

# University of Cincinnati

Date: 4/19/2016

**I. Prasad Reddy Mendu, hereby submit this original work as part of the requirements for the degree of Master of Science in Computer Science.**

It is entitled:

**Link Prediction in Time-Evolving Graphs**

Student's name: **Prasad Reddy Mendu**

This work and its defense approved by:

Committee chair: Raj Bhatnagar, Ph.D.

Committee member: Nan Niu, Ph.D.

Committee member: Yizong Cheng, Ph.D.



20999

# Link Prediction in Time-Evolving Graphs

A thesis submitted to the

Graduate School

Of the University of Cincinnati

In partial fulfillment of the

Requirements for the degree of

Master of Science

In the Department of Electrical Engineering and Computing Systems

Of the College of Engineering and Applied Sciences

By

Prasad Reddy Mendu

B.E. Osmania University, India, 2011

April 2016

Thesis Advisor and Committee Chair: Dr. Raj Bhatnagar

## Abstract

With the increase in number of social networks and technological advancements in the last two decades, there is vast amount of digital communication happening between people. Most of these communication networks evolve with time and can be represented in the form of graphs. Link Prediction is finding edges that may appear in the future in the network using the current data in the network. Link prediction finds many applications such as “Recommender systems” in social networks like Facebook, Twitter, LinkedIn etc.

Link prediction is a vast area and many link prediction algorithms exist today each catering to its specific purpose. Some of these algorithms deal with the problem of link prediction in Time-evolving graphs and they have comparatively better performance than a random predictor. However, their raw performance, when considered by itself, can still be improved. In our work, we aim to develop an algorithm that not only has comparatively better performance than the random predictor, but also to have very good raw performance.

In this work, the main idea for link prediction is to take into account the past data along with the current data to predict the edges to be formed in the future. To do this, we calculate the conditional probability of two nodes having an edge between them in the future given they have certain feature value. Proximity measures are taken as features between the nodes, and they are calculated for every possible edge. Bayesian inference is used to calculate the posterior probability of an edge to occur in the future, given we have current and past data. Past data is used to calculate Prior probability of edges. We also experimented with the way

Prior probability of an edge is computed by changing the way in which it is computed currently, and observed the algorithm's behavior with this version of Prior probability.

Using this methodology, our algorithm is tested against different types of datasets and the relevancy measures such as precision, recall, and specificity are calculated to evaluate the performance of link prediction algorithm. A detailed analysis of the results confirms that the proposed link prediction algorithm performs better than the existing algorithms with significantly improved precision and recall values. Our algorithm is also tested by using an alternative second method to calculate the Prior probability of an edge and interestingly, it showed that even though the performance of our first proposed method is better than this, the second version has better precision and recall values than some of the existing algorithms.



## **Acknowledgement**

Firstly, I would like to thank each and every one who stood with me during this whole journey. I will be forever indebted to Dr. Raj Bhatnagar for acting as my thesis advisor and for his guidance and constant supervision and support in completing this thesis. A special thanks to Dr. Nan Niu and Dr. Yizong Cheng for taking time out of their busy schedules to be a part of my thesis committee. Finally, I would like to thank my family for giving me all the freedom to study what I want to and for standing with me and believing in me during the hard times.

# Table of Contents

<b>Chapter 1: Introduction</b>	<b>1</b>
1.1 Motivation	1
1.2 Datasets	2
1.3 Overview of Other Chapters	4
<b>Chapter 2: Related Work</b>	<b>5</b>
2.1 The Link Prediction Problem for Social Networks	7
2.2 Using Bayesian Net Graphs to understand networks	10
2.3 Human Mobility, Social Ties, and Link Prediction	11
2.4 Link Prediction Approach to Collaborative Filtering	12
2.5 Using Markov Chains for Link Prediction in Adaptive Web Sites	13
<b>Chapter 3: Link Prediction Algorithm</b>	<b>15</b>
3.1 Terminologies	15
3.1.1 Bayesian inference	15
3.1.2 Conditional Probability	15
3.1.3 Graph	16
3.1.4 Directed and undirected graph	16
3.1.5 Evolving graph	17
3.1.6 Weighted graph	17
3.1.7 Proximity measures	17

3.1.8	Relevancy Measures	18
3.1.9	Iteration	18
3.2	Link Prediction Algorithm using Bayesian inference	18
3.3	Our Algorithm version-2: different method of calculating Prior Probability	37
<b>Chapter 4:</b>	<b>Experimental Setup and Results</b>	<b>38</b>
4.1	Datasets	38
4.1.1	Enron email database	39
4.1.2	Facebook WallPosts dataset	39
4.2	Data extraction	39
4.3	Parameters	41
4.4	Relevancy measures	43
4.5	Results	44
4.5.1	Enron email dataset - Comparing our algorithm results with results of Nowell's algorithm	44
4.5.2	Facebook WallPosts dataset - Comparing our algorithm results with the results of Nowell's Algorithm	57
4.5.3	Enron dataset – Comparing results of our algorithm with different version of Prior Probability with the first version	91
4.5.4	Comparing Results of second version of our algorithm with Nowell's Algorithm	112
4.5.5	Results of our algorithm with different version of Prior Probability	124



4.5.6	Our Algorithm vs second variant of Nowell's	124
4.5.7	Our Algorithm vs Nowell's third variant of Nowell's	127
<b>Chapter 5: Conclusion and Future Work</b>		<b>131</b>
<b>References</b>		<b>133</b>
<b>List of Tables</b>		
4.1	Enron Data Set Information	2
4.2	Facebook WallPosts Data Set Information	3
<b>List of Algorithms</b>		
1.	Getting the past, evidence, and test data	20
2.	Build Adjacency Matrix for Past Data	21
3.	Get the Prior Probability Matrix from the Adjacency Matrix	22
4.	Calculate all possible edges that can be formed between the nodes present in evidence data	24
5.	Calculate the binning criteria by calculating feature values for the evidence and past data	25
6.	Calculate the feature values for all possible edges and assign them respective bin values	28

7. Calculate the probability of a possible edge to be present in the test period using Bayesian inference	29
8. Calculate the number of possible predictions that can be made	32
9. Predict edges	33
10. Calculate relevancy measures	36

# Chapter 1

## Introduction

A communication network among any groups of people can be represented in the form of graphs, and the communication networks are ubiquitous in our society. This network data is often represented in the form of graphs, which are then used to study further about the domain from which the data is derived. With the growing popularity of social network data, we are now able to get large graph datasets which can be used to discover valuable social characteristics by “Graph Mining”. While graph mining covers wide variety of topics, link prediction is one of the important areas of graph mining.

