Large Knowledge Bases and Networks: Results on Analogies and Betweenness Centrality

BY

RAHUL KUSHWAHA

THESIS

Submitted in 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, 2014

Chicago, Illinois

Defense Committee:
Gyorgy Turan, Chair and Advisor
Robert H. Sloan, University of Illinois at Chicago
Bhaskar DasGupta, University of Illinois at Chicago

To my father & mother,

for their endless support and love.

ACKNOWLEDGMENTS

I would like to thank Prof. Gyorgy Turan for his endless support since the first day of my school. His advise in academic and professional front has helped me clear all my doubts and fear. This work would not be possible without his excellent guidance, patience and care. I would also like to thank Prof. Robert H. Sloan for helping me during the different phases of Analogy Solver and providing me with great insights.

I would also like to express my gratitude to Dr. Dimitris Diochnos who allowed me to take his research forward, a few inches. He was always there to help me regarding any topic. He is an excellent mathematician and programmer with great explanatory skills. Special thanks goes to Prof. Bhaskar DasGupta, who was willing to participate in my Thesis Defense and Prof. Peter D. Turney who provided us with the analogy questions to experiment on.

Finally, I would also like to thank my parents, brothers, sisters and friends. They have always supported and encouraged me with their good wishes.

RK

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Betweenness Centrality and Analogy Solver are the main themes of this Thesis Report. In the first part we study betweenness centrality on general trees with some results regarding the relation between k-betweenness centrality and betweenness centrality. We also study complete binary trees to derive some new formulas with regards to number of shortest paths through a vertex. In the second part we provide an approach which tries to solve analogies using ConceptNet 4.0. A detailed analysis of few examples and results on the data set is also provided.

CHAPTER 1

INTRODUCTION

With the advent of Information Age, Large Networks have become an integral part our day to day life. Whether we know it or not, but we are surrounded by it all the times. For e.g., Facebook uses large graph to maintain information about all their users, scientists are studying Protein-Protein interaction networks to gain valuable insights about diseases, etc,. The Internet is also a network made of a large number of nodes where each node can represent a computer or a group of computers. Studying these large networks provides us with the opportunity to explore more about the structure, properties, vulnerabilities, etc,. But this comes at a price, every network is different in organization and features, and calls for a systematic and different approach for the study of each one of them.

Different measures, like centrality, have been invented since a long time to generalize some of the properties of large networks. One such centrality measure that we discuss in this thesis report is betweenness centrality and its variant. We discuss both form theoretical and application point of view.

From theoretical point of view, we discuss about the behavior of betweenness centrality in complete binary trees. We also discuss some of the general properties of the complete binary trees that can help us explore more about betweenness centrality. Betweenness centrality in case of general tree is also is also discussed. A special section pertaining to k-betweenness centrality is also presented where we study the study the basic definition, with some examples, and examine a particular behavior, Zig-Zag in k-betweenness centrality, found in certain trees. Construction of such trees is also provided.

In the last section we discuss an approach using ConceptNet4.0, a large semantic network, to solve SAT analogies. We also take help of spreading activation technique. A detailed description of all the algorithms used is also presented in this section. We also give a list of 5 examples that try to tell us more about the reasons for success or failures of our algorithms. We conclude this section showing the results of the experiment.

CHAPTER 2

BETWEENNESS CENTRALITY

2.1 Introduction

Centrality indices have proved to be a very critical tool for analyzing networks . It helps us calculate the importance, in an intuitive way, of nodes in the network with respect to other nodes. Of the many centrality measures, we will discuss a centrality measure based on shortest paths, defined for vertices. Betweenness centrality (C_B) is measure of centrality which is the ratio of total number of shortest paths passing through a node, excluding the paths that start and end on the given node, to the total number of shortest paths in the network. It can also be visualized as a relative measure of a node's participation in a communication network assuming that all communication takes place using shortest paths and all the nodes are communicating. We can also interpret $C_B(v)$ as the probability that v is involved in communication. According to (3) we can say,

Let $\delta_{st}(v)$ denote the fraction of shortest paths between s and t that contain vertex v:

$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$ where $\sigma_{st} = 1$ and $\sigma_{st}(v) = 1$ or 0

where σ_{st} denotes the number of all shortest-path between s and t. Then the betweenness centrality $C_B(v)$ of a vertex is given by:

$$C_B(v) = \sum_{s \neq v \in V} \sum_{t \neq v \in V} \delta_{st}(v)$$

Accordingly a node is more central in the network if more shortest paths pass through it connecting other nodes.

For instance, consider the tree that is shown below.

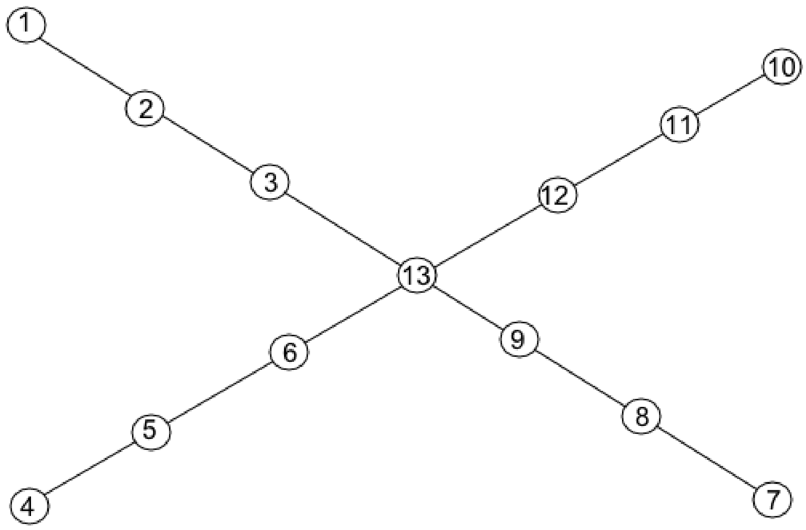

Figure 1. A simple tree (1).

In this example we can easily see from the table given below that the node 13 is more central as compared to other nodes followed by $3, 6, 9, 12$ and so on.

2.2 k-Betweenness Centrality

If we limit the length of Shortest Paths, say **at-most** "k", it is called **k-Betweenness Centrality** (C_B^k). It is this new localized measure which looks to provide some promising features. A claim is made by (1) which states that C_B^k is an approximation for C_B in large networks. This implies that the calculation of localized values can help us make decisions about global values. It is evidently true in the case of the tree in 2.1. Table II shows the different k-betweenness centrality values for the tree in 2.1. As we can see in this example the k-betweenness centrality certainly increases with increasing value of k.

