like','hate','pizza','pasta','and'\}$$

and we consider the following document:

$$I\,like\,pizza\,and\,I\,like\,pasta$$

we can represent the document as the vector:

$$[1, 0, 2, 0, 1, 1, 1]$$

This representation does not capture word order in the document, but it keeps all the information about words and their frequencies. However, for certain applications, e.g. documents classification, it would be useful to have a measure of how relevant a term is.

2.1.2 Tf-idf representation

The tf-idf representation was introduced by Karen Jones in [4]. A document is defined as a vector of length equals to vocabulary length, just like in BOW model. What changes is that the values in the vector are not the frequency of each term in the document, but they are given by the following function which takes as input the document d, the term t and the set of all documents D:

$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$

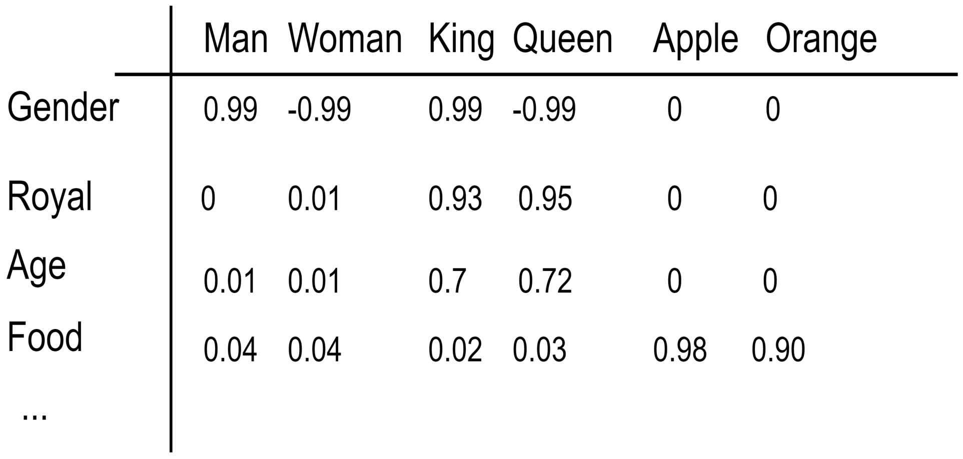
The first product term represents the term frequency used in BOW model. In tf-idf representation it is weighted by the Inverse Document Frequency term, which tells us whether a term is present in a lot of other documents of the corpus and thus is not very relevant or if it is in just a few/in only the one considered and thus is very relevant. Idf is defined as follows:

$$idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

2.1.3 Word embeddings

In the BOW representation each word (feature) has its own dimension and therefore what we have learned for a specific word, e.g. "orange", is not transferred to semantically close words, e.g. "lemon".

It would be better if we managed to build word representations which captured semantic similarity, so that our models could generalize more easily and would need a smaller training set. We would like our model to recognize that substituting e.g. the word "orange" with the word "lemon" is very different from substituting it with the word "president". Even if our model has not seen the words "lemon" and "president" in the training set, the knowledge embedded into the word vectors, which has been acquired on a very large training corpus, will allow it to exploit the similarity between the vector representations of "orange" and "lemon" and the distance between the representations of "orange" and "president". What we would like to get is something similar to Figure 1, that is to say a vector representation where the dimensions do not correspond to single words in the vocabulary, but rather to "concepts". Even if we cannot decide which concepts our model learns, we would like to have vectors whose cosine distance is smaller the more they represent semantically similar words.



	Man	Woman	King	Queen	Apple	Orange
Gender	0.99	-0.99	0.99	-0.99	0	0
Royal	0	0.01	0.93	0.95	0	0
Age	0.01	0.01	0.7	0.72	0	0
Food	0.04	0.04	0.02	0.03	0.98	0.90
...						

Figure 1: Semantic representation of words

It turns out that performing tasks such as the one depicted in Figure 2, that is to say predicting a word given a context, is really useful for obtaining vectors with the above mentioned characteristics.

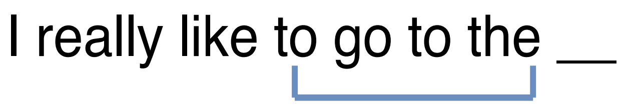


Figure 2: Predict word from context

In particular in 2013 [5] Mikolov and his team at Google proposed two algorithms that are able to create word embeddings efficiently and drawing from very large text dataset. Their approach is known as Word2vec. Word2vec embeddings have been one of the most popular in research, but other embeddings have emerged such as FastText [6] or GloVe [7].

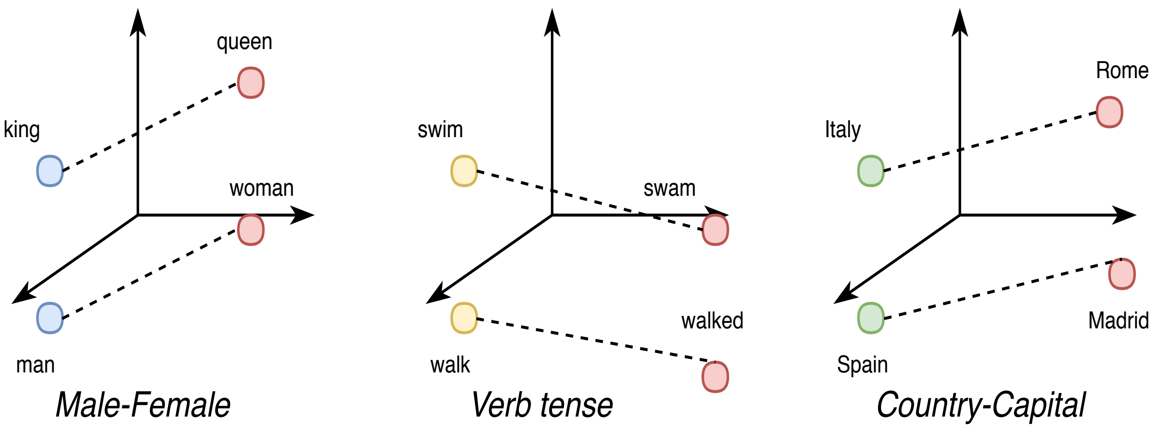


Figure 3: Semantic relationships between word2Vec vectors

In Figure 3 we can see examples of word2vec-like representations of different relationships which cannot be captured by classic vectorial representations, such as BOW, but are indeed very well captured by word2vec and similar word embeddings.

2.1.3.1 Main algorithms

Below we give a brief description of three algorithms which are extensively used in statistical NLP and that we will use as baselines in our experiments.

Naive Bayes: Naive Bayes, as the name suggests is based on Bayes' Theorem (Bayes) and on the assumption of independence of features (Naive).

$$\mathbf{x} = (x_1, ..., x_{\mathrm{n}}) \qquad \text{feature vector}$$

$$p(C_{\mathrm{k}}|\mathbf{x}) \qquad \text{for each possible class } \mathrm{C_k}$$

The above probabilities are difficult to calculate from a corpus, therefore we use Bayes' Theorem to obtain a better formulation:

$$p(C_{\mathrm{k}}|\mathbf{x}) = \frac{p(C_{\mathrm{k}})p(\mathbf{x}|C_{\mathrm{k}})}{p(\mathbf{x})}$$

We notice that the numerator is equal to the joint probability

$$p(x_1, ..., x_{\mathrm{n}}, C_{\mathrm{k}})$$

which, with chain rule and independence assumption, can be rewritten as

$$p(C_{\mathrm{k}}) \prod_{i=1}^{n} p(x_{\mathrm{i}}|C_{\mathrm{k}})$$

We can estimate both factors from the corpus and we can classify the sample by finding the class that maximizes the above expression.

SVM: Given a training dataset:

$$(\mathbf{x}_1, y_1), ..., (\mathbf{x}_\mathrm{n}, y_\mathrm{n})$$

where x_i are data points and y_i are the respective labels. SVM is a linear classifier which tries to find the "maximum-margin hyperplane" and then classify the new points based on which sides of the hyperplane they fall. A hyperplane is the set of points which satisfies:

$$\mathbf{x} * \mathbf{w} - b = 0$$

where w are weights which must be trained and b is a "bias", that is to say a parameter which is to be trained and is not associated with any input. If the classes are linearly separable, it is possible to draw two parallel hyperplanes that separate the points of the two classes and are as far as possible. The parallel hyperplane which is halfway in between them is the "maximum-margin". If the points are not linearly separable we can still classify them with SVM, but before we need to project them onto a higher dimension hyperplane, a procedure known as "kernel trick".

2.2 Deep learning architectures for NLP

We start by describing the main DL architectures which are currently used in the NLP field.

2.2.1 Neural networks

Artificial Neural Networks or, simply, Neural Networks, are formed by layers of Artificial Neurons. Each layer take as inputs the outputs of the previous one, apart from the first layer

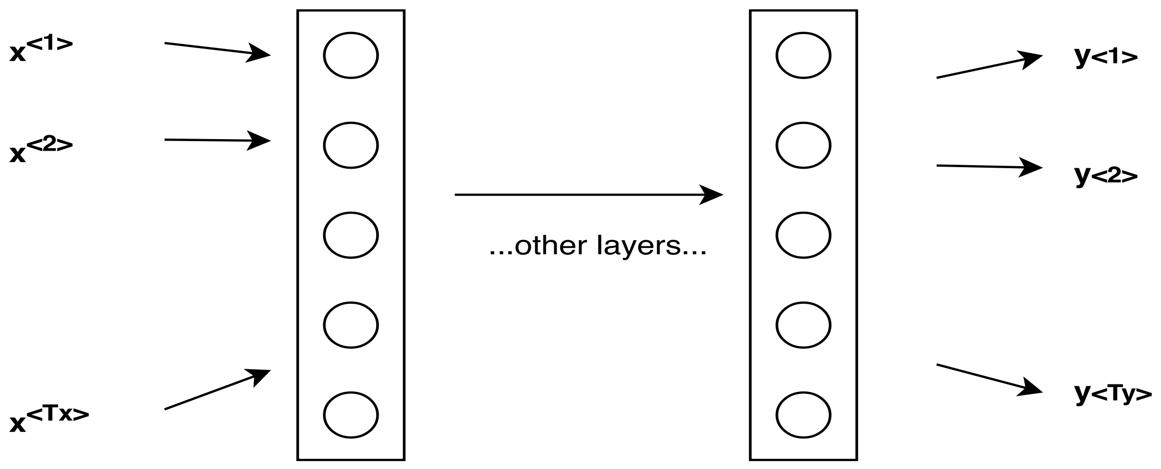


Figure 4: A sample multioutput NN

which takes a data point as input. Artificial neurons are inspired by biological one, in the sense that they receive inputs from other Artificial neurons and their output is a function of the inputs received.

Let's see the mathematical expression which defines an Artificial Neuron:

$$y = \phi(\sum_{j=0}^{m} w_j x_j)$$

Specifically:

- m is the number of inputs
- x_js are the different inputs
- w_js are the coefficients (weights) of the linear combination of the inputs which is fed to ϕ

- x_0 is a special input because it is always equal to 1 and its corresponding weight w_0 is also indicated as b_0 and is named bias
- ϕ is a nonlinear function of the above mentioned linear combination of inputs, $\phi : \mathbb{R} \rightarrow \mathbb{R}$
- y is the output of the neuron

A single Artificial Neuron cannot perform nonlinear classification, however a Neural Network can. Variation of standard Neural Network architecture have been extremely successful at certain tasks, such as Image Classification or Speech Recognition. We can see a standard NN in Figure 4.

2.2.2 Recurrent neural networks

Recurrent Neural Networks (RNNs) are a particular kind of Neural Network (NN) which was introduced in the 80s and they are particularly suited for modeling temporal dependencies. RNNs are not so different from feed-forward NN. We can see a RNN in Figure 5. In both cases we have a Tx input vectors and Ty output vectors and we assume $Tx = Ty$. While the standard NN takes in input all the word vectors, which we can assume to be word in a document, at the same time, the RNN is basically a succession of NNs which takes in input only two vectors: the hidden state vector of the previous time step $a < t - 1 >$ and the word vector of current time step $x < t >$. We list the two main advantages of using RNN over NN:

- RNN uses the same set of weights at every time step, therefore we can use very few weights with respect to a NN, which must have weights for each word of the longest input sequence.