Link prediction aims to use the current graph data to predict edges that may be formed in the future. Applications of link prediction can be seen everywhere on a daily basis. For example, Facebook graphs of people can be used to predict if there is a chance of that any two people will become friends in the future. Similarly, ‘People you may know’ on LinkedIn is an application of link prediction. However, link prediction is not just limited to social networks. It can be applied to any graph which evolves with time. For example, email communication network of Enron is used to test our algorithm.

### 1.1 Motivation

The motivation comes from the need to extract meaningful knowledge from the large amount of graph data that is available in digital form for a large number of domains. Also, link prediction is a vast area with so much room to experiment the way the link prediction

problem is handled. Both of the aforementioned things inspired me to pursue this problem of link prediction in time evolving graphs, and research on if we can try to solve the problem differently from the existing approaches and improve the performance of predictions.

There are quite a few existing works that dealt with the problem of link prediction, and most of them use proximity measures between the nodes in a graphs for solving the problem of link prediction. John Kleinberg's work [1] for solving the link prediction problem is amazing because it was able to prove that using the values of proximity measures alone for predicting the edges outperforms the random predictor as well as several other simple approaches.

We have put some thought into what if the past structure of the graph is used along with the current data, and also to find out if we can use the probabilistic knowledge from the past behavior of the graph in the problem of link prediction. Bayesian inference is one of the key ideas on which our algorithm is based. The reason is that we believe that using the past data in addition to the current data to predict the future edges is an effective way to solve this problem. Bayesian inference has numerous applications in various fields, including science, engineering, philosophy, medicine, law, and medicine.

## **1.2 Datasets**

### **Enron email dataset:**

This is the first dataset that we have tested for our link prediction algorithm. It consists of over 500,000 email communications among 151 employees of Enron Corporation over a period of three years. Firstly, we define the time periods for what we consider as the past,

the current evidence, and test data for prediction before running our algorithm against a dataset. In this case, we have decided to take the two months current data as evidence data, and the six months before the current period as the past data, which is used to predict the edges for the next two months.

### **Facebook WallPosts dataset:**

Once we tested our algorithm with the first dataset, we validated our algorithm against this real-world Facebook dataset collected from about 40,000 users over the period from September 2004 to January 2009 [13]. We used only that data period for which we have consistent amount of data, i.e., we ignored the initial periods during which there isn't much data, and also the last few months of data for which there is too much data. In total we omitted around 23 months' data out of 51 months and considered the remaining 28 months' data for running our algorithm. This dataset contains information of when a person posted something on another person's wall. Having timestamp information, the dataset is very crucial since our algorithm is mainly dependent on defining a past data period to compute the prior probability values of edges getting connected. It should be noted that this dataset is slightly different from the first dataset in one major aspect. There are edges from one node to itself, which indicates that a person has posted a Facebook status at that particular timestamp value. So, apart from predicting if a person posts on someone else's wall during test period, we are also interested in whether a person posts on his own wall, i.e., keeping a status message.

In total, there are 46,952 unique nodes in this dataset which has 876,993 records over the period of September 2004 to January 2009. However, since the data is sparsely populated

in the first few months and very densely populated in the last few months, we decided to use the data present in the middle 20 months which have almost equally distributed data. From this data, we extracted three different datasets by randomly selecting 400, 600, and 800 nodes and then extracting all the edges which contain the respective randomly picked nodes. The algorithm was executed on these three subsets of the dataset. Also, a second version of algorithm which uses a different version of prior probability computation was also executed with these datasets.

### **1.3 Overview of other chapters**

Using our approach for solving the problem of link prediction we obtain good link prediction results for both the Enron and the Facebook WallPosts datasets. Since we also take the past structure and edge labels of the graph along with the current structure and edge labels of the graph into account, we obtain better precision and recall value when compared with the other existing approaches. In the following chapters, we discuss related work in chapter two, our algorithm in detail in chapter three, experiments and results in chapter four, and will conclude with future work and conclusion in the fifth chapter.

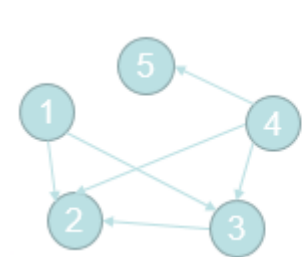
## Chapter 2

### Related Work

This section discusses the related research work done in the area of link prediction and how our work relates to it. Link prediction is an important field where significant amount of research has already been done. However, there are some sub-areas like link prediction in evolving graphs that still needs to be explored further. One reason for limited research activity in the field of link prediction in evolving graphs is that there is not much evolving social network graph data along with timestamps available in the public domains for research.

We use the term “proximity measure” in this document to represent a measure of how similar two nodes are in a given graph. The symbol ‘ $\Gamma$ ’ used below represents the number of neighbors of a particular node in the network. Proximity measures used in this thesis are explained below:

Common Neighbors: “Common Neighbors” of two nodes ‘ $x$ ’ and ‘ $y$ ’ in a network is the number of neighbors that are common to both ‘ $x$ ’ and ‘ $y$ ’.