But this raises a question: **Is is true that k-Betweenness Centrality (C_B^k) is monotonically increasing with k?** This may be true in some of the cases, like the example presented before, but there are certain trees where the k-betweenness centrality values of some nodes oscillate up and down with respect to each other. We call such a behavior as **Zig-Zag in k-Betweenness Centrality**. Figure 2.2 provides a simple example of such a tree. Calculation of betweenness centrality values for different nodes for different values of k is pre-

Node	**Betweenness Centrality Value (C^B)**
13	0.818
3, 6, 9, 12	0.303
2, 5, 8, 11	0.167
1, 4, 7, 10	0.000

TABLE I

Betweenness Centrality measures for the nodes of the tree shown in 2.1. Taken from (1).

Node(s)	C_2^B	C_3^B	C_4^B	C_5^B	$C_6^B = C^B$
13	0.273	0.506	0.709	0.788	0.818
3, 6, 9, 12	0.045	0.141	0.217	0.279	0.303
2, 5, 8, 11	0.045	0.056	0.098	0.131	0.167
1, 4, 7, 10	0.000	0.000	0.000	0.000	0.000

TABLE II

Betweenness Centrality measures for the nodes of the tree shown in 2.1. Taken from (1).

sented below in the form of a graph. Here each continuous line represent different k-betweenness centrality values for a particular node. As we can see from the graph shown that many lines cross each other many times indicating that their C_B^k values go up and down with respect to each other. In this case it is very difficult to predict the behavior of betweenness centrality from k-betweenness centrality.

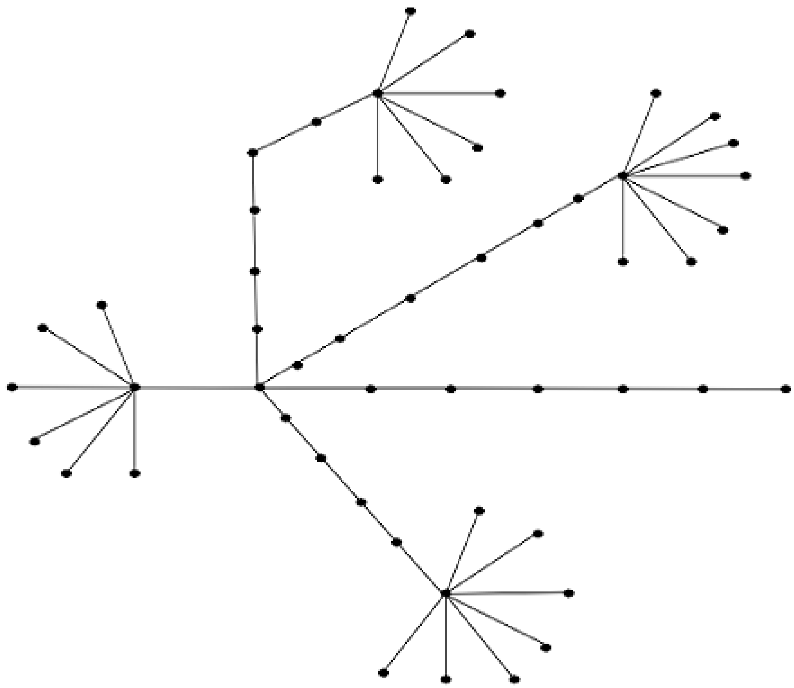

Figure 2. A simple tree in which the k-betweenness centrality of some nodes oscillate up and down.

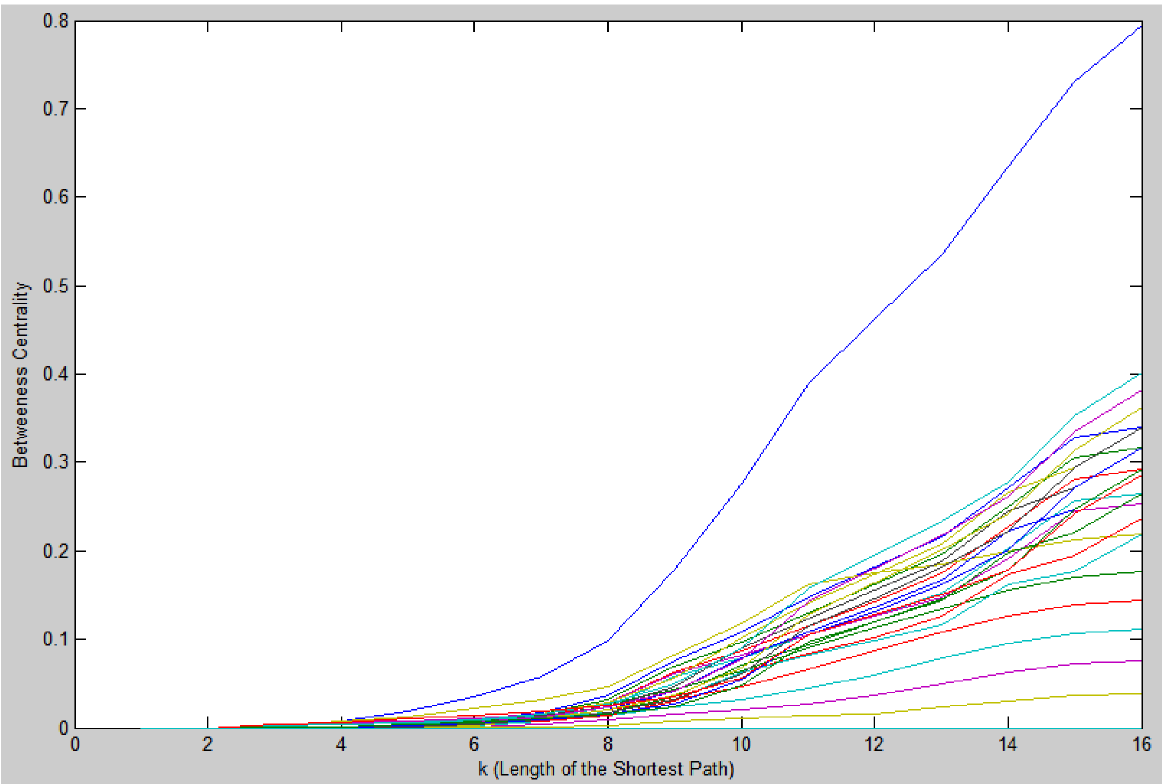

Figure 3. k-Betweenness Centrality for Tree in Figure 2.2

Another interesting example is given in Figure 1.4 where k-betweenness centrality of a particular vertex goes up and down with respect to itself.