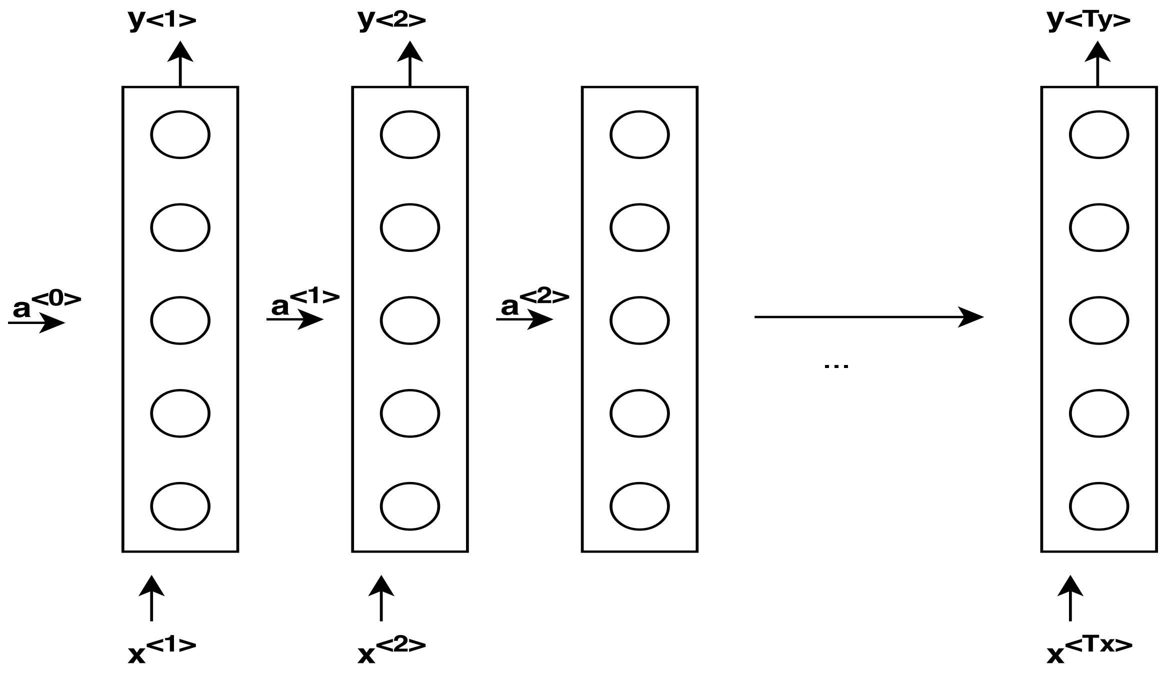


Figure 5: A sample multioutput RNN

- With RNN we can have sequences as long as we desire, without having to fix length in advance.
- With an NN the first word and e.g. the fourth word in a sequence are two completely different feature and NN cannot transfer something it learnt for the first word to the fourth word in a sequence. On the contrary RNN, by sharing weights, is able to capture pattern also if they appear in different positions.

As we have seen RNN is nothing more than a series of NNs and therefore we can train it with standard backpropagation, after having unrolled it for the number of time-steps of the input sequence. Unfortunately, an unfolded RNN can be a rather deep NN and as in all deep NN we encounter the exploding and vanishing gradients problem, that is to say a numerical impossibility of propagating the weight update needed to the first layers of the network during backpropagation.

This weak point in the implementation has been addressed with architectural changes. In particular, we will look at two of the variations which have been developed by researchers: Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) unit.

2.2.3 GRUs and LSTMs

In this section we give an overview of a slightly simplified version of a GRU. Both GRUs and LSTMs are based on the same principle of adding so-called "gates". Gates are functions which are in charge of changing the way the hidden status is updated. In this way we allow some information which is in the hidden state, i.e. some of the values of the vector, to be partially or even completely unmodified with respect to what would have happened without it.

By doing so we are able to mitigate the problem of vanishing gradients and we are able to obtain a network which captures long-term dependencies.

As we said, in Figure 7 we can see a simplified GRU, which has a single gate. LSTM was the first of this kind of architecture to be proposed [8] and we may be interested in discussing its structure or also the structure of the more recent GRU. Actually, there is no need to go into such details, since all these architectures are variations which have been developed by researchers in order to avoid vanishing gradients problem and have emerged as the best one. Still, the principle of filtering the hidden state with gate function is the main idea and it underlies all of them. LSTM was the first architecture to be proposed and presents 3 gates, which enable the network not to lose useful information of the first time steps. GRU is a simpler architecture, which was presented in 2014 [9] and presents 2 gates. An interesting point is made in [10] as to how much context LSTMs are really able to capture. In the paper, the authors use LSTM for the task of predicting the next word in a sentence. They consider different window sizes: if we want to predict the $n_t h$ word in a sentence we can predict it given the $(n-1)_t h$ word, given the $(n-1)_t h$ and the $(n-2)_t h$ and so on.

From the analysis it emerges that increases in window size after 200 time steps affect very little LSTMs performance and, furthermore, words order is only captured in the most 50 recent tokens.

2.2.4 Convolutional neural networks in NLP

Convolutional Neural Networks (CNNs) are well known in Machine Learning community. In particular, the Imagenet challenge results accomplished by Krizhevsky, Sutskever and Hinton [11] are considered to be the start of Deep Learning revolution. CNNs are widely used in Image Recognition and they are based on the concept of Convolution: a filter, Figure 8, slides over the image matrix and takes the dot product between the network's ordered weights and the ordered pixels of the image portion it is on at time t. Filter weights are shared, i.e. they do not change when the filter slides over the image, so that each filter is able to learn a specific feature and it can recognize it in every section of the image.

Each layer of the CNN comprises several filters and each of those is able to learn a specific feature. In CNNs we speak of "compositionality": going from one layer to the next one, features become more complex since they are obtained as a combination of the previous layer's features. As an example, given an image, the first layer of the network takes as input raw pixels. Pixels get combined by the filters: during the training phase, each filter learns parameters which enable it to detect simple patterns, e.g. vertical lines, horizontal lines, objects' edges. The second layer of the network takes as input the patterns learned by the first layer and combines them, yielding more complex patterns such as lines meeting at specific angles, specific shapes, etc. The third layer aggregates second layer's features and detects details in images: if, as an example, the image depicts a car, each filter of the third layer will detect either wheels or windows or fenders, etc.

CNNs also presents another kind of layer, the Pooling Layer, which makes the network robust with respect to translation and rotation. It achieves this by downsampling the image, that is to say by extracting a single value (usually average or maximum) from every group of adjacent pixels.

In NLP, Figure 9, instead of considering images, we consider the matrix obtained by juxtaposing the word vectors. Filters are a way to capture combinations of words, syntax and semantics.

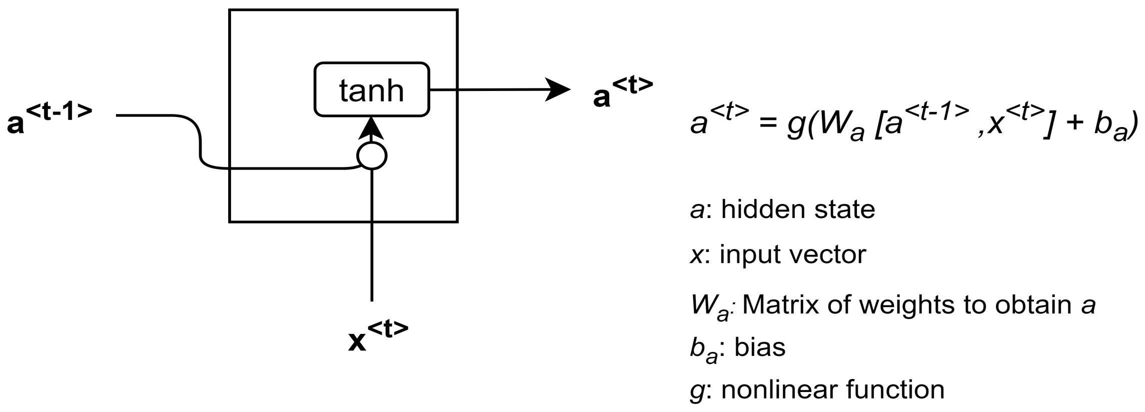


Figure 6: A sample RNN cell

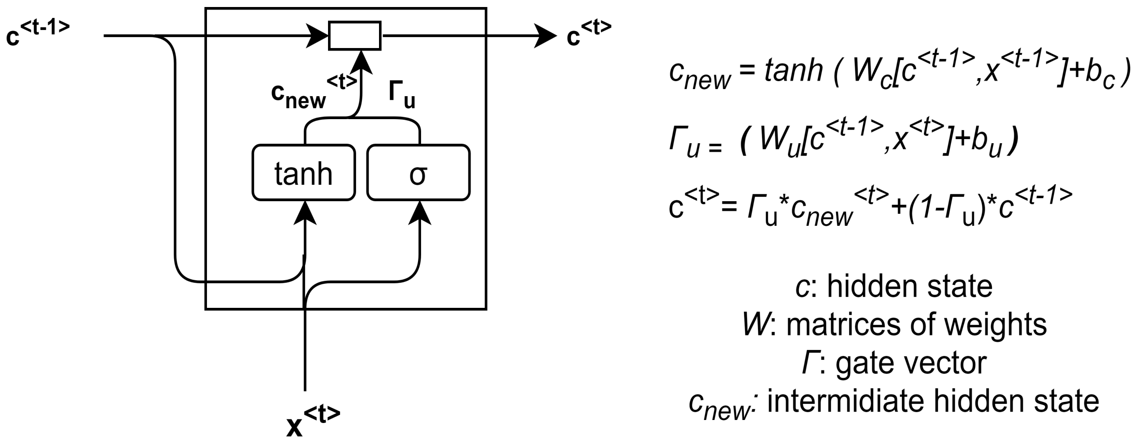

$c^{<t-1>}$
$c^{<t>}$
$c_{new}^{<t>}$
Γ_u
tanh
σ
$x^{<t>}$
$c_{new} = tanh\,(W_c[c^{<t-1>}, x^{<t-1>}]+b_c)$
$\Gamma_u = \ (W_u[c^{<t-1>}, x^{<t>}]+b_u)$
$c^{<t>} = \Gamma_u * c_{new}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$
c: hidden state
W: matrices of weights
Γ: gate vector
c_{new}: intermidiate hidden state

Figure 7: Simple GRU

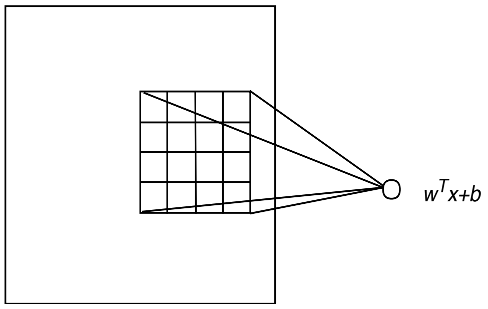


Figure 8: Example of CNN filter

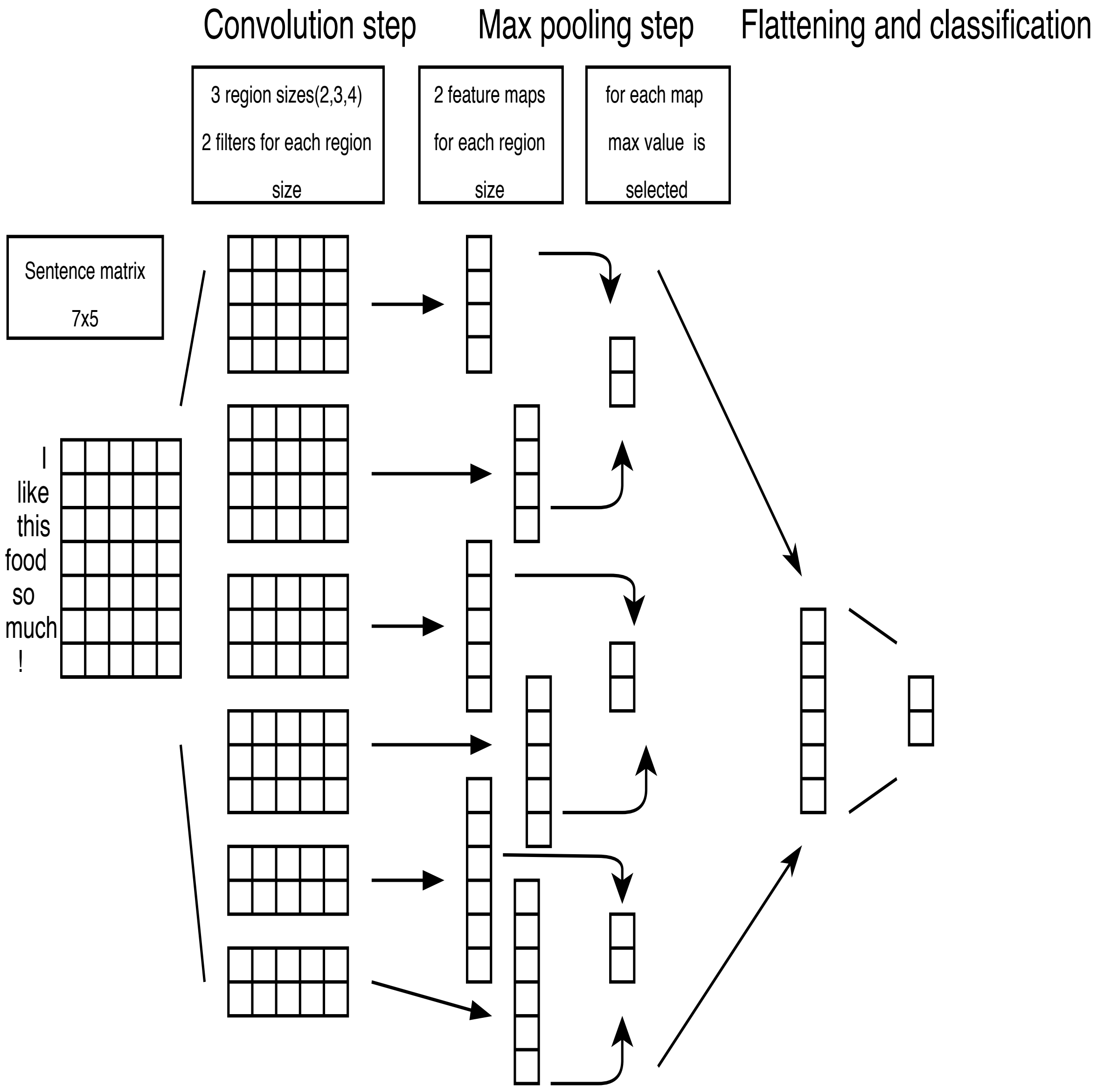

Convolution step
Max pooling step
Flattening and classification
3 region sizes(2,3,4)
2 filters for each region size
2 feature maps for each region size
for each map max value is selected
Sentence matrix 7x5
I
like
this
food
so
much
!