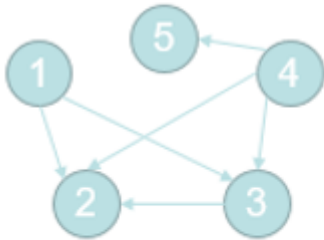


$$CN(x, y) = |\Gamma(x) \cap \Gamma(y)|$$

$$CN(1,4) = |(2,3) \cap (2,3,5)| = 2$$

Jaccard's coefficient: This similarity metric compares the similarity of two nodes based on their shared neighbors in the following way.

$$\text{Jaccard's coefficient}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

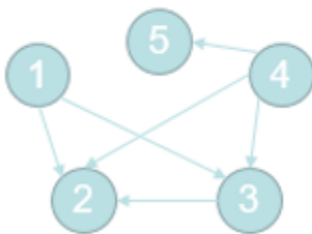


$$\begin{aligned} \text{Jaccard's coefficient}(1,4) &= \frac{|(2,3) \cap (2,3,5)|}{|(2,3) \cup (2,3,5)|} \\ &= \frac{|(2,3)|}{|2,3,5|} \\ &= 2/3 \end{aligned}$$

Preferential Attachment: It is based on the idea that the probability that a new edge to be formed in the future has node  $x$  as an endpoint. The value is proportional to number of neighbors the node currently has.

$$\text{Preferential Attachment}(x, y) = |\Gamma(x)| \cdot |\Gamma(y)|$$

$$\begin{aligned} \text{Preferential Attachment}(1,4) &= |\Gamma(1)| \cdot |\Gamma(4)| \\ &= 2 \cdot 3 \\ &= 6 \end{aligned}$$



Adamic Adar Measure: This measure weighs rarer features more heavily when determining whether two nodes are similar. Adamic Adar measure for two nodes  $x$  and  $y$  is defined as:



$$\text{Adamic Adar Measure}(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} 1/\log|\Gamma(z)|$$

$$\text{Adamic Adar Measure}(1,4) = \sum_{z \in \Gamma(1) \cap \Gamma(4)} 1/\log|\Gamma(z)|$$

$$= (1/\log|\Gamma(2)|) + (1/\log|\Gamma(3)|) = 2/\log(3)$$

## 2.1 The Link Prediction Problem for Social Networks by David Liben-Nowell and Jon Kleinberg

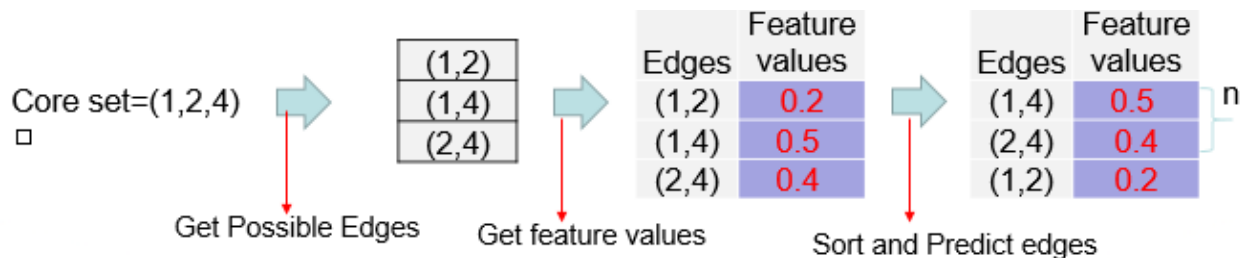
David Liben-Nowell and Jon Kleinberg analyzes the problem of link prediction in social networks in this work [1]. This work is aimed at predicting future edges given the present state of network using the proximity measures directly. It is of utmost importance to us since we refer this algorithm to benchmark our results. Their experiments on large co-authorship networks suggest that information about future interactions can be extracted from network topology alone. These proximity measures when used for link prediction can outperform the direct measures.

A summary of the work presented in this paper is as follows:

- The proximity measures used here are graph distance, common neighbors, Jaccard's coefficient, Adamic/Adar, preferential attachment, Katz, hitting time, rooted PageRank, and SimRank.
- Suppose that the given social network is represented in the form of a graph  $G=\langle V,E \rangle$  in which each edge represents an interaction between nodes at a particular time  $t(e)$ .

Multiple interactions between two nodes at different timestamps are represented as parallel edges.

- From the graph  $G$ , get the subgraph  $G(t,t')$  and define a training interval  $G(t_0,t_0')$  and test interval  $G(t_1,t_1')$  such that  $t_0 < t_0' < t_1 < t_1'$ .
- The aim is to get the proximity measures for all possible future edges that can be formed between the nodes present in a training interval and use those values to predict the new edges that will be formed in the test interval.
- To make better predictions, the test set of nodes is restricted to nodes which are active throughout both the periods, i.e., in the co-authorship network it refers to the authors who are active throughout the training and also in the test period. The set 'Core' is defined as the set that contains the set of nodes that meet the 'threshold' in both the training and test intervals.
- For example, if the threshold is defined as '3', the set Core will consist of all the authors who have coauthored at least 3 papers in both the training and the test periods. And this algorithm aims at predicting the future edges likely to be formed between the authors present in 'Core' in the test period.



The process by which the performance of a link predictor is calculated is explained below:

- a) Each measure outputs a ranked list of pairs in  $(AxA)-(E_{old})$ , which are the predicted new collaborations in decreasing order of confidence, where,
- A – Vertices present in training interval,
- AxA – all possible vertex combinations, and
- $(AxA)-(E_{old})$  – all possible future vertex combinations.
- b) All possible vertex combinations from the set 'core' can be obtained from [CoreXCore].
- c) Link predictor outputs the new possible predictions in decreasing order of confidence (from  $AxA-E_{old}$ )
- d) Since this algorithm is more interested in knowing the possible future edges between the elements of set 'Core', the intersection of  $(AxA- E_{old})$  and CoreXCore is computed, which gives all the new possible edges that can be predicted between the nodes present in set 'core' in decreasing order of confidence
- e)  $E_{new}$  denotes the set of all  $\langle u,v \rangle$  combinations that are present in training data that don't have an edge in the training interval but have it in the test interval
- f) Since we are interested more in the new predictions among the elements of set 'core',  $E_{new}^* = E_{new} \text{ (intersect) (CoreXCore)}$  is computed which are new edges that are formed between elements of Core in the test interval.
- g) Finally, to measure the accuracy of a proximity measure for link prediction,  $((AxA- E_{old}) \text{ intersect CoreXCore}) \text{ (intersect) } (E_{new}^*)$  is computed, which gives the number of correct link predictions

- h) To represent this link predictor quality, the random predictor is used as a baseline, which predicts the randomly selected pairs of vertices which doesn't have an edge in the training interval as the likely new edges to be formed in the test interval.
- i) The experimental results have proved that most of the proximity measures when used directly in the link prediction, have significantly outperformed the random predictor.