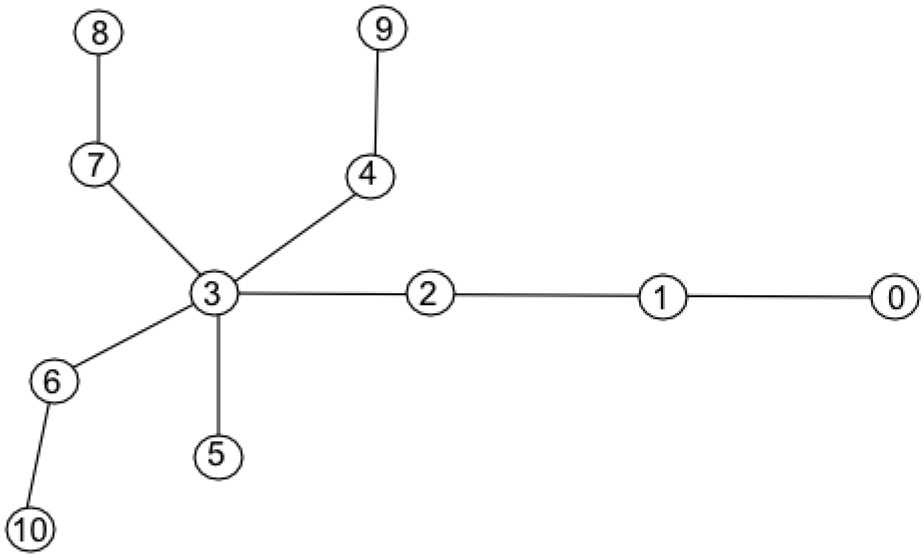

Figure 4. A simple tree in which the k-betweenness centrality of node 1 oscillates up & down with respect to itself. $C_B^2(1) = \frac{1}{15}$, $C_B^3(1) = \frac{2}{32}$ and $C_B^4(1) = \frac{7}{42}$. Therefore we can say that

$$C_B^2(1) > C_B^3(1) < C_B^4(1)$$

We study such kind of behavior as these are the cases where calculating the k-betweenness centrality (C_k^B) will not help us in approximating the betweenness centrality (C^B). Following we provide a construction of such a tree where k-betweenness centrality of nodes oscillates up

& down with respect to each other.

Claim: For every m, there exists a tree T_m and vertices v_m, u_m such that $C_B^k(v_m)$ and $C_B^k(u_m)$ have at least m zig-zags.

Proof: Construction of a tree with k zig-zags in C_B^k between vertices u and v is given below:

1. Let us make a path of length of greater than $2k$.

Figure 5. A simple path consisting of more than $2k$ vertices.

2. Consider a tree given in the picture.

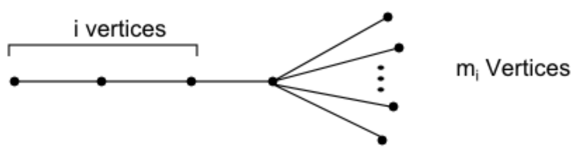

Figure 6. A simple tree where i vertices constitute the stem and m_i are tail vertices.

3. Add a_i to the vertex u and b_i to the vertex v where subscripts of a can take only even values and subscripts of b can take only odd values. Here we deviate from the normal convention of using m_i and denote the trees added to the vertex v as b_i and trees added to the vertex u as a_i.

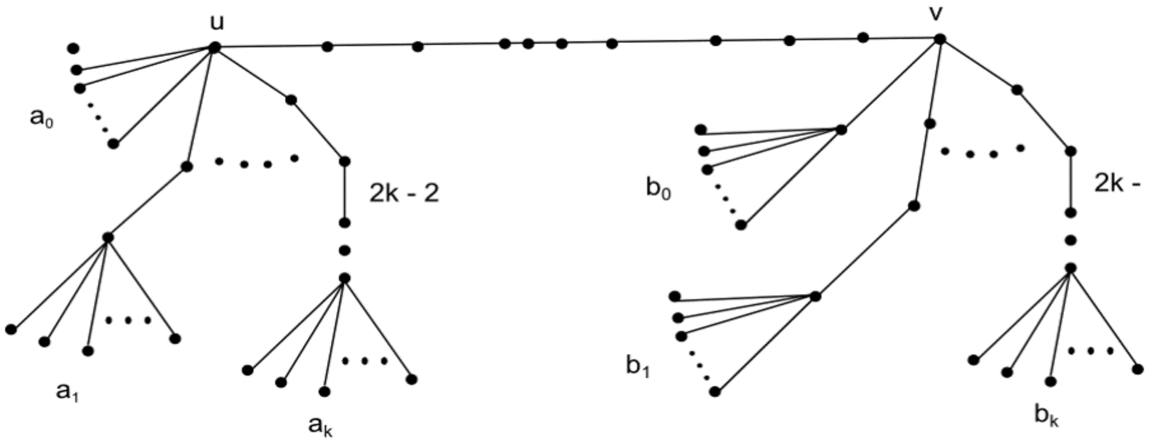

Figure 7. A simple tree where i vertices constitute the stem and m_i are tail vertices.

$P_i(u)$ denotes the number of paths of length i passing through the vertex u.

$P_{2i}(u) \geq a_i$

$P_{2i}(v) \leq O\left((b_1 + b_2 + ... + b_{i-1} + k^2)^2\right)$

$a_i \geq P_{2i}(v)$ and

$P_{2i+1}(v) \geq b_i$

$P_{2i+1}(u) \leq O\left((a_1 + a_2 + ... + a_i + k^2)^2\right)$

$b_i \geq P_{2i+1}(u)$

Here the point of consideration is that the subtree that is being added to the vertex u or v has length and tail vertices chosen in such a way that the total number of paths of length l passing through one vertex in will always be less than the number of paths of length $l+1$ passing through the other. This will cause the k-betweenness centrality of the nodes concerned to oscillate with respect to each other. The resulting tree will have at least k zig-zags in k-betweenness centrality considering nodes u and v. ■

2.3 Complete Binary Tree

Complete binary tree is a binary tree in which each node has two children except the leaf nodes. Due to this property the number of nodes at a particular height h is 2^{H-h} where $0 \leq h \leq H$(taking into consideration the leaf nodes) and the total number of nodes in tree is given by $2^{H+1} - 1$ where H is the height of tree. Here we make the assumption that height of the tree increases from leaf nodes to the top and thus, the root node is at height H.