Figure 9: Example of CNN applied to NLP

CHAPTER 3

LITERATURE REVIEW

3.1 Text clustering

In this section we introduce techniques which we will apply in the Text Clustering section of our work, as well as some use cases which can be found in the literature.

3.1.1 Clustering

Clustering is an *unsupervised* technique which aims to discover patterns in data, in particular to assign each data point to a group, i.e. a "cluster". It is *unsupervised* because data has no label to learn from and to be used for evaluation purposes. Clusters should represent a group of points which are more similar to each other than to the points which are not in their cluster, based on some similarity measure of choice.

3.1.2 Singular value decomposition (SVD)

SVD gives us a specific k-dimensional subspace which is contained in the original vectorial space. Specifically, SVD gives us the k-dimensional subspace which minimizes the mean projection square error and explains the most variance with respect to all the other k-dimensional subspaces.

3.1.3 Latent dirichlet allocation (LDA)

LDA [12] is a generative model, which is used for topic modeling, that is to say to "discover" which latent topics are hidden in a collection of documents. LDA assumes a fixed number of latent topics and describes each document as a distribution over topics and each topic as a distribution over words. Topics are not predefined but are discovered through inference. By using Bayesian inference we are able to see which words are most associated to a topic and which topics are most present in each document.

3.1.4 Text clusterings examples

In [13] the authors compare various text clustering approaches: K-means, Spectral Clustering and Convolutional Neural Networks. Convolutional Networks were used in the following way: the CNN was trained to predict a reduced vector representation of the document, starting from the original document. After the training was completed, the hidden state of the network when a document is fed becomes the new document representation and K-means clustering is applied. The authors cluster 2 labeled datasets of around 10^4 samples and they evaluate obtained clusterings in terms of Accuracy and Normalized Mutual Information. Both metrics are calculated with respect to the true labels.

In [14] the authors evaluate clusterings obtained after a process of feature selection with Precision. Precision labels a cluster with the label of the class which is found to be more present and gives the ratio between the number of samples from majority class and cluster's cardinality.

Again, in [15] authors consider Accuracy and NMI with respect to true labels as measures for Clustering Quality. They also evaluate the consistency of the clusterings, by running each method 30 times and giving a measure of how much assignment varies: they consider each couple of runs. For each couple they perform 1-1 matching of clusters from run 1 to clusters from run 2, by similarity, and then they calculate how many samples are not in the same clusters.

3.2 Opinion mining

Opinion Mining does not focus on the topic of the document, instead, it focuses on the opinion expressed in it.

3.2.1 Opinion base components

According to [16], main components of an opinion are:

- Opinion holder: the person/organization which holds a specific opinion on a specific object.
- Opinion object: what the opinion is about.
- Opinion: a statement regarding an object coming from the opinion holder.

3.2.2 Main tasks

Again, according to [16], these are the main tasks which are performed in Opinion Mining:

- Determining subjectivity: it consists in determining whether an opinion is objective or it expresses a personal view. It takes the form of a binary classification task.
- Determining the orientation (polarity): it consists of determining whether a *subjective* text expresses a positive or negative opinion.
- Determining the strength of orientation: it consists in determining how much a *positive* or *negative* opinion is strong.

Among the above mentioned three tasks, we decided to focus on task two and task three, which are known as Opinion Polarity Analysis.

3.2.3 Opinion polarity analysis

Opinion Polarity Analysis, also known in the industry as Sentiment Analysis, is usually framed as a classification problem. In its basic version, it is a binary classification (Positive and Negative), while in our case it becomes a multiclass classification problem since we have 5 different possible scores.
We will deal with Document Level Classification: this type of classification considers a document as a basic unit and tries to classify it. Below we give an overview of main approaches which have been tried by researchers.

3.2.4 Lexicon-based approach

This rather simple approach is based on counting how many positive and negative words are present in a text and classifying the document accordingly.

3.2.4.1 Dictionary-based approach

This approach is adopted, for example in [17]. The idea is very simple: compile a list of words with positive and negative orientation. Since positive or negative connotation is not something we can usually find in dictionaries, we need a strategy to collect the needed list. One idea, which is detailed in [18] is to use mutual information: you perform a web search on a term and you calculate the mutual information with a very positive term such as "excellent" and a very negative term such as "poor". The disadvantage lies in the number of web searches and operations to be performed.
A simpler idea, which is the one described in [17] is to exploit the similarity/antonymy information contained in a synonym/antonym dictionary(e.g. WordNet): you start from a list of

seeds for which orientation is given and you iterate over all adjectives. If they have a synonym in positive/negative seed list you add them to that list, while if they have an antonym in positive/negative seed list you add them to the opposite list. Of course, some of the words will not fall in either one or the other category, but if we run the procedure again they may, given that the size of the lists has increased. You keep running iteratively until the lists remain the same. Dictionary approach is not able to catch context-dependent orientation of words.

3.2.4.2 Corpus-based approach

In the corpus-based approach, starting from a seed list we want to extract other sentiment words from a specific corpus. One idea is explained in [19]: they designed a series of rules to predict orientation of words which is based on the fact that words joined by certain conjuction usually have either the same or different orientation. One example is with the 'and' conjunction: if a word is linked to another by 'and' their polarities are more likely to be the same. A graph is formed, where each couple of words appearing in the corpus joined by a conjunction is linked by a weighted graph edge (weight is +1 for "agreement" conjunction and -1 for "disagreement" conjunction). Then they use an iterative approach to split the set of words into two clusters so as to maximize the sum of edges weight of points in the same cluster.

3.3 Machine learning-based approach

3.3.1 Supervised learning

The first paper to use the supervised machine learning approach to classify reviews was [20]. Researchers did not rely on prior knowledge and they let models free of selecting the most discriminant features for the task. They worked with movie reviews, which they selected because of the large availability and the lack of need to hand-label data. The authors show how the results of a classification based on words count from positive and negative lists vary considerably depending on which words are put in the lists. They then resorted to classic supervised learning algorithm for topic classification and they obtained good results using unigrams as features.

3.3.1.1 Example features

As we can find in [21], different kind of features are commonly used in sentiment classification:

1. **Words and Word Frequencies**: these are the most important features for classic ML models and are captured by the BOW representation. TF-IDF can also be applied.
2. **Word Embeddings**: these are the most important features for DL models, as we have seen in the previous chapter.
3. **POS and Syntactic Features**: these features, although more generic, can help in weighing more important words (e.g. adjectives) or in capturing peculiar sentences structure.
4. **Sentiment shifters**: whether a word comes after a negation can completely reverse the meaning of a sentence.

3.4 Transfer learning

Transfer Learning in Machine Learning context refers to the problem of reusing knowledge gained while solving a certain problem to solve a different but related problem. In [22] Domain Adaptation for multiclass classification, a form of Transfer Learning, is detailed: while in standard classification task we assume both Training and Test data $D = (x_i, y_i) \in X \times Y$ to be drawn from the same distribution p, in Domain Adaptation setting, we are presented with training data drawn from two different distributions $p1$ and $p2$ and we need to classify data extracted from $p2$, having $N2 << N1$, with $N1$ and $N2$ being the number of training examples we have for $p1$ and $p2$ respectively.

In our case we deal with a specific kind of Transfer Learning, known as "Cross-Domain" Sentiment Classification. In our case, as we already said, the different domains will be the different product categories. More generally, a domain can be defined either by the content which are discussed in a text, like in our case, or by the way people express themselves in a text, e.g. Social Networks, Amazon reviews, poetry etc. According to [21] Sentiment Classification has been shown to be highly sensitive to the domain of the training data, both because in different domains there are different ways to express opinions and because some words have different meaning and connotation in different domains.

In [23] Aue and Gamon, from Microsoft Research, focus on transfer learning for two-class sentiment classification. They make use of SVMs in all experiments but one, where they use Naive Bayes, and they experiment with various configurations of training sets. They work with four domains: Books, Movies and two user feedbacks datasets.

In Table I we see the number of samples used in the paper we are discussing and we can see that dataset sizes are of the order 10^4.

In experiment 1 the authors train 4 classifiers for each of the domains and then test them on all domains. In experiment 2 the authors also train 4 classifiers: each classifier is trained on a training set comprising equal number of samples from 3 domains and is tested on the remaining

TABLE I: LABELS AND CLASSES DISTRIBUTION

	Positive	Negative	Total
Books	1000	1000	2000
Movies	1000	1000	2000
Survey_1	2564	2371	4935
Survey_2	6035	6285	12320

TABLE II: RESULTS OF 1ST AND 2ND EXPERIMENT

Accuracy			
	Classifier trained on all domains (test domain excluded)	Best performing classifier trained on single domain (test domain excluded)	Domain of best performing single domain classifier
Movies	72.89	72.08	Books
Book	64.58	70.29	Movie
Survey_1	63.92	70.48	Survey_2
Survey_2	74.88	81.42	Survey_1

domain.

It is interesting to compare performances obtained in Experiment 2 with best performances obtained in 1.

Table II shows us that best training domain (different from original), is better or almost equal (Movies) than the classifier trained on multiple domain. It seems that the most similar domain is the one which drives performances.

In [24] they work on a 22 domains, binary-label dataset of 340.000 Amazon reviews. First they extract features in an unsupervised manner, using Stacked Denoising Autoencoders, which is a DL architecture. Autoencoder works in the following way: a Neural Network (Encoder) encodes the input into a lower-dimensional space. Another Neural Network (Decoder) takes as input the lower-dimensional vector and outputs a vector in the original vector space. Both networks are trained together to reconstruct original input with minimum loss. Stacked Denoising Autoencoders are a series of Autoencoders, where the output of one Autoencoder is used as input for the next one. Also, noise is added to the input, making the process of reconstruction more difficult. After the unsupervised learning phase, which is performed on all domains, they train an SVM on a domain and evaluate both on the training domain itself and on all the others. For the full datasets results for each model are given as the average of the Transfer Loss considering all possible couples of Source and Target domains. Transfer Loss is defined as:

$$t(S,T) = e(S,T) - e_b(S,T)$$

Where e is the error of the considered model and e_b is the error of the baseline. The main authors' finding is that by performing the unsupervised phase and using as input the representations obtained with the Autoencoder, SVM performances as measured by Transfer Loss, increase significantly, both with respect to SVM and Multi Layer Perceptron trained only on original input.

CHAPTER 4

DATA STRUCTURE

In this section we describe the structure of Amazon's reviews and the corresponding structure of the chosen dataset. Furthermore, we give dataset statistics which we computed for the period of interest.

The dataset [25] has been collected by Julian McAuley, currently Assistant Professor at UCSD.

4.1 Amazon.com

Amazon.com, inc. was founded on July 5, 1994. In terms of revenue and market capitalization it is the largest online retailer in the world, while in terms of total sales it is second only to Alibaba Group. According to eMarketer.com its US market share of e-commerce is 49.1% followed by eBay with 6.6%. This makes it by far the most important e-commerce website for the US market and, on top of that, Amazon.com shopping makes up 5% of total US retail.