To summarize, the authors of this work [1] David Liben-Nowell and Jon Kleinberg have proved that directly using the proximity measures for the future link prediction outperforms more direct measures like random prediction and lays foundation for the future work that can aim at increasing the link predictor accuracy.

Although this research has proved that using the direct measures by themselves can give a good accuracy for link prediction, it did not explore the idea of analyzing the structure of past data, which we have done in our algorithm, and have obtained improved results.

## **2.2 Using Bayesian Net Graphs to understand networks**

In the work by Alden Goldenberg and Andrew W. Moore [2], Bayes Nets were obtained using the SBNS (Screen based Bayes Net structures search) algorithm, which were then used to analyze the structure of the graphs. This research also suggested that there is scope for exploring the usage of Bayesian Networks to analyze social circles.

The SBNS algorithm is a two stage process. In the first stage, which is called the Local Screening, SBNS performs a Bayes Net structural search on each of the small subsets of

variables defined by Frequent Sets. The resulting local structures comprise the restrictive pool of edges from which the global Bayes Net will be constructed at the second stage. Experimental results have proved that analyzing the Bayes Net graphs of simple network example has provided a deeper understanding of the data and stated that these Bayes Nets can further be explored.

While the authors used Bayes Net to analyze the Bayes Net graphs of network sample, we are taking this research further by using it for link prediction in our algorithm.

### **2.3 Human Mobility, Social Ties, and Link Prediction**

Research has also been done to improve our understanding of how individual mobility patterns shape and impact social networks by Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Barabasi. This research [3] has concluded that the mobility patterns in a social network greatly affect the formation of new edges in the graph. For this purpose, a dataset which contains trajectories and communication records of these 6 Million mobile phone users has been used.

Although the proximity measures when used directly are a good indicator of how close the nodes in a graph are, the question of whether analyzing their mobility patterns has superior predictive power has been explored in this research. It has been found out that the mobility measures alone yielded a superior predictive power compared to using the proximity measures directly.

Even though the research was able to prove that analyzing the mobility measures is very helpful in the link prediction, it did not explore the area of analyzing the graph structure which we did in our algorithm.

## **2.4 Link Prediction Approach to Collaborative Filtering**

Research has been done by Zan Huang, Xin Li, and Hsinchun Chen to see whether link prediction approach can be used in some of the recommender systems. This research [4] talks about how a specific link prediction approach greatly outperforms the standard collaborative filtering approaches that are used in the recommender systems.

The user-item interactions are represented as unweighted edges in a bipartite user-item graph  $G$ . For each unconnected user-item in the graph, the six proximity measures given below are calculated and used as scores, which in turn are used to predict the new user-item links.

- Common Neighbors
- Jaccard's
- Preferential Attachment
- Adamic/Adar
- Graph Distance
- Katzp

The results of this research indicated that the link prediction approaches and network analysis in general are important directions that need to be explored to improve the existing collaborative filtering algorithms.

## 2.5 Using Markov Chains for Link Prediction in Adaptive Web Sites

Markov chain is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. In this research by Jianhan Zhu, Jun Hong, and John G. Hughes [5], Markov chains are used to determine which website link the user is likely interested in clicking next. This basically sums up to the problem of link prediction with the web pages as nodes and transitions as edges. And the user's navigational history is used to build a transition matrix from which the probability that the user will go a specific web page is calculated.

Initially, a transition probability matrix is calculated from the given set of nodes, where probability of transiting from node  $i$  to another node  $j$  is the fraction of traversals from  $i$  to  $j$  over total number of traversals from  $i$  to all the nodes in the graph. Once the transition probability matrix is computed, it is compressed by using the spears algorithm. A transition matrix compression algorithm is used to cluster pages with similar transition behaviors together for efficient link prediction. Experiments were performed using a weblog file recorded on a university website and this link prediction is attached to a prototype called 'ONE' (Online Navigation Explorer) which lets the users navigate the university website.

The feedback from the test subjects is very positive and they have spent less time to find the information that interests them using ONE prototype, thereby proving that link prediction using this approach is giving better results. Even though this research was able to prove that Markov chains can be used for predicting the next web link the user is likely to click

from the current page, the link prediction problem for social networks using a similar model still isn't explored. And we aim to use the past data in our algorithm to predict the future edges.



## Chapter 3

### Our Proposed Approach

This chapter gives a detailed explanation of our methodology for approaching the problem of link prediction. The aim of the link prediction algorithm is to predict the future edges in an evolving graph based on the given past data.

#### 3.1 Terminology

This section covers the basic terminology used in the algorithm and presents a formal description of the problem we are trying to solve.

##### 3.1.1 Bayesian inference

Bayesian inference is a method of statistical inference in which Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available.

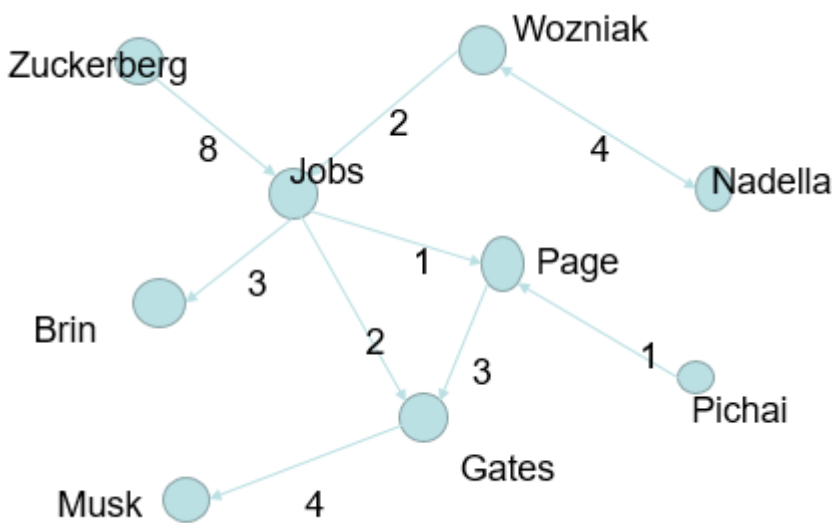
##### 3.1.2 Conditional Probability

Conditional probability is a measure of the probability of an event given that (by assumption, presumption, assertion or evidence) another event has occurred. If the event of interest is A and the event B is known or assumed to have occurred, "the conditional probability of A given B", or "the probability of A under the condition B", is usually written as  $P(A|B)$ , or sometimes  $P_B(A)$ .