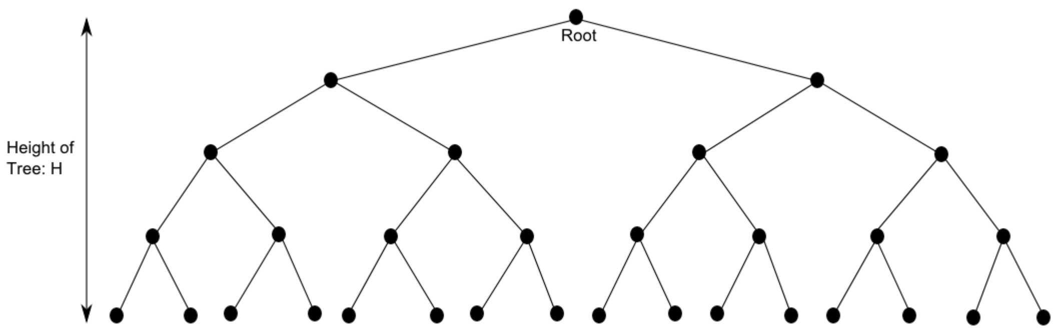

Figure 8. A Complete binary tree of height H. As we can see all the nodes except the leaf node have 2 children. The leaf nodes have the lowest height 0, and the root has the highest, H.

2.3.1 Nodes at a distance d

In this section we provide an explicit formula for calculating the number of nodes at distance d from a particular node, not contained in the subtree of that node.

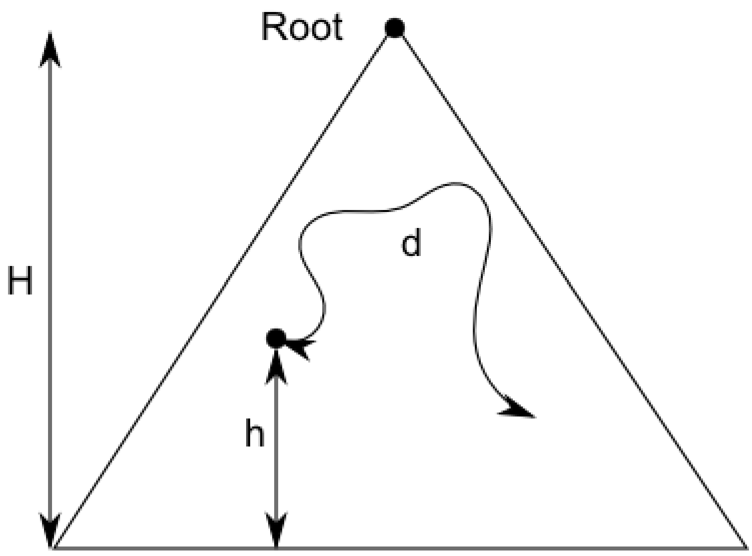

Figure 9. A complete binary tree of height H. We are looking for nodes at a distance d not contained in the subtree formed by that node.

d: Distance d from a particular node.

$\exists$ path of length d if: $1 \leq d \leq 2H - h$

h: Height of the node. $0 \leq h < H$

H: Height of the Tree(calculated from the root).

f: MAX(1, $\lceil \frac{d-h}{2} \rceil$)

g: MIN $(H - h, d)$

<u>Claim:</u> $f \le g$

<u>Proof:</u> As we know $1 \le d$ and $\lceil \frac{d-h}{2} \rceil < d$, so MAX(1, $\frac{d-h}{2}$) $\le d$

And, as $0 \le h < H \Rightarrow 1 \le H - h$,

and $1 \le d \le 2H - h \Rightarrow \lceil \frac{d-h}{2} \rceil \le H - h$

So MAX(1, $\lceil \frac{d-h}{2} \rceil$) $\le H - h$

Therefore f $\le$ g. ■

What we are trying to say in the derivation below is that for calculating the number of nodes we can go some distance in the upward direction, j, and then again go downward, $d - j$, which adds to the total distance d. A point to remember is that we are always talking in terms of complete binary tree ignoring the subtree formed by that node. The bounds for the value of j depends on the height of the tree, height of the node and distance d.

if $g < d$

$$N = \sum_{j=f}^{g} 2^{d-j-1} = \frac{1}{2} \sum_{j=f}^{g} 2^{d-j} = 2^{d-1} \sum_{j=f}^{g} \frac{1}{2^j} = 2^{d-1} . \frac{1}{2^f} \sum_{j=0}^{g-f} \frac{1}{2^j} = 2^{d-1} . \frac{1}{2^f} \left(2 - \frac{1}{2^{g-f}} \right)$$

$$= 2^{d-f} - 2^{d-g-1}$$

if $g = d$

$$N = \sum_{j=f}^{g} 2^{d-j-1} + \frac{1}{2} = \frac{1}{2}\sum_{j=f}^{g} 2^{d-j} + \frac{1}{2} = 2^{d-1}\sum_{j=f}^{g}\frac{1}{2^j} + \frac{1}{2} = 2^{d-1}.\frac{1}{2^f}\sum_{j=0}^{g-f}\frac{1}{2^j} + \frac{1}{2}$$
$$= 2^{d-1}.\frac{1}{2^f}\left(2 - \frac{1}{2^{g-f}}\right) + \frac{1}{2} = 2^{d-f} - 2^{d-g-1} + \frac{1}{2} = 2^{d-f} - \frac{1}{2} + \frac{1}{2}$$
$$= 2^{d-f}$$

Therefore,

$$N = \begin{cases} 2^{d-f} - 2^{d-g-1} & \text{if } g < d \\ 2^{d-f} & \text{if } g = d \end{cases}$$

2.3.2 Total number of paths of length k through a vertex

In this section we provide an explicit formula for calculating the total number of shortest paths of length exactly k passing through a vertex of a complete binary tree. Lets us assume that the vertex is at height h as all the nodes at a particular height will have the same value.

For going a distance d we can go a distance j in the upward direction and then again $d - j$ in

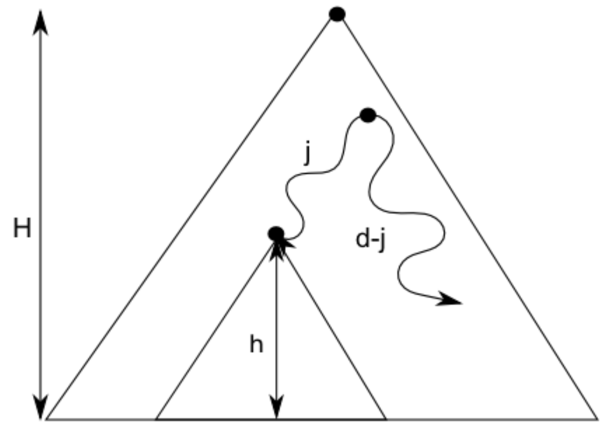

Figure 10. A complete binary tree of height H. The node concerned is at height h and we are looking for nodes at a distance d not in the subtree formed by that node.

the downward direction. The maximum distance we can go in the upward direction is limited to the root of the tree.

$$P_k = \sum_{d=a}^{b} 2^{k-d}. \left(2^{d-f} - 2^{d-g-1}\right) = 2^k \sum_{d=a}^{b} \left(\frac{1}{2^f} - \frac{1}{2^{g+1}}\right)$$

$a = \text{MAX } (1, k - h)$

$b = \text{MIN } (2H - h, k - 1)$

$f = \text{MAX } (1, \lceil \frac{d-h}{2} \rceil)$

$g = \text{MIN } (H - h, d)$

Using the above formula we can find the k-betweenness but the use of Max, Min, Ceiling in the summation presents us with a lot of cases to handle and prevents getting the closed form.