4.2 Amazon review structure

As we can see from Figure 10, Amazon review has different components:

1. review title
2. review text
3. star rating, which is an integer from 1 to 5
4. reviewer's username
5. number of people who found the review helpful

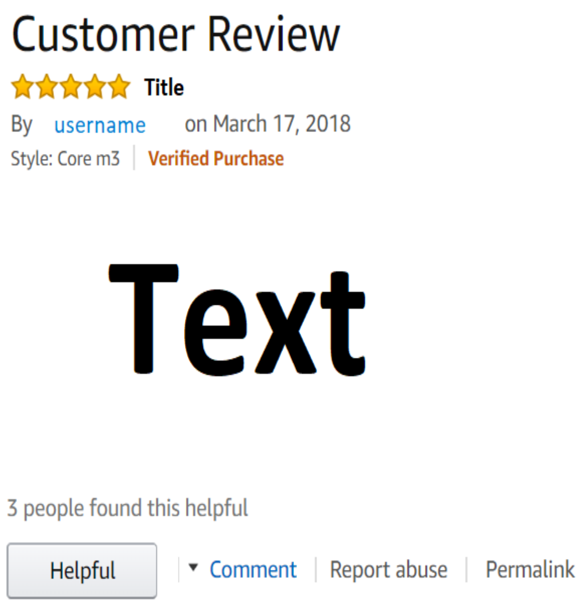


Figure 10 Sample Amazon.com review

6. review time

Amazon reviews previously featured a count of how many people found the review unhelpful, but it has been recently removed.

4.3 Dataset structure

Here we give a look at the main features of the dataset:

1. 82.83 million reviews
2. spanning May 1996 - July 2014
3. 24 product categories

```
['case', 221836],                ['great', 228176],
['cover', 134032],               ['one', 197438],
['black', 118281],               ['use', 192429],
['samsung', 90951],              ['works', 191386],
['hard', 77407],                 ['good', 179933],
['iphone', 74979],               ['would', 174754],
['g', 73711],                    ['product', 163912],
['protector', 67790],            ['well', 148498],
['galaxy', 64506],               ['price', 141582],
['screen', 63284],               ['like', 140358],
['amp', 59128],                  ['work', 129791],
['battery', 48959],              ['get', 123671],
['skin', 47437],                 ['bought', 122939],
['x', 46957],                    ['time', 109157],
['charger', 46226],              ['easy', 103446],
['usb', 45900],                  ['buy', 98069],
['apple', 45355],                ['nook', 97484],
['phone', 43094],                ['used', 97384],
['cable', 41078],                ['cable', 96088],
['inch', 38183]                  ['quality', 87455]
```

Figure 11 Common words and their number of appearence, on the left we see common words in product names, on the right common words in products' reviews

We will focus our attention mainly on "Electronics" and "Cell Phones and Accessories", furthermore we will take into consideration only the years from 2012 to 2014.

4.3.1 Some descriptive statistics

Figure 12 plots reviews against dates and we can see that there has been an upward trend in the two years under consideration. In Figure 15 we can see the plot of reviews against rating and we notice that the distribution is skewed to higher ratings.