For example, the probability that any given person has a cough on any given day may be only 5%. But if we know or assume that the person has a cold, then they are much more likely to be coughing. The conditional probability of coughing given that you have a cold might be a much higher 75%.

### 3.1.3 Graph

A graph consists of a finite set of vertices and edges and can be represented in the form of  $G=\langle V,E\rangle$  where  $V$  is a set of vertices and  $E$  is a set of edges between vertices. An example graph is given below:



### 3.1.4 Directed and undirected graph

The graphs whose edges are directed are called directed graphs, and the graphs whose edges aren't are directed are called undirected graphs. In case of directed graphs, please note that the edge from node 1 to node2 is treated differently from the edge from node 2 to node 1.

### 3.1.5 Evolving graph

Evolving graph is a graph in which new nodes get added over the time. Many networks in the real life can be considered as an evolving graph since new nodes get added to the network as the time progresses. For example, a network graph of a person's Facebook friends can be considered as an example since number of friends that person will connect in Facebook increases as the time progresses.

### 3.1.6 Weighted graph

A graph whose edges are given some weight value are called weighted graphs.

### 3.1.7 Proximity measures

Proximity measures are used to measure the proximity or how close two nodes are in a graph.

The four proximity measures used in this work are Common Neighbors, Jaccards, Preferential attachment, and Adamic Adar. Please find the detailed explanation below:

**Common Neighbors:** The number of neighbors which are common to both x and y [7]

$$CN(x, y) = |\Gamma(x) \cap \Gamma(y)|$$

**Jaccards:** It measures the probability that both the nodes x and y have a feature f, for randomly selected feature x and y has [8]

Features here are taken as neighbors of node

$$Jaccards(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

**Preferential Attachment:** It is based on the idea that the probability that a new edge to be formed with one of the nodes as 'x' is directly proportional to number of neighbors of 'x' [7] [9] [10] [7] [11]

$$Preferential Attachment(x, y) = |\Gamma(x) \cdot \Gamma(y)|$$

**Adamic Adar measure:** It refines the simple counting of common features by weighing rarer features more heavily [6]

$$Adamic Adar measure(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} 1/\log|\Gamma(y)|$$

### 3.1.8 Relevancy Measures

Relevancy measures are used to assess how good the results of an algorithm are. F-1 Measure, Precision, Recall, and Specificity can be considered as relevancy measures.

### 3.1.9 A Test Iteration

For our terminology, we considered one run of our algorithm for an input dataset as one test iteration. For example, we use the 6 months past data and 2 months evidence data to predict the links in the next 2 months test data. We executed a total of 20 iterations when testing our algorithm with input as Enron email dataset and of 11 iterations for the Facebook Wallposts dataset.

## 3.2 Link Prediction Algorithm using Bayesian inference

Please find the step by step approach of how this link prediction algorithm using Bayesian inference is implemented below. Detailed explanation about each step is given further below.

For each iteration in the dataset

- 1) Read the past data, evidence data and the test data(the data for which we are trying to predict edges)
- 2) Build the adjacency matrix for past data
- 3) Get the probability matrix by adding 1 to each element of adjacency matrix (so that every pair of nodes will have some prior probability) and then column normalize the adjacency matrix
- 4) Get the nodes present in evidence data and calculate all the possible edges that can be formed between those nodes
- 5) Get the feature values for all the nodes present in past and evidence data. After eliminating the outliers, divide the range of the features into fixed number of bins
- 6) Calculate the feature values for all possible edges from step 4 and assign the feature values to respective bins
- 7) Using the feature values from step 6, calculate the Bayesian probability for all the possible edges from step 4
- 8) Get the number of possible correct predictions that can be made by analyzing the test data (i.e., the edges present in test data whose nodes are present in evidence data)
- 9) Using the Bayesian probabilities of all possible edges from step 7, take the edges with top X% of the probability as the possible edges that are predicted by this algorithm

10) Calculate True Positives, True Negatives, False Positives and False Negatives.

11) Using the values from step 10, calculate Sensitivity(recall/True Positive rate), Specificity(True negative rate) and Precision

End

Using the metrics calculated from all the iterations, plot the graphs.

Each step given above is explained in detail with an example below:

### Step 1: Getting the past, evidence, and test data


From the dataset, based on how we define the time intervals of these past, evidence, and test data, we will extract the respective data to use it as inputs for our algorithm. Each record in the data will be in the form of:

Node1-Node2-edgeWeight

Which implies that there's an edge from Node1 to Node2 with weight 'edgeWeight'.

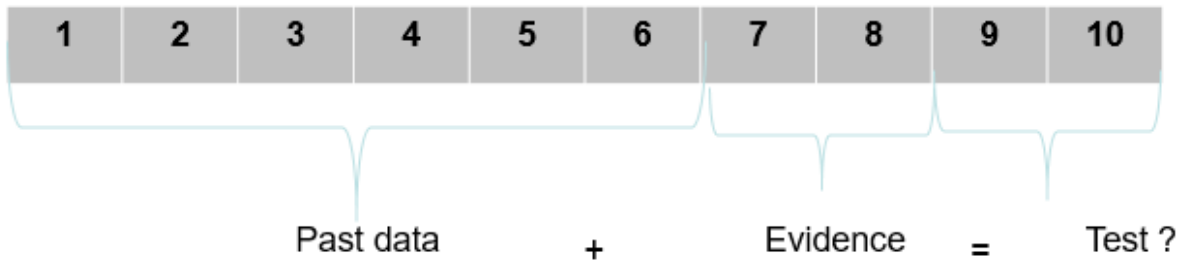
### Example:

S.no	R.no	Total
1	2	3
3	4	2
2	3	1
1	3	5



1	2	3
3	4	2
2	3	1
1	3	5

For our experiments, we defined the past data interval as 6 months, evidence data as next 2 months, and we try to predict next 2 months of test data.