In future we plan to use to use this formula for determining whether Complete Binary Trees exhibit zig-zags in k-betweenness centrality.

2.3.3 Total number of paths through a vertex

In this section we provide an explicit formula for calculating the total number of paths passing through a vertex of a complete binary tree.

Let t_h = number of children nodes of vertex including children, grand children and so on.

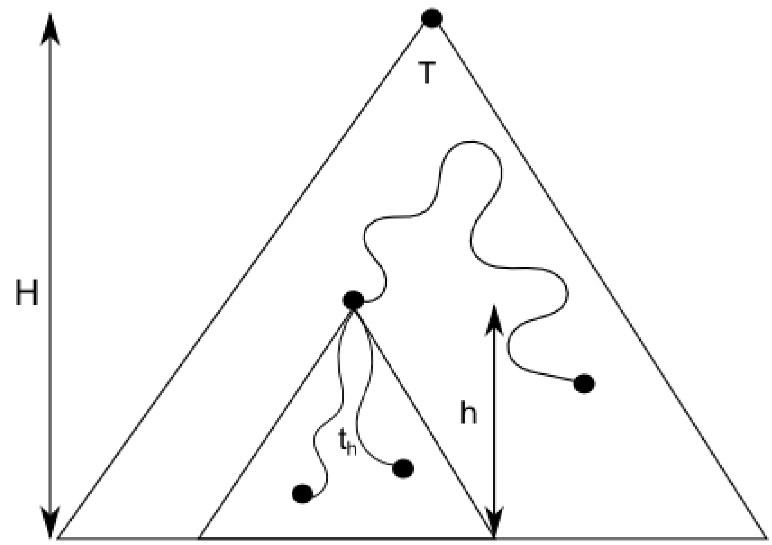

Figure 11. A complete binary tree of height H. T denotes the total number of nodes in the tree whereas t_h denotes the total number of the nodes contained in the subtree not including the node itself.

$t_h = 2^1 + 2^2 + ... + 2^h = 2^{h+1} - 2$

Let T denote the total number of vertices in the tree. $T = 2^{H+1} - 1$.

Therefore, total number of shortest paths through a node at height h is given by

$$P = (2^{H+1} - 2^{h+1})(2^{h+1} - 2) + (2^h - 1)^2$$

where $(2^{H+1} - 2^{h+1})(2^{h+1} - 2)$ denote the total number of paths from the subtree to the rest of the tree and $(2^h - 1)^2$ denote the number paths in the subtree itself.

CHAPTER 3

ANALOGY SOLVER

What is an Analogy?

An analogy is of the from $A : B :: C : D$ which means A is to B as C is to D. For example, Ruler : Line :: Compass : Circle, says that a ruler is used to draw a line and in the same way compass is used to draw a circle. In other words we can say that the features that are used to connect A to B are the same as those that are used to connect C to D. Usually an analogy is given in the form of an example word pair and some options.

```
Given: Ruler : Line
A. Stamp : Letter
B. Period : Dot
C. Key : Door
D. Compass : Circle
E. Thermometer : Degree
where A to E are options to select from.
```

In the above example **To Draw** is the feature that connects **Ruler** to **Line** and the same feature also connects **Compass** to **Circle**.

3.1 A brief introduction to ConceptNet 4.0

It is a large semantic Network in which each node represents a concept. A concept is connected to other concepts using some predefined relations.

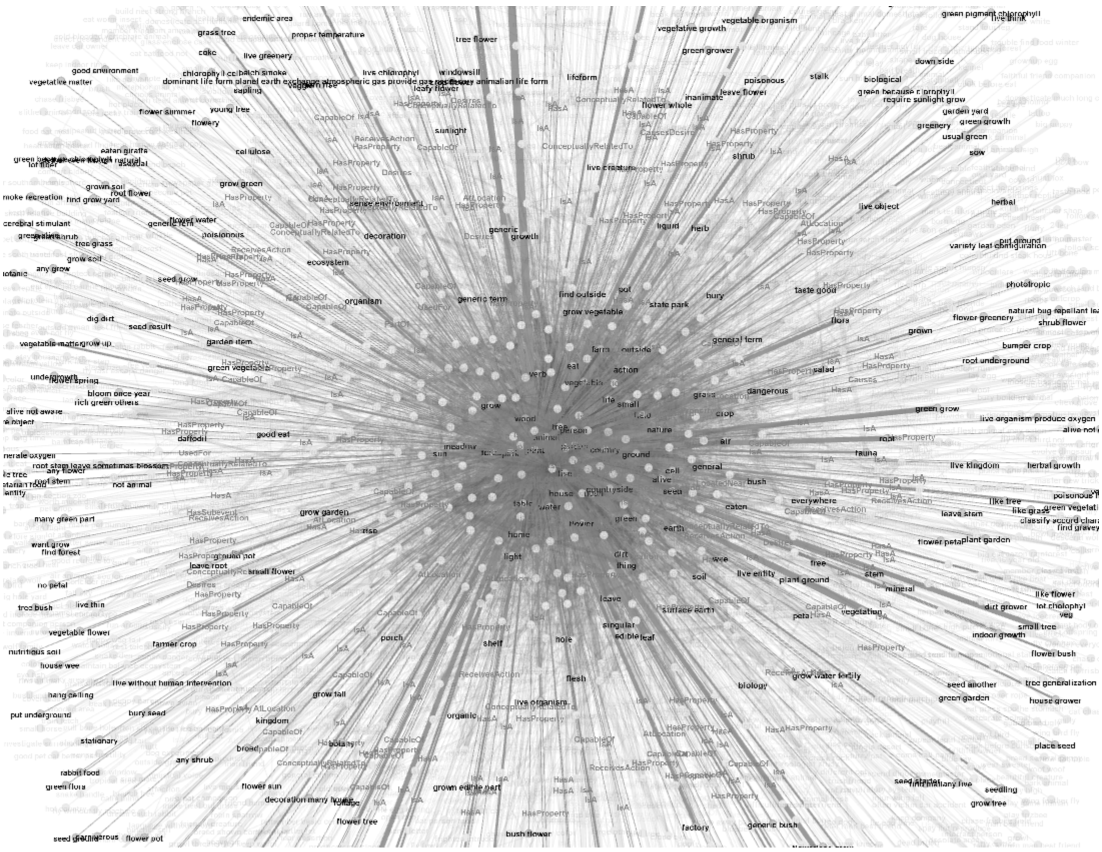

Figure 12. A snapshot of ConceptNet 4.0 graph.

```
For instance,

Concept1: Fish

Concept2:  SeaCreature

Relation: IsA
```

Concept1 and Concept2 represent 2 nodes to the ConceptNet 4.0 which are connected by a relation "IsA". ConceptNet 4.0 permits directed relations which says that that if Concept1 is connected via a particular relation to Concept2. It doesn't necessarily imply that Concept2 is connected to Concept1 via the same relation. It also permits multiple relations between 2 nodes.