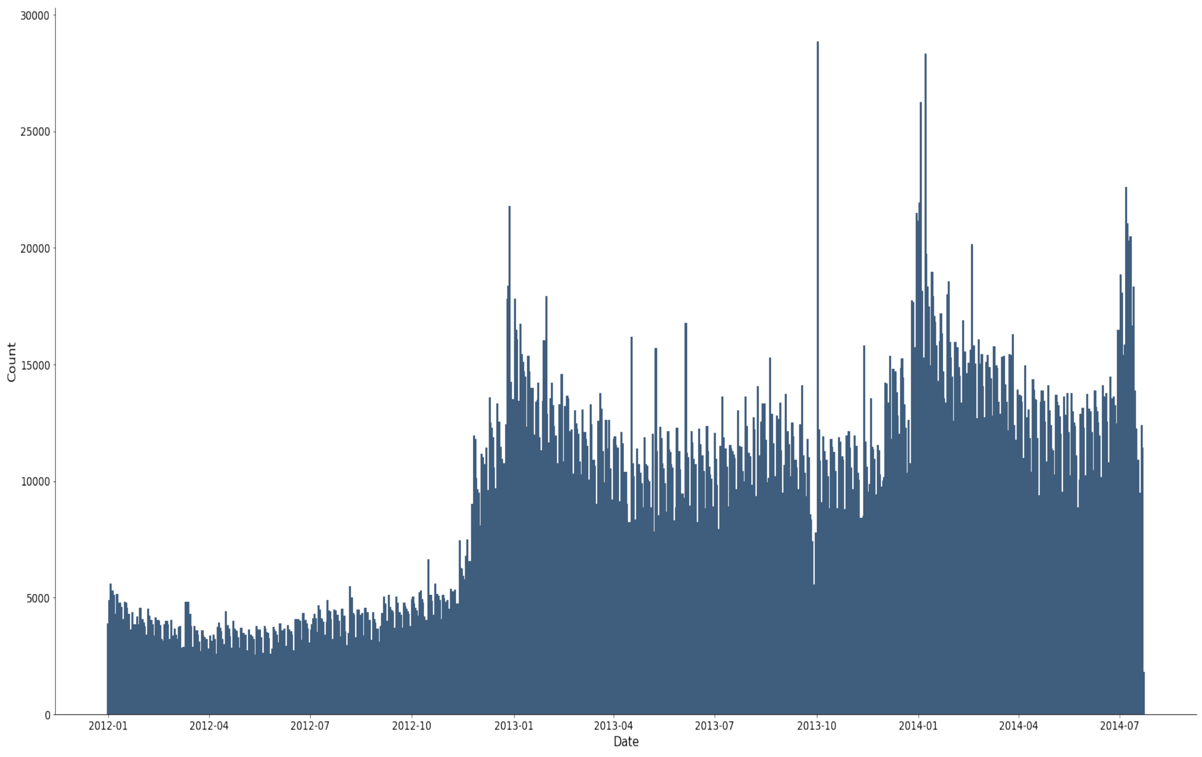


Figure 12 “Electronics”+“Cell Phones” reviews by date, 2012/01 - 2014/07

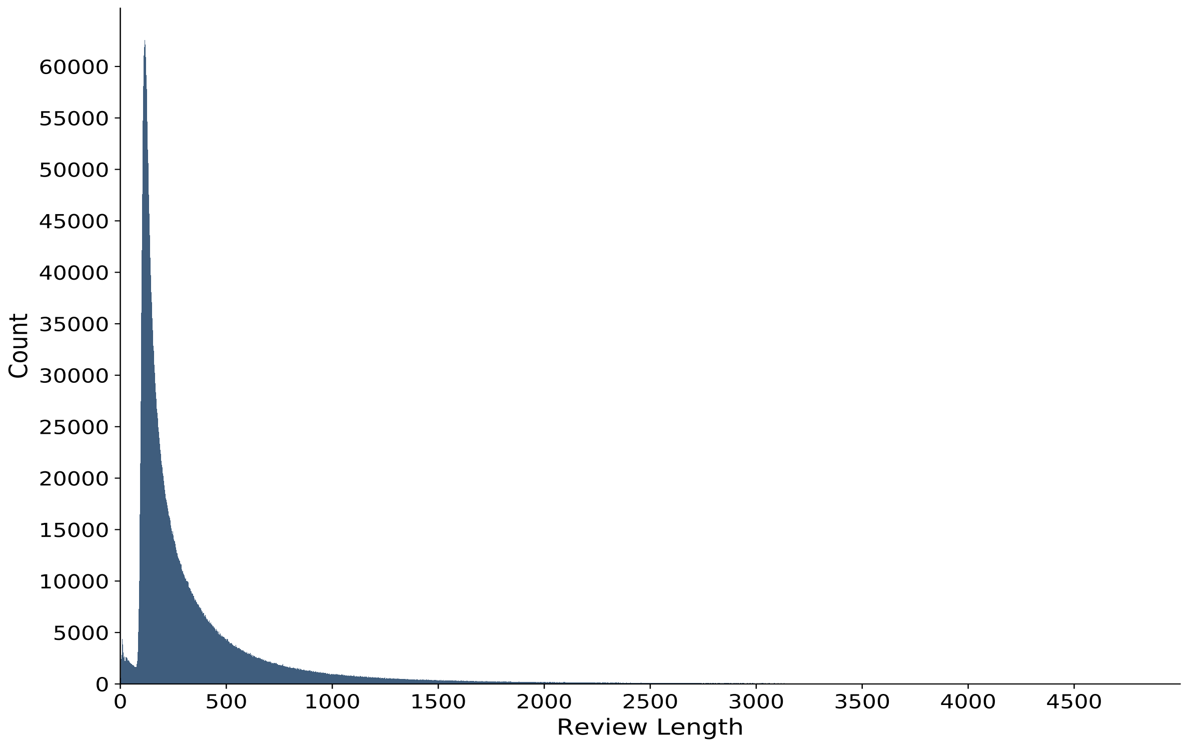


Figure 13 Distribution of reviews lengths

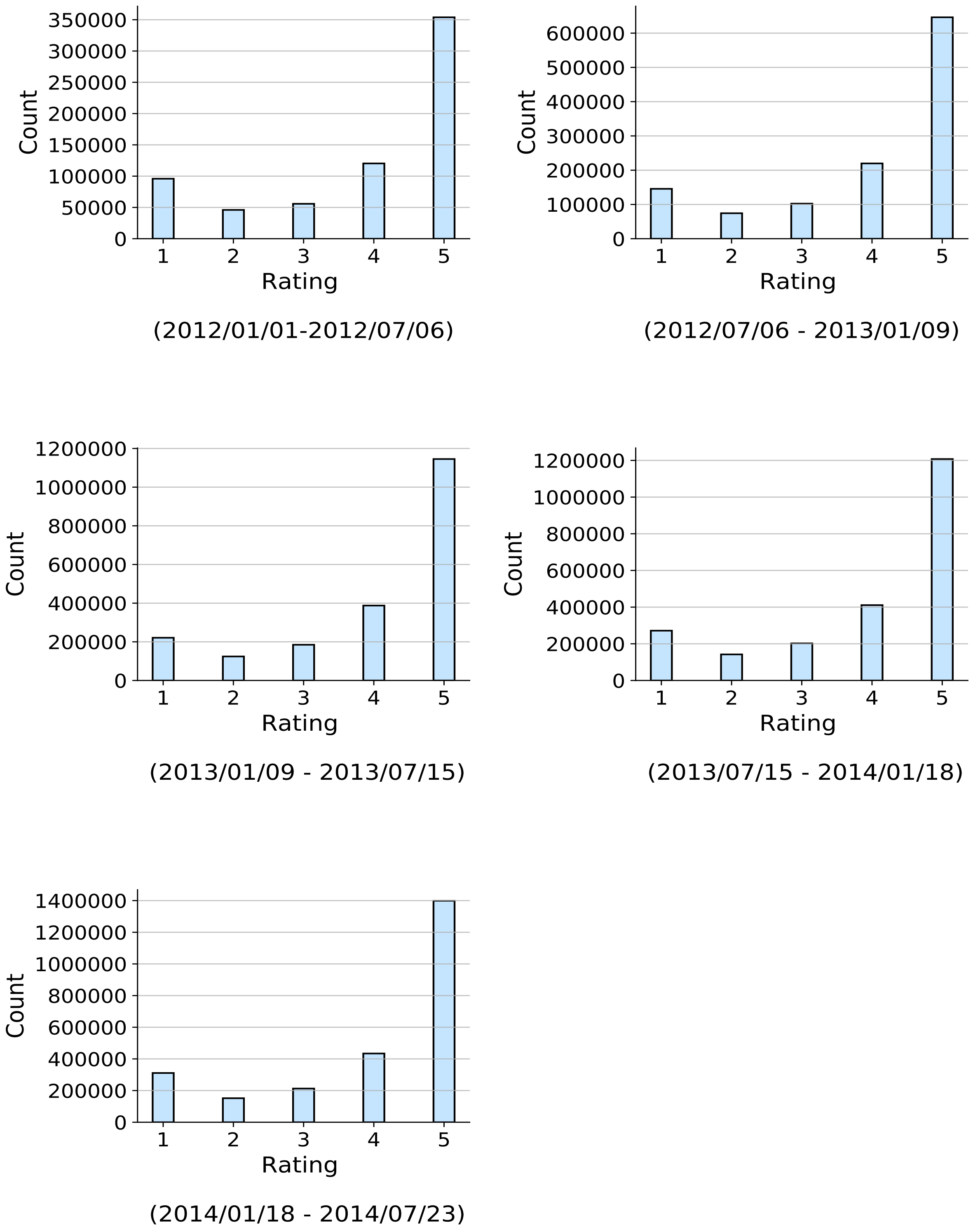


Figure 14 Evolution of ratings distribution from 2012/01/01 to 2014/07/23

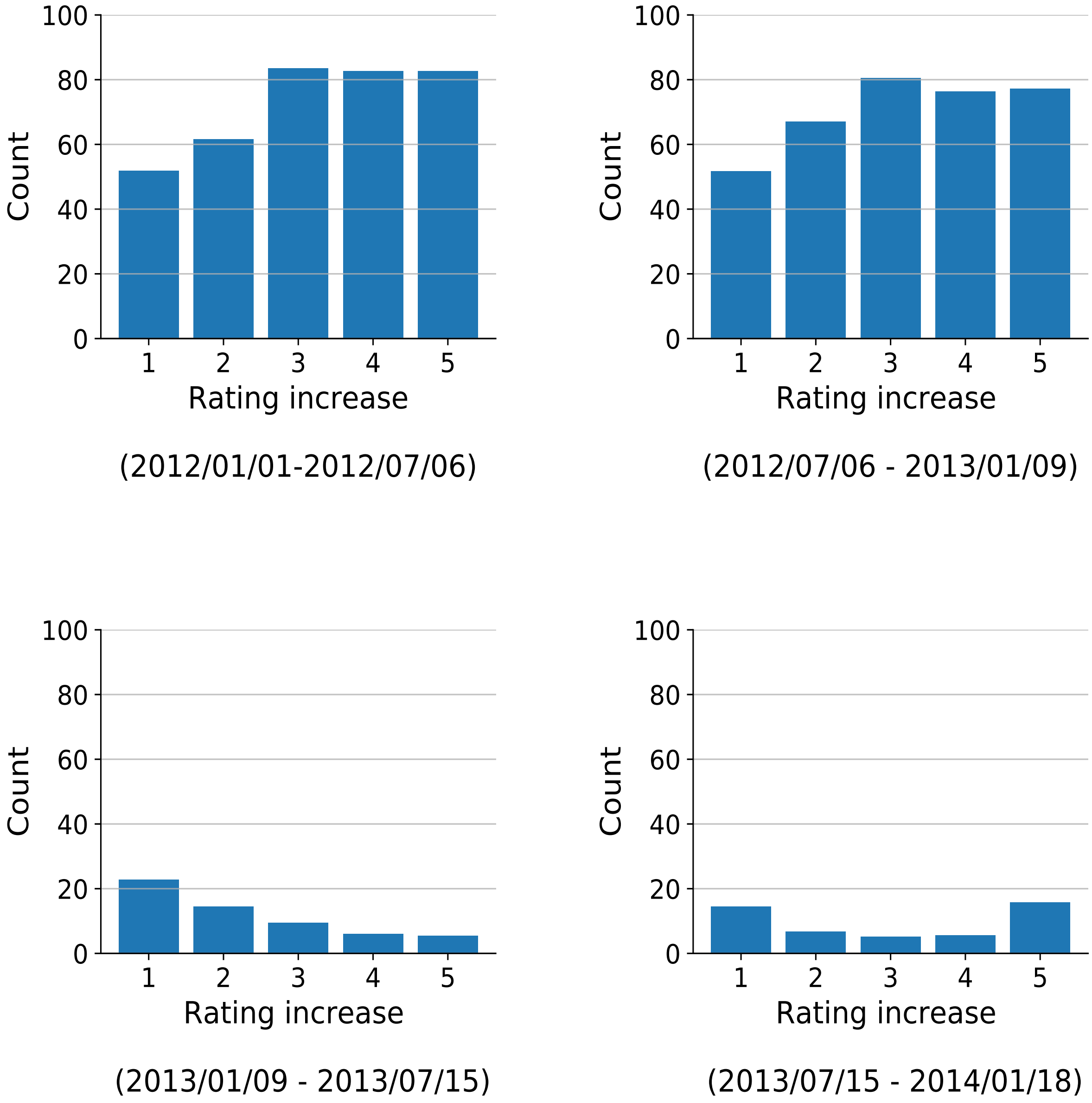


Figure 15 Pct. increase of num. of reviews per label (2012/01/01-2014/07/23)

CHAPTER 5

TEXT CLUSTERING

5.1 Proposed approach

This section of our work is split into two separate parts: in the first part we work on product titles, while in the second part we work on product reviews. In the first part we focus more on obtaining the best number of clusters, while in the second part we make a more extensive evaluation of similarities between clusters obtained starting from various text representations, which is something we touch on only briefly in the first part.

In the first part we take the following steps:

- we preprocess and tokenize product titles.
- we vectorize product names according to these three vectorization flows:
  - tf-idf+ SVD
  - Count-vectorization + LDA
  - word2vec + averaging
- for each vector representation we perform K-means clustering and we select best number of clusters with elbow-knee method, which is explained below.
- we compare clusterizations obtained with different vectorization flows using Adjusted Mutual Information score (AMI).

In the second part we take the following steps:

- we preprocess and tokenize product reviews.
- we vectorize product reviews according to the same vectorization techniques used for product titles.
- we perform hierarchical clustering at different granularities and, again, we use AMI score to compare clusterizations.

5.2 Product names clustering

We work on a subset of the dataset, composed of 50,000 product names, which was obtained through random sampling.

5.2.1 Product names preprocessing

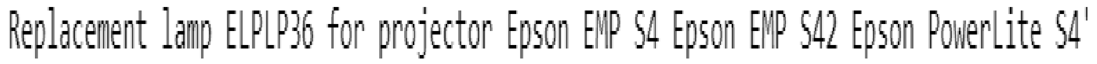

Replacement lamp ELPLP36 for projector Epson EMP S4 Epson EMP S42 Epson PowerLite S4'

Figure 16: Example of product name

As we can imagine and can see in Figure 16, product names are usually quite short strings and do not need much preprocessing.

We will perform only the following simple steps:

- tokenization, splitting at every punctuation mark

- lowercasing
- stopwords removal
- delete ”quot” and ”amp” tokens, since they are obtained from splitting the encoded version of ampersand and quotes: ”&amp”, ”&quot”

5.2.2 Vectorization of product titles

As we can see in Figure 17 we follow three different dimensionality reduction workflows and we are going to compare the clusterings obtained with them. Given that we do not have a ground truth, we don’t have a measure of which dimensionality reduction flow works best. Our idea is to see how similar the clusterizations yielded by them are.

5.2.2.1 Tf-idf + SVD

In the first dimensionality reduction flow, we start by transforming each document in their tf-idf counterparts. At this point we are left with 58581-dimensional vectors. In order to make clustering feasible we need to reduce the dimension of these vectors, therefore we apply SVD. In particular we set k to 300.

5.2.2.2 Count-vectorization + LDA

In the second dimensionality reduction flow we transform documents in their bag of words counterparts and then we apply LDA to reduce dimensionality. Given the application, we are not interested in what the topics extracted by LDA are, in fact LDA should enable us to capture various aspect of products, which ideally should be ’product type’, ’product colour’, ’brand’ etc. on the basis of which we will perform clustering

As for SVD we would like to obtain 300-dimensional vectors, therefore we select a number of topics equal to 300 and obtain the representation of documents in terms of the 300 topics extracted by LDA.

5.2.2.3 Word2vec training + averaging

We are confronted with a vocabulary which is full of uncommon occurences, given that we are dealing with product names. For this reason we prefer not to use pretrained word embeddings and we choose to train new word2vec vectors directly on our corpus. Again, the desired dimension is 300 and this will be the size of the embeddings.
Once we have obtained embeddings, we need a representation for each document. We choose to represent a document as the average of the word vectors corresponding to the words it contains.

5.2.3 K-means

K-means is a clustering technique which usually considers Euclidean distance as similarity metric. Starting from data points and a fixed number k of clusters, its objective is to minimize the Within Cluster Sum of Squares (WSS):

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} (||\mathbf{x} - \mu_i||)^2$$

In the formula above, S is a partition of the data points $S = (S_1, S_2, ..., S_k)$ and we want to find the partition which minimize WSS. Since this is a NP-hard problem and is therefore very expensive computationally, some heuristics have been developed, which converge to a local optimum. The standard one is so common that it is referred as the "k-means" algorithm.

5.2.3.1 Standard K-means algorithm

Standard k-means algorithm is an iterative procedure. We start by randomly selecting k points in the vector space of our data points. These are the initial centroids of the clusters. Then we iteratively perform the following steps, until convergence is reached, that is to say when clusters stay the same after an iteration:

- we assign each data point to the cluster whose centroid is nearest to it, in terms of Euclidean distance.
- we calculate the new centroid of each cluster by averaging all data points of each cluster.

5.2.4 Elbow knee method

The aim of clustering is to have a low WSS (Within Clusters Sum of Squares), so that points in a cluster are very near each other and a high BSS (Between Clusters Sum of Squares) so that clusters are very well separated. Of course we could set the number of clusters equal to the number of points so as to minimize WSS and maximize BSS, but that would not be a sensible way of clustering points. On the contrary, the elbow knee method selects the number of clusters after which the WSS decreases very slowly and the Between Clusters Sum of Squares (BSS) increases very slowly, so that there is no sense in increasing the number of clusters any more.

TABLE III: WSS METRICS FOR SVD+TF-IDF

Cluster Metrics tf-idf									
Metric	2	100	500	1000	1500	2000	2500	3000	3500
WSS	1.	0.716	0.554	0.495	0.461	0.436	0.415	0.397	0.381
WSSdec	NA	0.283	0.161	0.059	0.033	0.025	0.020	0.018	0.016
BSS	0.044	0.477	0.726	0.818	0.871	0.912	0.944	0.974	1.
BSSinc	NA	0.432	0.249	0.091	0.053	0.040	0.032	0.030	0.025

5.2.5 Cluster number selection

As we said, K means is given in input a number k of clusters. We use the elbow knee method in order to select the optimal number of clusters. We ran the procedures ten times for each vectorizations to ensure consistency of clusterings, given that K-means makes use of random initialization.

In the case of SVD (Table III) we can see there is a strong discontinuity in the rate of WSS decrease and of BSS increase, therefore we can select 500 as best number of clusters.

TABLE IV: WSS AND BSS METRICS FOR LDA

Cluster Metrics LDA									
Metric	2	100	500	1000	1500	2000	2500	3000	3500
WSS	1	0.321	0.219	0.186	0.168	0.156	0.147	0.139	0.132
WSSdec	NA	0.678	0.102	0.032	0.017	0.012	0.009	0.007	0.006
BSS	0.249	0.819	0.914	0.944	0.961	0.974	0.984	0.992	1
BSSinc	NA	0.570	0.095	0.029	0.016	0.013	0.009	0.007	0.007

TABLE V: WSS AND BSS METRICS FOR WORD2VEC

Cluster Metrics w2v									
Metric	2	100	500	1000	1500	2000	2500	3000	3500
WSS	1	0.539	0.407	0.359	0.331	0.311	0.295	0.281	0.270
WSSdec	NA	0.460	0.131	0.048	0.028	0.019	0.015	0.0134	0.011
BSS	0.236	0.725	0.857	0.906	0.936	0.955	0.973	0.988	1
BSSinc	NA	0.489	0.131	0.049	0.029	0.019	0.017	0.015	0.011

Even in the LDA case (Table IV) and the word2vec case (Table V), k=500 presents a discontinuity both in the rate of WSS decrease and in the rate of BSS increase, therefore it is a good number of clusters.

Now that we have selected the optimal number of clusters for all vectorizations, we can calculate adjusted mutual information between the partitions obtained by the clusterizations. Our idea is to see whether clusterizations obtained by means of SVD, LDA and word2vec have a certain degree of similarity and, for this reason, can be considered a reliable basis to split products in groups and perform the identification of spam reviews.

5.2.6 Adjusted mutual information between hard clustering

AMI is a variation of mutual information which takes into account the increase of baseline mutual information, that is to say mutual information of two random clusters, when the dimensions of the clusters increase.
Given a set $S = s1, s2, ..., sn$ of n elements and two pairwise-disjoint partitions (hard clusterings) : $J = J1, J2, .., JM$, $L = L1, L2, .., LK$, we are interested to know how much the partitions agree between each other. AMI measures exactly that and it is perfect for our problem, because we are dealing with hard clusterings.

$$AMI = \frac{MI(J,L) - E(MI(J,L))}{max\{H(J), H(L)\} - E(MI(J,L))}$$

where MI(x,y) is the Mutual Information, H(x) is the Entropy and E(x) is the Expected Value.

5.2.7 Clusterings AMI agreements results

Now we can presents the AMI agreement between all the couples of clusterings we took into consideration. In Table VI we see the results. As we can see in Table VI there is very good agreement between word2vec and tf-idf and way above chance agreement in the other two cases.

5.3 Product reviews clusterings

For product reviews we chose to evaluate the evolution of AMI between different clusterizations when varying the number of clusters.

TABLE VI: AMI BETWEEN COUPLES OF CLUSTERINGS

Agreement between clusterings of product titles			
Metrics	tf-idf+SVD w2v+avg	tf-idf+SVD LDA	w2v+avg LDA
AMI	0.46	0.26	0.26

TABLE VII: AMI SCORES FOR PRODUCT REVIEWS

Agreement between clusterings of product reviews			
Metrics	tf-idf+SVD w2v+avg	tf-idf+SVD LDA	w2v+avg LDA
AMI	0.18	0.054	0.053

5.3.1 Data

In the previous section we obtained 3 clusterizations of product titles. In this section we consider the clusterization obtained starting from product titles vectorized with tf-idf+SVD. For each cluster we consider the reviews of all its product, provided that the number of reviews is greater than 1000. From the above filtering, we are left with 208 sets of reviews, given that the other 292 clusters had less than 1000 reviews each.