**Step 2: Build Adjacency Matrix for Past Data**

Using the past data, create an adjacency matrix such that each edge of the past data is mapped to an element in the matrix with its edge weight.

Adjacency(i,j) contains weight of the edge between 'i' and 'j'.

S.no	R.no	Total
1	2	3
3	4	2
2	3	1
1	3	5

→

0	3	5	0
0	0	1	0
0	0	0	2
0	0	0	0

**Input**

*Ep= edges in the Past data in the form of  
(node1,node2)Ee= edges in the Evidence data*

**Output**

*Adjacency Matrix-Adj*

**For each** *edge e present in Ep*

**If** *nodes of the edge are present in Ee*

*Adj(node1,node2) = weight of the edge*

**End**

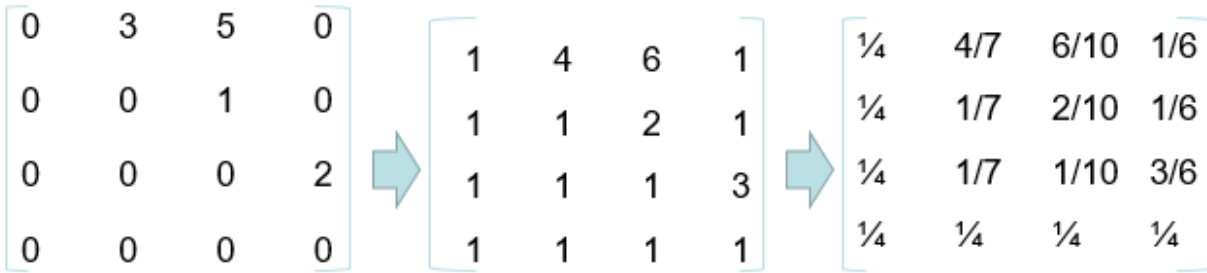
**End**

**Step 3: Get the Prior Probability Matrix from the Adjacency Matrix**

To calculate prior probability of the edges, add 1 to the all of the elements of adjacency matrix from the step two that correspond to possible edges, and then column normalize the matrix. Now the resultant matrix will have prior probabilities corresponding to the respective edges. The reason for adding 1 to all the elements is to make sure that each and every possible edge will have some prior probability, because if it doesn't, the posterior probability will be equal to zero.

Please look at the calculation given below that first adds 1 to the elements of adjacency matrix and column normalizes after that.





Add 1

Column normalize

**Input**

*Adj= Adjacency matrix*

*Pe= Possible edges*

**Output**

*Prior probability matrix*

**For each** edge present in *Pe*

*Add 1 to respective element in Adj;*

**End**

**For each** column present in *Adj*

*Calculate the sum of the elements;*

*Divide each element in that column by the sum;*

**End**

*Store the resultant matrix as prior probability matrix;*

**Step 4: Calculate all possible edges that can be formed between the nodes present in evidence data**

Using the evidence data, get the unique nodes present in evidence data. From the unique evidence nodes, calculate the possible edges that can be predicted.

- If  $n$  nodes are present in the evidence data, there will be  $nC2$  possible edges that can be predicted.
- For example, if the nodes present in evidence data are [2,3,4,6], the possible edges are:

2	3
2	4
2	6
3	4
3	6
4	6

***Input***

*Evidence data*

***Output***

*possibleEdges= All possible edges*

*uniqueEvidence= Unique evidence nodes*

*Get the unique evidence nodes from the evidence data;*

*Store them in uniqueEvidence;*

***For each*** node 'node1' from 1<sup>st</sup> element to last element in uniqueEvidence

***For each*** node 'node2' from 2<sup>nd</sup> element to last element in uniqueEvidence

*Add the (node1,node2) as a possible edge;*

*And store them in possibleEdges matrix;*

***End***

***End***

**Step 5: Calculate the binning criteria by calculating feature values for the evidence and past data**

Calculate the feature values for all the data present in the evidence and past data combined. Treat the top 1% of the elements as outliers and divide the feature values range into specific bins. The main purpose of dividing feature values into bins is to convert the continuous variable

range of the feature values to discrete range, which is later used when calculating the probability using Bayesian inference. For example, binning criteria for one of the iterations looks like:

<b>Bin number</b>	<b>Lower bound</b>	<b>Upper bound</b>
1	0	0.166666667
2	0.166666667	0.333333333
3	0.333333333	0.5
4	0.5	0.666666667
5	0.666666667	0.833333333
6	0.833333333	1

**Input**

*Past data*

*Evidence data*

*Number of bins*

**Output**

*Bin range matrix*

*Get the feature values for past data and evidence data;*

**For each feature**

*Eliminate the outliers in the feature values obtained above;*

*Calculate the minimum and maximum of the remaining values;*

*Create a linearly spaced vector between the minimum and maximum values;*

**End**

**For each feature**

**For each element in the linearly spaced vector**

*Treat the current element as the minimum boundary;*

*And the next element as the maximum boundary;*


**End**

**End**

**Step 6: Calculate the feature values for all possible edges and assign them respective bin values**

In this step, we will calculate the feature values for all possible edges, and then change the feature value into respective bin number using the Bin range matrix obtained from the step above. For example,

Node1	Node2	Feature
2	3	0.45
2	4	0.1
2	6	0.7
3	4	0.9
3	6	0.31
4	6	0.15



Node1	Node2	Feature
2	3	3
2	4	1
2	6	5
3	4	6
3	6	2
4	6	1

**Input**

*All possible edges with feature values*

*Bin range matrix*

**Output**

*Feature values classified with bin numbers*

**For each** row present in the possible edges

*Using the bin range matrix, find out which bin does the feature value fall into;*

*Replace the feature value with the bin number;*

**End**

**Step 7: Calculate the probability of a possible edge to be present in the test period using Bayesian inference**

This step forms the crux of our algorithm. Instead of using the feature value directly for predicting edges, we also take the past data into account. The data from the past accounts for the prior probability, and with the knowledge of evidence data, we calculate the posterior probability of a link to be present between two nodes given they have some feature value as 'f'.

$$P\left(\frac{L}{F}\right) = P\left(\frac{F}{L}\right) \cdot \frac{P(L)}{P(F)}$$