For the work contained in this thesis we are working with a sparse multi graph representation of ConceptNet 4.0. There are a total of 279497 concepts in the graph. For more details on the graph and ConceptNet 4.0 refer (4). Some of the simple properties of graph are shown in the table.

Relation	General Idea
HasFirstSubevent	What do you do first to accomplish it?
HasLastSubevent	What do you do last to accomplish it?
HasPrerequisite	What do you need to do first?
MadeOf	What is it made of?
IsA	What kind of thing is it?
AtLocation	Where would you find it?
UsedFor	What do you use it for?
CapableOf	What can it do?
MotivatedByGoal	Why would you do it?
Desires	What does it want?
ConceptuallyRelatedTo	
DefinedAs	How do you define it?
InstanceOf	*What type of thing is it a specific example of?
SymbolOf	
HasA	
CausesDesire	What does it make you want to do?
Causes	What does it make happen?
HasSubevent	What do you do to accomplish it?
HasProperty	What properties does it have?
PartOf	What is it part of?
ReceivesAction	What can you do to it?
InheritsFrom	
CreatedBy	How do you bring it into existence?
HasPainCharacter	*What is the character of pain associated with it?
HasPainIntensity	*What is the intensity of pain associated with it?
LocatedNear	
SimilarSize	

TABLE III

Types of Relation

Concept	Degree
person	19,172
something	2,893
human	1, 794
this	1, 637
child	1, 500
fun	1, 378
water	1, 366
book	1, 241
it	1, 208
man	1, 204
dog	1, 152
money	1, 133
party	1, 128
paint	1, 124
music	1, 123
horse	1, 122
car	1, 114
write	1, 095
house	1, 089
dance	1, 076
food	1, 042
cat	1, 010
exercise	986
animal	971
eat	960
drink	927
home	906

TABLE IV

Top 27 Concepts with respect to degree.

3.2 Algorithms

In this section we provide the pseudo code of algorithms used to solve the analogies. In each section we provide the basic outline of the algorithm.

3.2.1 Baseline Algorithm

The very first algorithm that we would like to mention is the baseline algorithm which is a main component binding together various small algorithms. Only those analogies are tested for which all the concepts are present in the graph.

Algorithm 1 BaseLine Algorithm

// Stem Concept pair is the given concept pair in the Analogy.
// Candidate Pairs are the options given with the Analogy.
$ConceptsPresent \leftarrow$ Check whether all the concepts present in the Analogy question are present in the graph(ConceptNet4.0).
if $ConceptsPresent$ is *true* **then**
  $StemDirectlyConnected \leftarrow$ Check whether the Concept Pair given in the Question are Directly Connected.
  if $StemDirectlyConnected$ is *true* **then**
    $Relations_S \leftarrow$ Get all the relations connecting the stem concept pairs.
    for Each Candidate Concept Pair C_{i1}, C_{i2} **do**
      $Relations_{C_i} \leftarrow$ Get all the relations connecting C_{i1}, C_{i2}.
    $\chi \leftarrow 0$
    for Each $Relations_{C_i}$ **do**
      if $Relations_{C_i} = Relations_S$ **then**
        $\chi \leftarrow \chi + 1$
    if $\chi \neq 1$ **then**
      Run Refinement Algorithms
    else
      Output the matching Candidate.
  else
    Run Spreading Activation based Algorithm

Algorithm 2 Refinement Algorithm 1: NeighborHood Similarity

```
S = S_1, S_2 Stem Concept Pair
for Each Candidate Concept Pair C_i1, C_i2 do
    s_1 ← Calculate the number of Shared Neighbors between C_i1 and S_1.
    s_2 ← Calculate the number of Shared Neighbors between C_i2 and S_2.
    Γ_i ← s_1 + s_2
if Unique Maximum Γ exists then
    return χ ← Maximum value of Γ
else
    return Randomly any Maximum Value
```

3.2.2 Refinement Algorithms

In this section we provide 2 algorithms which help in resolving ties. The first Refinement Algorithm that we have is the **NeighborHood Similarity**. In essence this algorithm computes the number of common/shared neighbors between the respective concepts in the stem concept pair and candidate concept pairs. The candidate which has the highest number of shared neighbors is chosen. Again if we have any ties between the candidates, we randomly choose any candidate which has the maximum value.

The second refinement algorithm that we use is the **Squared Euclidean Distance**.

The idea is that we treat as *features* the ratios of the relations for incoming and outgoing edges in order to characterize the similarity between different concepts. Here the `NormalizedRelationVector` is normalized vector that shows the distribution of edges and has size $2.27 = 54$ where the first 27 are for Incoming Edges and second 27 for Outgoing edges. Each entry in the vector indicates the ratio of the type of edges that are coming into the node or going out form the node. Here

Algorithm 3 Refinement Algorithm 2: Squared Euclidean Distance

$r_{s1} \leftarrow$ `NormalizedRelationVector`(S_1)
$r_{s2} \leftarrow$ `NormalizedRelationVector`(S_2)
for `Each Candidate Concept Pair` C_{i1}, C_{i2} **do**
  $\Gamma_i \leftarrow 0$
  $r_{c_{i1}} \leftarrow$ `NormalizedRelationVector`(c_{i1})
  $r_{c_{i2}} \leftarrow$ `NormalizedRelationVector`(c_{i2})
  $\Gamma_i \leftarrow$ `SquaredEuclidean`$(r_{s1}, r_{c_{i1}})$ + `SquaredEuclidean`$(r_{s2}, r_{c_{i2}})$
if `Unique Minimum` Γ `exists` **then**
  return $\chi \leftarrow$ `Minimum value of` Γ
else
  return `Randomly any Minimum Value`

we calculate the Squared Euclidean Distance between the vectors and choose the Candidate Pair which reduces this distance.