5.3.2 Clustering

For each one of the 208 sets of reviews we apply hierarchical clustering. We chose hierarchical clustering, given that it is deterministic, except for tie-breaking, so that a single run is needed in order to ensure consistency. Given a set of reviews and a vectorization we cluster reviews with 10 different granularities: we go from 2 clusters, to $|set\_of\_review|/10$ clusters.

5.3.3 Agreement between clusterings of product reviews

Each score in Table VII is averaged over all sets of reviews. Furthermore, we averaged over all the granularities, since their scores were very similar.

Results seem to confirm that tf-idf+SVD and w2v+averaging vectorizations give rise to text clusterizations which have substantial agreement in terms of AMI.

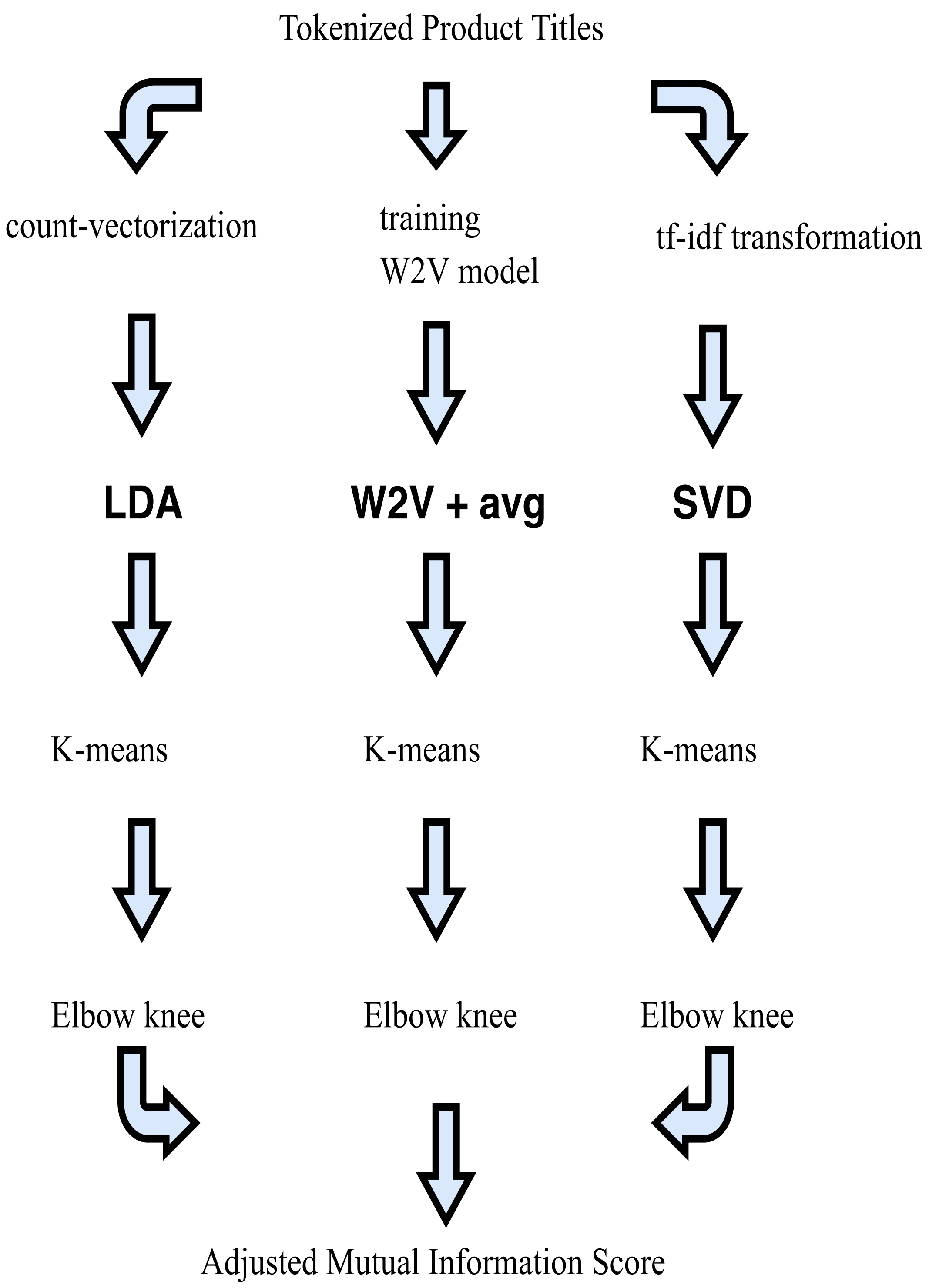


Figure 17: The three different vectorization processing flows

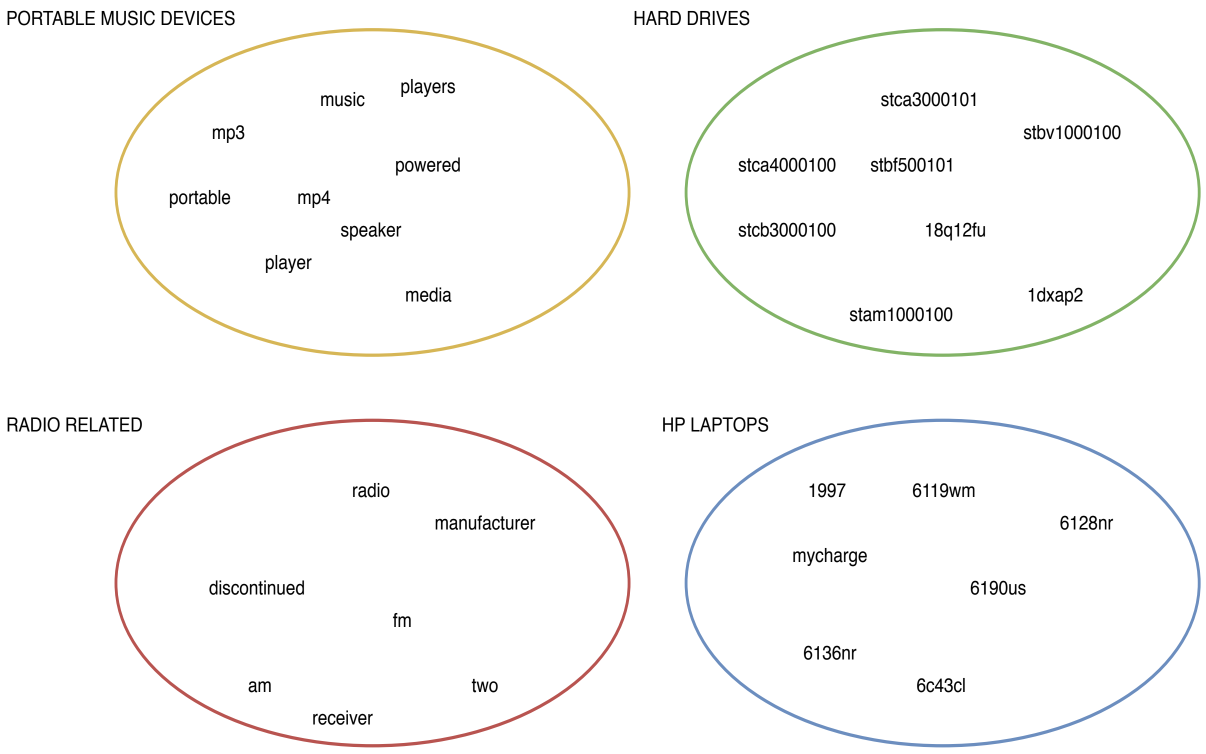


Figure 18: Some topics extracted with LDA and most important words

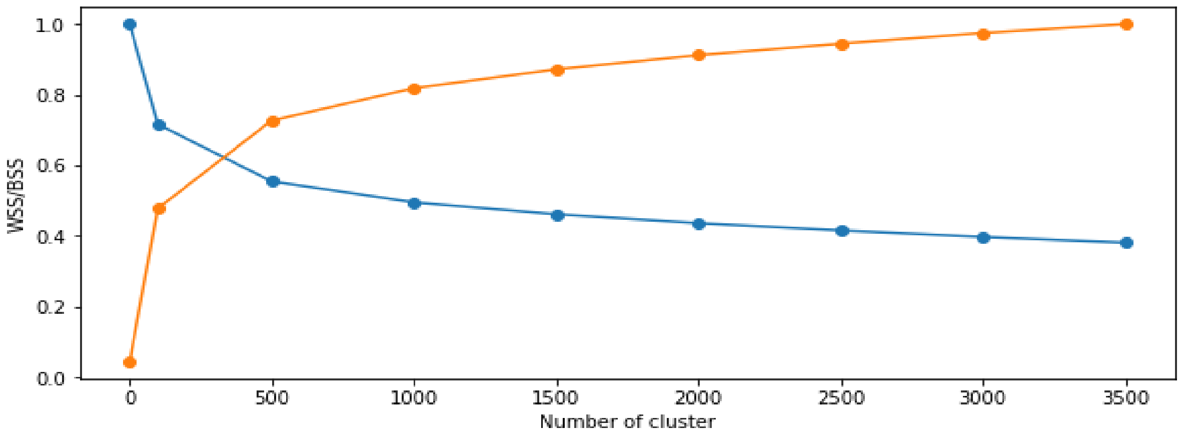


Figure 19: Knee elbow graph for SVD+tf-idf

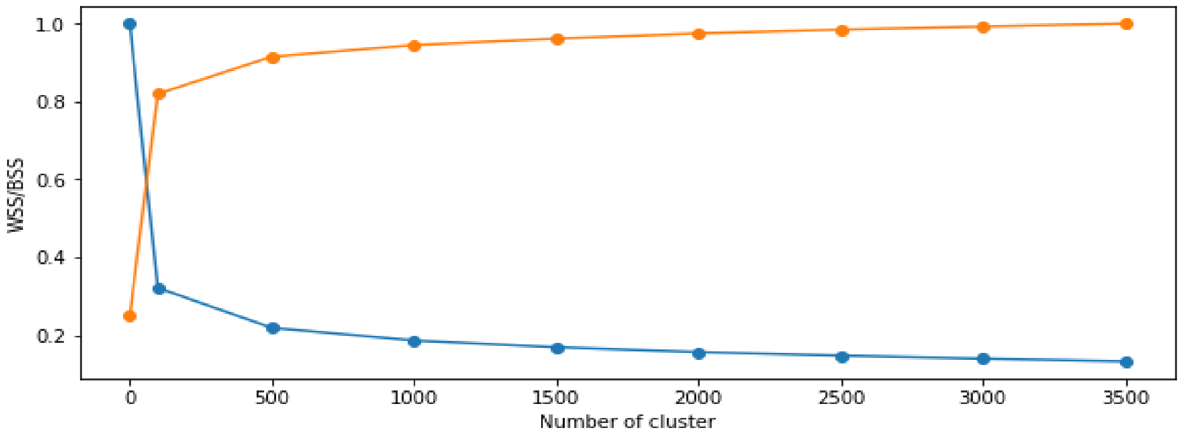


Figure 20: Knee elbow graph for LDA

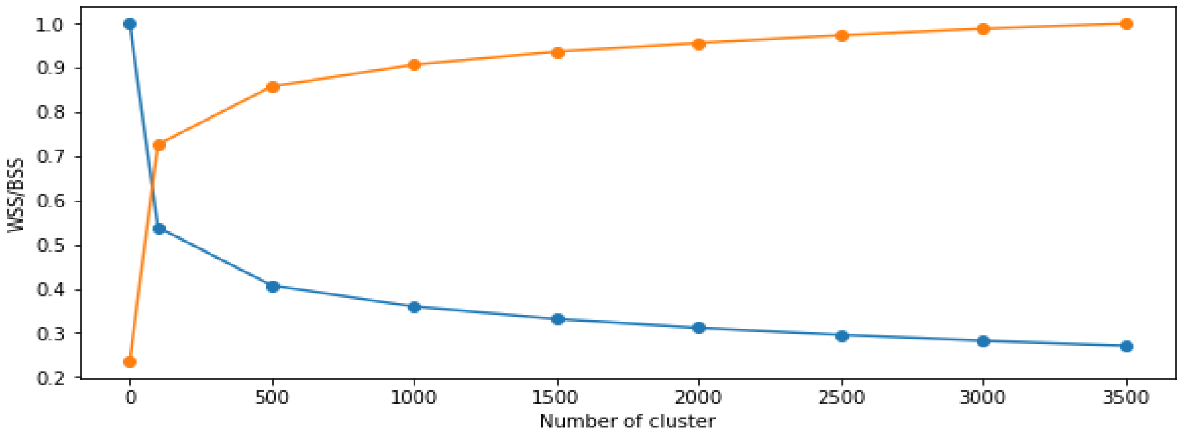


Figure 21: Knee elbow plot for word2Vec

CHAPTER 6

OPINION POLARITY ANALYSIS

For the opinion polarity analysis of our work, we focused on predicting the star rating of reviews, which is interpreted as the sentiment of the review's text and ranges from 1 to 5, where 1 is very negative sentiment and 5 is very positive sentiment. We will compare different approaches: we will start with baselines such as Naive Bayes and SVMs and we will get to Deep Learning approaches. We will see he strengths of both worlds and compare how different Deep Learning architectures perform.