$P(F=f)/L$  – Probability of feature value equal to 'f' among the edges present in evidence

$P(L)$  – Prior probability of the edge, which is obtained from the prior probability matrix(mat) obtained in step 3

For example,  $P(1,2)=mat(1,2)+mat(2,1)/2$ ;

$P(F=f)$  – Probability of feature value equal to 'f' among feature values of all possible edges

For example, let's try to calculate the probability of edge (2,4) to be present in the future.

$$P(\text{Link (2,4)/feature=1}) =$$

$$P(\text{feature=1/Link}) * \text{Prior probability of (2,4)} / P(\text{feature=1});$$

$$P(\text{feature=1/Link}) = 2/4 = 0.5$$

Evidence data edges  
with feature values



Node1	Node2	Value
2	3	1
2	6	1
4	2	3
4	6	4

Prior Probability is obtained from the prior probability matrix obtained in step 3

$$P(2,4) = P(2,4)+P(4,2)/2$$

$$= (1/6+1/4)/2$$

$$= 0.2083$$

$$P(F) = P(F/L)*P(L)+P(F/\sim L)*P(\sim L)$$

$$P(F=1) = 0.5*0.2083+1/6*(1-0.2083) = 0.1418$$

$P(F/\sim L)$  is the probability that the feature value to 'f' among the possible edges that are actually not present in the evidence data.

$$P(\sim L) = 1 - P(L)$$



**Input**

*Feature values for all possible edges- fPossibleEdges*

*Evidence data- evidenceData*

*Prior probability matrix*

*Number of features*

**Output**

*Bayesian probability for all edges Matrix*

**For each** edge present in the fPossibleEdges

**For each** feature

*$P(F=f/L) = \text{number}(F=f) / \text{actual number of links present};$*

*$P(L) = \text{prior probability from the prior probability matrix};$*

*$P(\sim L) = 1 - P(L);$*

*$P(F=f) = P(F=f/L) * P(L) + P(F=f/\sim L)P(\sim L);$*

*$P(L/F=f) = P(F=f/L) * P(L) / P(F=f);$  (Bayesian inference)*

*Store the value of the posterior probability calculated above;*

**End**

**End**

**Step 8: Calculate the number of possible predictions that can be made**

Using the evidence data and test data, get all the edges present in test data whose both nodes appear in the evidence data. Main aim of this step is to find out what all links in test data can be predicted with the knowledge of the evidence data, and how many of such edges are presented.



**Input**

*eData= Evidence data*

*tData= Test data*

**Output**

*numPossiblePredictions= Number of possible predictions*

*possiblePredictions= Possible predictions that can be made*

*Get unique nodes from eData;*

*Initialize numPossiblePredictions to 0;*

**For each** *edge present in tData*

**If** *both the nodes of the edge are present in the unique evidence data*

*Add the edge to the possiblePredictions;*

*Increment the numPossiblePredictions by 1;*

**End**

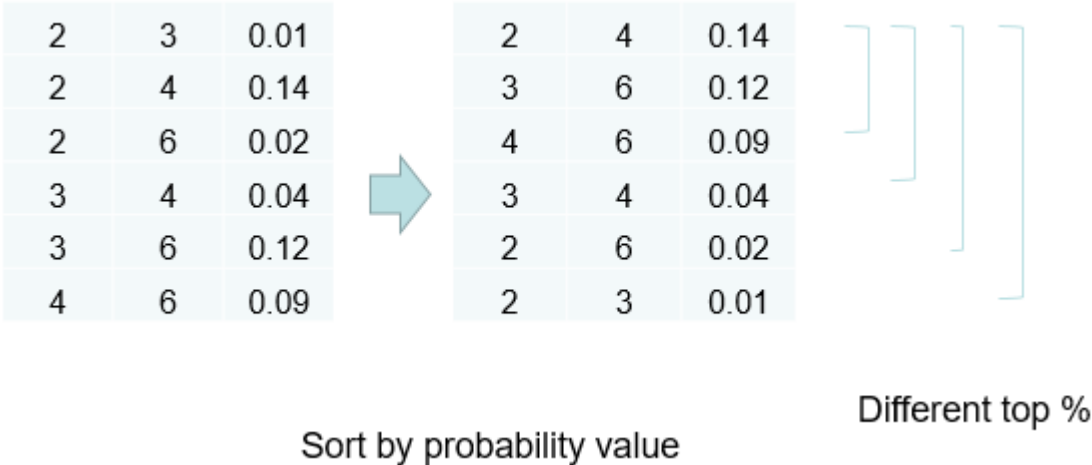
**End**

**Step 9: Predict edges**

After the posterior probabilities for all possible edges are calculated, predict edges with certain top % of posterior probabilities as the edges to appear in the future and calculate the true

positives, false positives, false negatives, and true negatives, which can be used to identify how good algorithm is able to classify the edges.

Top 5%, 10%, 15%, 20%, and 25% of the edges are generally considered to be predicted edges which can later be used to evaluate the algorithm performance.



### **Input**

*All possible edges with Bayesian probability*

*Test data*

*Number of possible predictions*

*Possible predictions*

*topThresholdMatrix= [5 10 15 20 25]*

### **Output**

*Link predictor matrix containing:*

*True positives, True negatives, False positives, and False negatives for each feature*

**For each** *element of topThresholdMatrix*

*Get the edges with the top topThresholdMatrix(i)% of Bayesian probability values ;*

*Consider these edges as the predicted edges;*

*True positives= (predicted edges) intersect (possible predictions)*

*True negatives= edges that're not predicted and are also not present in possible predictions;*

*False positives= edges that are predicted but are not present in possible predictions*

*False negatives= edges that are present in test data but are not predicted*

*Store all these values;*

**End**

## Step 10&11: Calculate relevancy measures

Detailed explanation about relevancy measures is given in the results chapter.