3.2.3 Spreading Activation

When the Concepts in the Stem are not directly connected we use the spreading activation technique. During this phase we perform spreading activation starting with stem concept pair which provides a modified graph as output. This new graph contains activation values for each node. We need to extract **Primary Paths** from this modified graph. By **Primary Path** we mean the paths which receive highest activation. We can use Dijkstra Shortest path algorithm to find primary paths. But to convert the problem of finding paths of highest activation to the problem of finding paths of minimum weight we subtract some initial value, I, from each node's activation value. These paths are of the form `[Stem Concept 1]` $->$ `[Related Via]`

`−> [Concept i] −> ..... −> [Related Via] −> [Stem Concept 2].`

For instance,

```
1: architect (46965, 41287) >-(Related Via) 10-> design (6756, 6175)
  >-(Related Via) 4-> blueprint (119899, 104550)
```

Each path provides an array of relations that can be used to connect the two concepts. These relation are directed. After we have the paths between the stem concepts we try to find the "Same Path" between the candidate pairs. Here "Same Path" implies that the sequence of relations that are used to connect the stem concept pair is also connecting the candidate pair maintaining the direction and order of relations. If more than 1 candidate pairs satisfy this constraint we again resort to refinement algorithms for resolving ties. Only the tied candidate pairs are sent to the refinement algorithms.

3.2.4 Path Similarity

This is a very simple algorithm where we find a measure of similarity between the primary path of the stem concept pair and candidate concept pair. The ides used here is the same as that of the **NeighborHood Similarity** but instead of using the stem concept pairs and candidate pairs as reference we use the concepts that are on the path. This algorithm is only used to resolve ties when we are able to find multiple candidate pairs which are connected by the same **primary path**.

Algorithm 4 Spreading Activation

```
S = S1, S2 Stem Concept Pair
Tie ← 0
ModifiedGraph ← Run Spreading Activation using S1 and S2 as the Starting Nodes.
// After running Spreading Activation nodes of the graph will have
// Activation Values.
for Each Node u ∈ ModifiedGraph do
    ActivationValue(v) ← I−ActivationValue(v)
// Here I is some Initial value which is greater than the Max
// ActivationValue
PrimaryPath ← Run Dijkstra Algorithm to return Minimum Weight Path.
for Each Candidate Concept Pair Ci1, Ci2 do
    Γi ← Check if the same path exists between the Candidate Concept Pairs considering only the type and direction of links.
    if Γi is True then
        Tie ← Tie + 1
if then|Tie| = 1
    return True(Γ)
else
    Run Refinement Algorithms on Tied Options.
```

3.3 The "FIVE" Examples

In this section we provide detailed analysis of 5 sample analogy questions. We present the both the cases: success or failure.

3.3.1 Question 1: Bird `is to` Avian `as` Dog `is to` Canine

We are given the following:

```
Ex] bird (756) :: avian (177790)

A) plant (649) :: tropical (44522)

B) meat (1586) :: carnivorous (46038)
```

Algorithm 5 Path Similarity

Let $P_s = \{s_1, s_2 ...\}$ be the concepts on the Primary Path between the Stem Concept Pair.
for Each Concept C_{i1}, C_{i2} **do**
    Let $P_i = \{c_1, c_2 ...\}$ be the Path between the Candidate Concept Pair C_{i1}, C_{i2}
    $\Gamma_i \leftarrow 0$
    for Each Concept $c_i \in P_i$ **do**
        $s \leftarrow$ NeighborHood Similarity(c_i, s_i)
        $\Gamma_i \leftarrow s + \Gamma_i$
if Unique Maximum Γ exists **then**
    return $\chi \leftarrow$ Maximum value of Γ
else
    return Randomly any Maximum Value

C) snake (326) :: slippery (3830)

D) dog (482) :: canine (41066)

E) lung (6631) :: amphibian (16006)

Answer: D

The Concepts in the question are directly connected.

The answer to the given analogy is: D

This is the case where all the concept pairs are directly connected in the graph via a single edge. Here the relation that is used to connect **Bird** to **Avian** is also used to connect **dog** to **canine**. Also the other candidate concept pairs are directly connected but they don't have such a relation connecting them. So our algorithm directly outputs the answer. A point to note is that no refinement algorithm is used in this case as we don't encounter any ties between the Candidates.

3.3.2 Question 2: Consider `is to` Contemplate `as` Examine `is to` Scrutinize

No Refinement Algorithm is used in this case also.

```
Ex] consider (20870) :: contemplate (1521)

A) smile (1362) :: greet (23369)

B) write (1625) :: compose (26303)

C) complain (30064) :: bicker (128151)

D) examine (22640) :: scrutinize (44370)

E) ignore (20165) :: notice (36883)

Answer: D

The Concepts in the question are directly connected.

The answer to the given analogy is: B
```

In this case we make a mistake as all the concepts are directly connected. But the relation that is used to connect **Consider** to **Contemplate** is not present between **Examine** and **Scrutinize**. Rather different relations, **IsA** and **HasProperty**, connects **Write** and **Compose**. The same relations are present between **Consider** and**Contemplate**, so we output the answer **B**. This type of analogy is typically very hard for ConceptNet4.0 to answer using the current approach. Again as there are no ties no refinement algorithm is used here.

option	concept 1	concept 2	forward rels	backward rels
given	`consider`	`contemplate`	`HasProperty` (20) `IsA` (5)	
(a)	`smile`	`greet`		
(b)	`write`	`compose`	`HasProperty` (20) `IsA` (5) `ConceptuallyRelatedTo` (12)	
(c)	`complain`	`bicker`		
(d)	`examine`	`scrutinize`	`Causes` (18)	
(e)	`ignore`	`notice`		

TABLE V

Directly connected concepts.

concept	# outgoing neighbors	# incoming neighbors
`consider`	30	3
`contemplate`	134	22
`smile`	163	151
`greet`	4	25
`write`	642	146
`compose`	0	2
`complain`	6	15
`bicker`	0	1
`examine`	581	19
`scrutinize`	0	2
`ignore`	1	7
`notice`	52	19

TABLE VI

Number of *distinct* outgoing and incoming neighbors for the various concepts that appear in the analogy `consider/contemplate`.