6.1 Workflow description

For this part of our work we will proceed according to standard supervised learning procedure and we will build a train set and a test set. We will use only a part of the Mobile Accessories data we have available, in order to speed up training. Our training set consists in 100,000 reviews, while our test set is of 20,000 reviews. Training and test set were randomly drew from the dataset as a whole.

As we said in the introduction, our final aim is to deploy our sentiment analysis engine on a variety of media such as Twitter, online blogs, Facebook pages, online news, Instagram etc., therefore we are interested in building models which are able to generalize to new domains and be robust to noise which is particularly present in Social Networks domain. Given our intentions, we will simulate a setting which is different from training domain. Of course, each of the above mentioned media will present its own challenges, e.g. Twitter has very short messages, Instagram has lot of slang, etc. For this reason, we decided to start evaluating Transfer Learning in a setting where only the domain (Electronics, Clothing) changes, rather than the way users express their views. We will proceed according to the following pipeline:

- Standard Training and Test on Electronics and Mobile Phone Accessories.
- Build Train and Test set for this category: Clothing.
- Compare training on a domain + Transfer Learning versus training directly in the domain of interest.

One aspect of our research is the comparison between classical ML algorithms and DL models. In the first part of this chapter we will train and evaluate a wide variety of models, both classic and DL.

Subsequently, we will select a subset of these models to be evaluated in the second (Transfer Learning) step. In the end we will train all of the models for the third step.

6.2 Classic ML algorithms

In the Literature Review section we described two algorithms which are very commonly used in NLP classification problems, that is to say Naive Bayes and SVM.

However, we will not limit ourselves to those. Instead we will consider a variety of ML algorithms such as Naive Bayes, XGboost, Random Forests etc.

Random forests

Random Forests is an ensemble method which is based on decision trees, in particular on averaging a number of unpruned regression trees. Given that it is an ensemble method, in order to be able to exploit its potential we need to generate a series of different weak learners. We start from the same dataset so we need a strategy to generate the above mentioned different learners. Random Forests makes use of bootstrap aggragation, that is to say it generates a variety of new training sets by uniformly sampling the starting dataset with replacement. Furthermore it adds another source of randomness by splitting each trees on only a part of the available featyres. Random Forests improves the variance reduction of bootstrap aggregation by creating little correlated trees.

XGBoost

XGboost is the short for Extreme Gradient Boosting and it is an implementation of Gradient Boosting Machines. GBM is a family of model which have yielded good results in various practical applications. XGBoost has a parellel implementation which makes it faster compared to similar algorithms. Just like Random Forest XGBoost is an ensemble method. In general GBMs method works by trying to iteratively improve the weak learners.

Starting from a model F we can calculate the loss function L which for ecample could be MSE. At each iteration the learner F is updated in such a way $F_{m+1}(x) = F_m(x) + d(x)$ where our objective is to make F equal to the target function. The way we select $d(x)$ is by going in the direction opposite of the gradient of the loss function L.

6.3 Deep learning

6.3.1 Architectures

We will focus on RNN and CNN. For RNN we will employ the LSTM unit while for CNN we will consider the architecture proposed in [26], which consists in 1 Convolutional Layer with six filters, 2 for each filter size(2,3,4), followed by a MaxPooling layer, whose outputs are concatenated and input into a fully connected layer, that is to say a standard NN.

6.3.2 Word embeddings

Our experiments will not make use of pre-trained vectors, but rather both DL architectures will have an Embedding Layer which will be trained together with the rest of the Network.

6.4 Evaluation metrics

The task we are dealing with is a multiclass classification. Therefore, we could consider both Accuracy measure and F1 measures. However, our dataset is imbalanced as we can see in Figure 15 (Chapter 3), therefore we prefer F1 as accuracy would not be really informative.
In particular, for multiclass problem, it is not sufficient to speak of F1 measure. We have to specify how we deal with the presence of more than 2 classes in the dataset.

6.4.1 F1 for multiclass

F1 in multiclass problem is calculated in one-versus all manner, that is to say TP, FP, FN are calculated as if we were in a binary problem with the considered class versus all the others grouped together.
Main F1 measures for multiclass are:

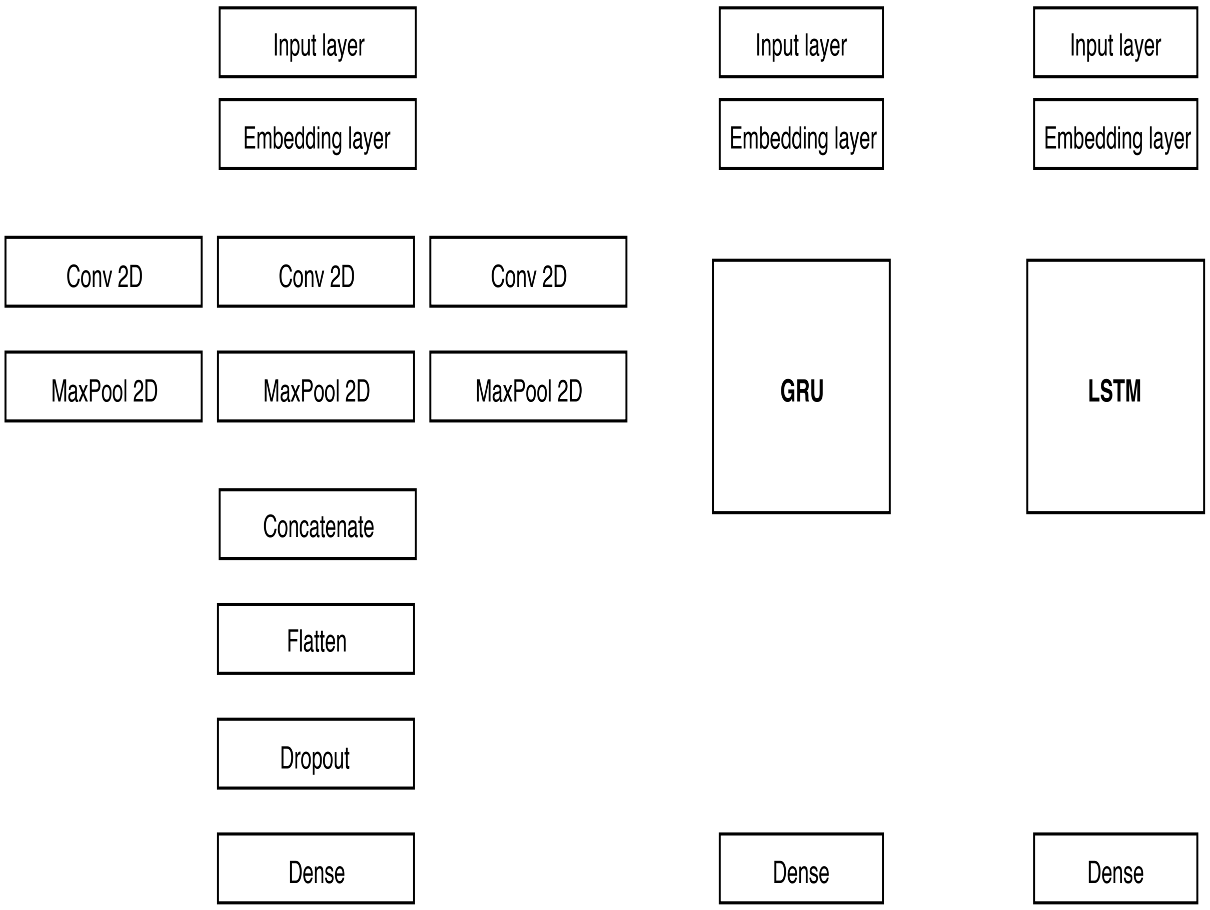


Figure 22: Architecture of implemented models

- micro-F1: TP,FP,FN are calculated globally
- macro-F1: considers each class separately and then averages the F1 scores

However, we would like to obtain a good F1 for each of the classes, since each of the classes is equally important for us. Therefore we consider F1 for each class separately.

TABLE VIII: NORMAL 10-FOLD CV

F1 for each class					
Model	Class1	Class 2	Class 3	Class 4	Class 5
MNB	0.626	0.088	0.180	0.332	0.793
BNB	0.570	0.116	0.121	0.270	0.737
RF	0.	0.	0.	0.	0.711
SVM	0.641	0.076	0.175	0.173	0.794

6.5 Dataset imbalance, undersampling+crossvalidation

By performing crossvalidation on the training set with a variety of models, we see that some classes are really overlooked. This is due to the fact that they are underrepresented. In order to overcome this fact, we implement a crossvalidation with undersampling:

- create all train test splits
- undersample only the train sets, with n samples for each class, where n = number of samples of the minority class in specific train set

We don't want to undersample the test sets, otherwise we would have a completely different distribution from the real test set and this would compromise our results.
As we can see by looking at Table VIII and Table IX, undersampling, although reducing F1 for Class1 and Class 5, considerably increases F1 for Class 2 and Class 3, so that performances are better overall.

TABLE IX: UNDERSAMPLING+10-FOLD CV

F1 for each class					
Model	Class1	Class 2	Class 3	Class 4	Class 5
MNB	0.557	0.256	0.285	0.361	0.693
BNB	0.582	0.211	0.252	0.285	0.747
RF	0.535	0.173	0.242	0.324	0.727
SVM	0.612	0.200	0.282	0.343	0.775
LSTM	0.641	0.125	0.326	0.399	0.735
CNN	0.672	0.143	0.363	0.429	0.757

6.6 Error analysis

In Figure 23 we can see a confusion matrix obtained by applying Naive Bayes in Cross Validation, both with and without undersampling the Training Set. When we did not apply undersampling, the error distribution for Class 1 and Class 2 follows an unusual patterns, e.g. more reviews from Class 1 are classified as Class 5 than Class 2. When applying undersampling the error distribution becomes more sensible and misclassified reviews are usually classified in the "neighbour" classes.

6.7 Transfer learning

First we obtain a small training and test set for the class that we are interested in: Clothes. Training and test set will be small, 8000 reviews for training set and 2000 reviews for test set, in order to simulate hand-labelling set.

In Twitter, Instagram and other social-network based studies, in fact, we don't have star-ratings and the only way to obtain reliable datasets for sentiment analysis is to use hand-labeling, unless some posts present emoticons, which could be used for automatic sentiment extraction.

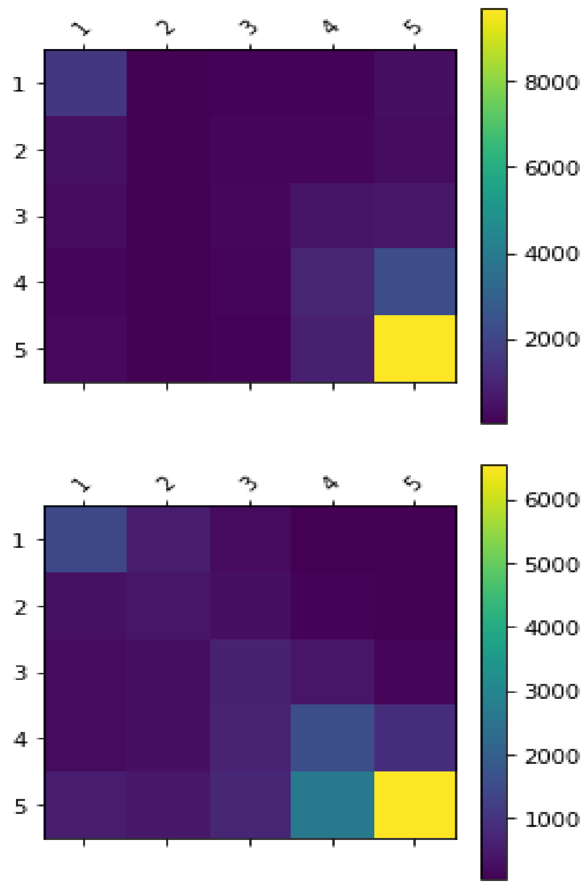


Figure 23: Confusion matrix without (above) and without (below) undersampling

In order to evaluate the effect of Transfer Learning we will proceed in two different ways depending on whether the model is a DL model. For standard ML models we will proceed in the following way:

- We use partial-fit to train the model on the Electronics dataset
- We use partial-fit to train the model on the small training dataset of Clothes.

TABLE X: UNDERSAMPLING+10-FOLD CV, NO TRANSFER LEARNING

F1 for each class					
Model	Class1	Class 2	Class 3	Class 4	Class 5
MNB	0.412	0.236	0.258	0.324	0.414
BNB	0.432	0.112	0.187	0.241	0.729
SVM	0.331	0.151	0.234	0.292	0.581
LSTM	0.408	0.252	0.262	0.301	0.704
CNN	0.434	0.267	0.282	0.331	0.722

TABLE XI: UNDERSAMPLING+10-FOLD CV, TRANSFER LEARNING

F1 for each class					
Model	Class1	Class 2	Class 3	Class 4	Class 5
MNB	0.507	0.194	0.298	0.360	0.754
BNB	0.478	0.201	0.339	0.283	0.754
SVM	0.514	0.187	0.281	0.356	0.773
LSTM	0.512	0.285	0.354	0.341	0.744
CNN	0.534	0.306	0.371	0.381	0.786

TABLE XII: PERCENTAGE CHANGE WITH TRANSFER LEARNING

F1 for each class					
Model	Class1	Class 2	Class 3	Class 4	Class 5
MNB	0.095	-0.042	0.040	0.036	0.340
BNB	0.046	0.089	0.152	0.042	0.025
SVM	0.183	0.036	-0.047	0.064	0.192
LSTM	0.104	0.033	0.092	0.040	0.040
CNN	0.100	0.039	0.089	0.050	0.064

For DL Models we do:

- We train models on electronics
- We save weights and reuse them to train model on the small training dataset of either Movies or Clothes

After this procedure we will confront the results and we will see how much Tranfer Learning can improve performance on small training datasets and also to what extent the change of domain degrades performances.

In the tables we can see the results of undersampling+crossvalidation both in the case where we train only on the new domain dataset or where we use transfer learning. As we can see in Table X and Table XII, in all crossvalidation experiments by doing Transfer Learning we have an improvement in performance.

6.7.1 Comparison with results from literature

To the best of our knowledge there are no works which consider single class F1-score in Sentiment Classification. F1-score and also multiclass classification do not seem to be the norm in Sentiment Classification literature. As a reference we mention [27] and [28] which deal with 5-class classification and use Accuracy as evaluation metric. Results for multiclass classification are in line with what we obtained in our experiments, however, as we already said, we preferred to highlight single-class results.

In Table XIII we can see a comparison of results obtained in our work and in [27], in terms of Accuracy. Of course, results depdends on the nature of the data (in [27] they deal with

TABLE XIII: OUR RESULTS AND RESULTS FROM SOCHER'S PAPER

Accuracy		
	Socher's results	Our results
NB	67.2	63.2
SVM	64.3	64.4
BNB	71.0	57.9

IMDB reviews) and on the dataset distribution, which in the case of the considered paper is less unbalanced than in our case.

CHAPTER 7

CONCLUSIONS AND FUTURE WORKS

In this work we have explored the world of online reviews. We have recognized the relevance data have for consumers and for companies and we have thought of ways to extract values from them. We have identified two problems which we considered worth of particular attention and we have seen how Machine Learning is a good tool to tackle them, these problems are Document Clustering and Transfer Learning for Sentiment Classification. Before devising our specific solutions, we analysed Machine Learning Literature looking for various approaches in the field of NLP.

Text Clustering part was divided into two steps: clustering of product names and clustering of product reviews. For product names we explored three different vectorization flows: tf-idf+SVD, LDA and word2vec+ averaging. After having vectorized our documents, we applied K-means. Two interesting points emerged during this phase: knee elbow plots were very similar for the different vectorization flows so that all methods agreed in the optimal number of clusters. Furthermore, by measuring adjusted mutual information score between the obtained clusterings, we found good agreement and this gives us confidence that the obtained clusterings are meaningful. It is also interesting from a problem-agnostic point of view that vectorization methods quite different in nature (tf-idf+SVD) and word2Vec should give rise to very similar clusterings (0.46 AMI).

Given the interesting results of our first attempt, we replicated our pipeline on product reviews, with some modifications: we chose Hierarchical Clustering instead of K-means, in order not to have random inizialization which would have led us to run the experiments multiple times, and we explored different levels of granularities. Even in this case, at each level of granularity, we found good agreement between tf-idf+SVD and word2vec+averaging(0.18 AMI).

As for the Opinion Mining part, we first compared standard Machine Learning approaches to Deep Learning one on a single domain (Electronics). We dealt with class imbalance problem by undersampling our training set and this gave better results in all test cases. Deep Learning approach outperformed traditional approaches and may be preferred depending on whether we would like maximum performance, and then we would choose DL or a more explainable behaviour and then we would stick with traditional ML.

In the Transfer Learning phase we evaluated how sentiment classification performance of algorithms both from classic ML and from DL on a test set for which very few training data are provided can be boosted to a significant degree by pretraining the model on a different and unrelated domains (Clothing). Our findings allowed us to confirm the hypothesis. We are thus led to believe that training models on multidomain dataset should boost performances on datasets with usually small training datasets such as online news or blogs.

This is in fact one of the possible extensions of our work, that is to say, to validate Transfer Learning from reviews to both related and unrelated domain but with data other than online reviews. Other possible integrations in Sentiment Classifications are different DL architectures, such as Attention Network or GANs, while, as for the field of Spam Detection, we used a dataset subsample and it would be interesting to replicate our findings on the whole dataset, given enough computational power. Another less related but interesting experiment would be to exploit reviews data to help in forecasting of sales data time series.

In this work, our main contribution in Text Clustering is in terms of comparison between clusterings obtained after different text vectorizations: tf-idf+Singular Value Decomposition (SVD), LDA and word2vec. As for the Sentiment Classification and Transfer Learning part, our main contribution is an extended evaluation of 5-item scale Sentiment Classification on a vast unbalanced dataset and the assessment of both standard Machine Learning (ML) and Deep Learning (DL) models' performances in a multiclass classification Transfer Learning setting. The work done so far has given us the possibility of understanding how much value can be extracted from online reviews and how versatile NLP tools and especially ML ones are when tackling the most varied of challenges.

APPENDICES

Appendix A

SAMPLE PRODUCTS FROM CLUSTERS OBTAINED WITH TF-IDF+SVD VECTORIZATION

Cluster 300:

['[10824] METALLIC LIGHT PURPLE — Apple iPod Nano 3 3rd Generation Case with Belt Clip amp; Hook Ring. Bonus Ekatomi screen cleaner'] ['Red Silicone Rubber Gel Soft Skin Case Cover for Ipod Nano 6th Gen Generation 6g 6 8gb 16gb 32gb by Electromaster'] ['Apple iPod Nano 5th Generation Crystal Silicone Skin Case Transparent Purple Diamonds'] ['Skque Premium Wall Charger Adapter and Car Charger Adapter for Apple Ipod Nano 6th Generation (Newest Model)'] ['iTALKonline BLACK Sports GYM ArmBand Case Cover for Apple iPod Nano 7 7G 7th Generation (2012)'] ['Green Color Apple iPod nano 4G (4th Generation) Aluminum Metal Case'] ['Apple iPod Nano 7 (7th Generation) Bunny Skin Case + 1 Fruity Dust Plug, Purple [Cellular Connection Packaging]'] ['4 Cases for iPod Nano 4th Generation Protective Snap-On Cases, Pack of 4: Rubberized Black, Blue, Green, and Red']

Cluster 400:

['Epson POWERLITE 475W PowerLite 475W Multimedia Projector'] ['PCMD Projector Ceiling Mount for Epson PowerLite Home Cinema 730HD'] ['84x84 Manual Projector Projection Pull Down Screen 120quot; 1:1 4:3 16:9 Display'] ['ISONIC HD8600 HDMI PICO PROJECTOR'] ['PCMD All-Metal Projector Ceiling Mount with 14quot; Extension for Optoma HD20'] ['3M Mobile Projector Spare Battery for MP225a Projector'] ['iSiVaL LED Mini HD Projector']

Appendix A (continued)

['Eco Expedition Starlight Projector'] ['Optoma PT100, WVGA, 50 LED Lumens, Gaming Projector'] ['VideoSecu Projector Ceiling Mount Bracket Holder PJ2B B13'] ['InFocus Work Big IN2102EP Projector']

Cluster 490:

['Hktimes Wireless Bluetooth Keyboard Leather Case Folio for Ipad 3 2 1 (Green Leather+green Keyboard)'] ['BoxWave Type Runner Keyboard for iPad - Ultra Portable Bluetooth Keyboard for iPad with Integrated Apple Commands - iPad Wireless Bluetooth Keyboard (Silver White)'] ['GLYBY-Folding Leather Protective Case Wireless Bluetooth Keyboard For Ipad/Ipad 2/Ipad3 - Black'] ['SUPERNIGHT 135 Rotating Angle Wireless Bluetooth Keyboard Case Cover Stand Combo for Apple iPad Mini Aluminum Modern Silver Style'] ['Aluratek Slim Color Folio Case with Bluetooth Keyboard for iPad 2/3 - Sky Blue (ABTK02FSB)'] ['TrendyDigital Laminated Canvas PadGear(TM) Plus Folio Case, Cover, Pouch for Apple iPad 2, with Viewing Stand and Pocket for Apple Wireless Keyboard (Wireless Keyboard is not included), Black']

CITED LITERATURE

1. Blitzer, J., Dredze, M., and Pereira, F.: Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In Proceedings of the 45th annual meeting of the association of computational linguistics, pages 440–447, 2007.

2. Jindal, N. and Liu, B.: Opinion spam and analysis. In Proceedings of the 2008 International Conference on Web Search and Data Mining, pages 219–230. ACM, 2008.

3. Harris, Z. S.: Distributional structure. Word, 10(2-3):146–162, 1954.

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

5. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J.: Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages 3111–3119, 2013.

6. Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759, 2016.

7. Pennington, J., Socher, R., and Manning, C.: Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014.

8. Hochreiter, S. and Schmidhuber, J.: Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.

9. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, 2014.

10. Khandelwal, U., He, H., Qi, P., and Jurafsky, D.: Sharp nearby, fuzzy far away: How neural language models use context. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), pages 284–294, 2018.

CITED LITERATURE (continued)

11. Krizhevsky, A., Sutskever, I., and Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, pages 1097–1105, 2012.

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

13. Xu, J., Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F., and Hao, H.: Short text clustering via convolutional neural networks. In Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing (VS@HLT-NAACL), pages 62–69, 2015.

14. Liu, T., Liu, S., Chen, Z., and Ma, W.-Y.: An evaluation on feature selection for text clustering. In Proceedings of the 20th International Conference on Machine Learning (ICML-03), pages 488–495, 2003.

15. Kuang, D., Choo, J., and Park, H.: Nonnegative matrix factorization for interactive topic modeling and document clustering. In Partitional Clustering Algorithms, pages 215–243. Springer, 2015.

16. Esuli, A. and Sebastiani, F.: Determining the semantic orientation of terms through gloss classification. In Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pages 617–624. ACM, 2005.

17. Hu, M. and Liu, B.: Mining and summarizing customer reviews. In Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 168–177. ACM, 2004.

18. Turney, P. D.: Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pages 417–424. Association for Computational Linguistics, 2002.

19. Hatzivassiloglou, V. and McKeown, K. R.: Predicting the semantic orientation of adjectives. In Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics, pages 174–181. Association for Computational Linguistics, 1997.

20. Pang, B., Lee, L., and Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing-Volume 10, pages 79–86. Association for Computational Linguistics, 2002.

21. Liu, B.: Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies, 5(1):1–167, 2012.

22. Daume III, H. and Marcu, D.: Domain adaptation for statistical classifiers. Journal of Artificial Intelligence Research, 26:101–126, 2006.

23. Aue, A. and Gamon, M.: Customizing sentiment classifiers to new domains: A case study. In Proceedings of Recent Advances in Natural Language Processing (RANLP), volume 1, pages 2–1. Citeseer, 2005.

24. Glorot, X., Bordes, A., and Bengio, Y.: Domain adaptation for large-scale sentiment classification: A deep learning approach. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pages 513–520, 2011.

25. He, R. and McAuley, J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In Proceedings of the 25th International Conference on World Wide Web, pages 507–517. International World Wide Web Conferences Steering Committee, 2016.

26. Kim, Y.: Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, 2014.

27. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1631–1642, 2013.

28. Tai, K. S., Socher, R., and Manning, C. D.: Improved semantic representations from tree-structured long short-term memory networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1556–1566, 2015.

CITED LITERATURE (continued)

29. Brightlocal: Local consumer review survey, online reviews statistics trends. `https://www.brightlocal.com/learn/local-consumer-review-survey/#Q1, 2018.` [Online; accessed 09/01/2018].

VITA

NAME	Paolo Polimeno Camastra
EDUCATION	
	Bachelor of Science in Computer Science and Engineering, Politecnico di Milano, 2013-2016
	Master of Science in Computer Science, University of Illinois at Chicago 2017-today
	Master of Science in Computer Science and Engineering, Politecnico di Milano, 2017-today
LANGUAGE SKILLS	
Italian	Native speaker
English	Full working proficiency
	2017 - IELTS examination (8.0/9.0)
	2013 - Cambrige Certificate of Proficiency in English(2013)
TECHNICAL SKILLS	
Advanced level	Data Mining, Machine Learning, NLP, Java, Python,C
WORK EXPERIENCE AND PROJECTS	
Dec 2017 - May 2018	NLP project on topic modeling
Sep 2017 - Dec 2018	Deep RL experiments on grounded language learning
Sep 2016 - Dec 2017	Data Science project on Credit Score prediction