3.3.3 Question 3: **Weave** is to **Fabric** as **Write** is to **Text**

```
Ex] weave (54095) :: fabric (1644)

A) illustrate (35440) :: manual (102080)

B) hang (29962) :: picture (2047)

C) sew (13162) :: thread (13496)

D) bake (8340) :: oven (8254)

E) write (1625) :: text (4040)

Answer: E


Spreading Activation..There are 105 concepts active (stopOnMerge = YES).

After 1 / 10 passes the labels were merged.


********* Added additional node with index 5872


********* Added additional node with index 8844


********* Added additional node with index 44277


Intermediate nodes = 3

These are: 5872 8844 44277

The paths list contains 5 different nodes.
```

```
The paths are:

  1: fabric (1913, 1644) >-(Related Via) 4-> wool (6425, 5872)

  >-(Related Via) 6 -> weave (62164, 54095)

  2: fabric (1913, 1644) >-(Related Via) 4-> cotton (9729, 8844)

   >-(Related Via) 4 -> weave (62164, 54095)

  3: fabric (1913, 1644) >-(Related Via) 10-> stitch (50513, 44277)

  >-(Related Via) 4 -> weave (62164, 54095)

  4: weave (62164, 54095) >-(Related Via) 6-> wool (6425, 5872)

  >-(Related Via) 3 -> fabric (1913, 1644)

  5: weave (62164, 54095) >-(Related Via) 4-> cotton (9729, 8844)

  >-(Related Via) 18 -> fabric (1913, 1644)

  6: weave (62164, 54095) >-(Related Via) 4-> stitch (50513, 44277)

  >-(Related Via) 10 -> fabric (1913, 1644)


Participating nodes are

 - fabric (1913)

 - wool (6425)

 - cotton (9729)

 - stitch (50513)

 - weave (62164)

0 ] Path Similarity : 0
```

```
1 ] Path Similarity : 0

2 ] Path Similarity : 0

3 ] Path Similarity : 0

4 ] Path Similarity : 5

The answer to the given analogy is: E
```

In this example we don't have the stem concept pair directly connected so we run spreading activation. We get 6 primary paths in this case. We predict the correct answer due to the path similarity algorithm.

3.3.4 Question 4: Abbreviation is to Word as Abstract is to Report

```
Ex] abbreviation (31710) :: word (44)

A) outline (40088) :: story (952)

B) plot (53249) :: fiction (47443)

C) page (5727) :: paper (128)

D) paragraph (32562) :: book (1748)

E) abstract (62995) :: report (23112)

Answer: E


Spreading Activation..There are 237 concepts active (stopOnMerge = YES).

After 1 / 10 passes the labels were merged.


********* Added additional node with index 117
```

```
Intermediate nodes = 1

These are: 117

The paths list contains 3 different nodes.

The paths are:

  1: word (51, 44) >-(Related Via) 14-> it (137, 117) >-(Related Via) 4

  -> abbreviation (35769, 31710)

  2: abbreviation (35769, 31710) >-(Related Via) 4->

  it (137, 117) >-(Related Via) 14-> word (51, 44)


Participating nodes are

 - word (51)

 - it (137)

 - abbreviation (35769)

0 ] Path Similarity : 0

1 ] Path Similarity : 1

2 ] Path Similarity : 56

3 ] Path Similarity : 2

4 ] Path Similarity : 0

The answer to the given analogy is: C
```

Here we answer incorrectly.

3.3.5 Question 5: Skull `is to` Head `as` Skeleton `is to` Body

```
Ex] skull (14433) :: head (9278)

A) heart (12847) :: organ (13307)

B) finger (3032) :: hand (1992)

C) skeleton (7415) :: body (1593)

D) elbow (14539) :: joint (70004)

E) scalp (73877) :: hair (4500)

Answer: C

RefinementAlgorithm 1

NeighborHood Similarities.

A) 7

B) 24

C) 0

D) 2

E) 3

The answer to the given analogy is: B
```

In this case we don't answer correctly as the neighborhood similarity of **Finger** and **Hand** to **Head** and **Skull** is more than compared to any of the candidates. **Finger** and **Organ** has more common neighbors with **Skull** and **Head**. In total they have 24 common neighbors. But at the same time if Refinement Algorithm 2: Squared Euclidean Distance is used, we get the following results:

```
RefinementAlgorithm 2

Squared Euclidean Distance

A) 0.213261

B) 0.433141

C) 0.000000

D) 0.314762

E) 0.108700

The answer to the given analogy is: B
```

This presents us the case where Squared Euclidean Distance finds the answer whereas Neighborhood Similarity does not.

3.4 Results

In this section we present the results of the algorithms on the Data Set. In total 105 questions were tested. We tried different refinement algorithms when faced with tied options. Also 2 variants of **Neighborhood Similarity** were used to resolve ties in the case when more than one candidate pair had same path between them. One was **Neighborhood Similarity** and other was **Path Similarity**. The results are presented below.

Refinement Algo.	**Refinement Algo. in Spreading Activation**	**Correct/Total**
`NeighborHood Similarity`	`PathSimilarity based on NeighborHood Similarity`	28/105
`Squared Euclidean Distance`	`PathSimilarity based on NeighborHood Similarity`	27/105
`NeighborHood Similarity`	`NeighborHood Similarity`	30/105
`Squared Euclidean Distance`	`NeighborHood Similarity`	31/105

TABLE VII

Results of the various combinations of refinement algorithms.

	Algorithm	Score		Algorithm	Score
1	Phrase Vectors	0:382	11	Holonym:member	0:200
2	Thesaurus Paths	0:250	12	Similarity:dict	0:180
3	Synonym	0:207	13	Similarity:wordsmyth	0:294
4	Antonym	0:240	14	Combined [16]	0:450
5	Hypernym	0:227	15	Proposed (SVM)	0:401
6	Hyponym	0:249	16	WordNet [19]	0:428
7	Meronym:substance	0:200	17	VSM [15]	0:471
8	Meronym:part	0:208	18	Pertinence [13]	0:535
9	Meronym:member	0:200	19	LRA [12]	0:561
10	Holonym:substance	0:200	20	Human	0:570

TABLE VIII

Results of the previous similarity measures. Taken from (2).

Table VIII shows results of the previous algorithms.

CITED LITERATURE

1. Pfeffer, J. and Carley, K. M.: k-centralities: Local approximations of global measures based on shortest paths, 2012.

2. Bollegala, D., Matsuo, Y., and Ishizuka, M.: Www sits the sat: Measuring relational similarity on the web. In In Proc. of ECAI08, pages 333–337, 2008.

3. Brandes, U. and Erlebach, T.: Network analysis: Methodological foundations [outcome of a dagstuhl seminar, 13-16 april 2004]. In Network Analysis, volume 3418 of Lecture Notes in Computer Science. Springer, 2005.

4. Diochnos, D. I.: Commonsense reasoning and large network analysis: A computational study of conceptnet 4. CoRR, abs/1304.5863, 2013.

5. Turney, P. D.: Analogy perception applied to seven tests of word comprehension. CoRR, abs/1107.4573, 2011.

VITA

Rahul Kushwaha

Education	**B.S., Computer Science** Career Institute of Technology and Management, MDU 2010
	M.S., Computer Science (*current*) University of Illinois at Chicago, Chicago, IL 2014
Professional	**Graduate Assistant, Office of Vice Chancellor, University of Illinois at Chicago** 01/2013 - Present
	Software Developer Intern, Bloomberg LP, New York 06/2013 - 08/2013
	Senior Software Engineer, Nagarro Software Private Limited, NCR, India 11/2010 - 07/2012