### **Input**

*Link predictor matrix from step 9*

### **Output**

*Relevancy measures such as:*

*Precision, Recall, F-1 measure, and specificity*

### **For each feature**

*precisionValue= truePositives/(truePositives+falsePositives);*

*recallValue= truePositives/(truePositives+falseNegatives);*

*specificityValue= trueNegatives/(trueNegatives+falsePositives);*

*f1Measure= 2\*precisionValue\*recallValue/(precisionValue+recallValue);*

### **End**

### 3.3 Our Algorithm Version-2: different method of calculating Prior Probability

- We experimented with the way the prior probability is calculated from the past data to see if it produces any interesting behavior.
- For each month in the past data period, we keep track whether one particular edge is present or not with the Boolean values 0 and 1.
- Then the prior probability will be calculated as:

*Prior Probability*

*= (sum of corresponding Boolean values for all the months in past data + 1)/(number of months in the past data + 1)*

The reason for adding 1 to both the numerator and denominator is to make sure that each and every possible edge has a non-zero prior probability value, which in turn implies that it will have a non-zero posterior probability value.



$$\text{Prior probability} = (1+1+1)+1/7 = 4/7$$

## Chapter 4

### Experimental Setup and Results

#### 4.1 Datasets

##### **Enron email database:**

For the purpose of evaluating our technique we selected two datasets, one of them is Enron email dataset. This dataset was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation. A lot of preprocessing was done on the dataset later by outside sources to remove the integrity problems and also some emails were deleted upon request of the employees.

The MySQL dump of this database was imported and table structure of all the tables was analyzed to extract data to test our algorithm. The tables and the fields present in each of those tables are:

1. Employeeelist

- a) Eid
- b) Firstname
- c) Lastname
- d) Email\_id

2. Message

- a) Mid
- b) Sender
- c) Date
- d) Message\_id
- e) Subject



- f) Body
- g) Folder

3. Recipientinfo

- a) Rid
- b) Mid
- c) Rtype
- d) Rvalue
- e) Dater

4. Referenceinfo

- a) Rfid
- b) Mid
- c) Reference

### **Facebook WallPosts dataset:**

Our algorithm was also tested with another dataset- “Facebook WallPosts dataset”, which contains information about wallposts written on Facebook by its users in a city during certain timeframe. While doing one of their research works[5], Bimal Viswanath and Alan Misolve collected this dataset and did the pre-processing, which was then made available to me upon request. This dataset meets our requirement of having the edge timestamp along with graph information, and hence was used to evaluate results.

## **4.2 Data extraction**

### **Enron Email dataset:**

The tables in the MySQL database were queried to get the data required to test our algorithm,. The format of the data that can be used as input data will have the record format like - <node1,node2,weight>, meaning there is a directed edge from node1 to node2 with the

respective weight as the third argument. In case of Enron dataset, the node1 and node2 in the aforementioned record format refers to sender and receiver respectively, and the weight refers to the number of messages sent by sender to receiver during a particular period.

The query given below is used to extract the data from the tables. Please note that the 'where clause' was changed in accordance with the timeframe we are testing our algorithm with.

Query to create view:

Create view view1 as

```
Select e.eid as sno,e1.eid as rno,date(m.date) as new_date,count(*) as total
```

```
From message m, recipientinfo r, employeelist e, employeelist e1
```

```
Where m.mid=r.mid and m.sender=e.Email_id and r.rvalue=e1.Email_id
```

```
Group by e.eid,e1.eid,date(m.date);
```

Query to extract the data:

```
SELECT Sno,Rno,SUM(Total) AS Total
```

```
FROM view1 where year(new_date)="2000" and month(new_date) between 9 and 10
```

```
GROUP BY Sno,Rno;
```

The results from these queries will look like:

Sno	Rno	Total
1	73	2
2	19	7

2	70	14
3	68	2
9	48	13
9	50	12
9	57	4
9	67	19

**Facebook Wallposts dataset:**

The data from Facebook Wallposts dataset is in the form of: person1-person2-number of wallposts-time interval, which means person1 has written those many number of wallposts on the Facebook wall of person2. And this in the graph form can be represented as node1-node2-weight. The data looks like:

person-1	person-2	numberOfWallposts
24215	18322	1
14664	11321	13
21458	5584	7
13013	4376	4
19158	15682	3
5386	5377	10
7198	491	1
109	521	6

For our testing purposes, we selected some fixed number of unique random nodes from the entire dataset, i.e., 400, 600, and 800 specifically.

**4.3 Parameters**

**Number of iterations:**

One iteration of running our algorithm represents executing it once to predict the edges to be formed in the test period taking both the past data and evidence data as inputs.

Typically, the test period is two months, and the two months before the test period is taken as evidence data, and the six months before the evidence data is taken as the past data. The algorithm will be executed against several of such consecutive periods in the dataset where there is enough data to make meaningful predictions. For example, in the Enron dataset, we have executed our algorithm for 20 iterations to test the algorithm.

### **Number of Bins:**

After calculating feature values for all the possible edges that can be predicted based on the nodes present in the evidence data, these feature values are assigned into bins to make meaningful predictions. The way the feature values are divided into bins is explained clearly in the previous 'Bayesian Link Prediction Algorithm' section. We tried experimenting by changing the number of bins to 4, 5, and 6 and observed that the results look very much similar with all of these values. For our experiments, we kept the number of bins as '6'.

### **Criteria for eliminating outliers when dividing into bins:**

We have eliminated the top 1% of values as outliers when dividing the feature values into respective bins. The method by which this is done is explained clearly in the previous section about algorithm.

### **Criteria for predicting edges from the final results:**

We have taken the top 5%, 10%, 15%, 20%, and 25% of the edges sorted in the descending order of their feature values as the possible predictions, and then to make different evaluations of the results of the algorithm.

#### 4.4 Relevancy Measures

Calculate True Positives, True Negatives, False Positives, and False Negatives from the results.

- True Positives(TP): number of edges that are predicted to be present in the test data and are actually present
- True Negatives(TN): number of edges that are predicted to be not present in the test data and are actually not present
- False Positives(FP): number of edges that are predicted to be present in the test data but are not actually present in the test data
- False Negatives(FN): number of edges that are predicted to be not present in the test data but are actually present

Calculate relevancy measures using the above values:

- **Precision:** The fraction of extracted items that are relevant [12]

$$Precision = \frac{TP}{TP + FP}$$

- **Recall:** The fraction of relevant items that are extracted [12]

$$Recall = \frac{TP}{TP + FN}$$

- **Specificity:** It measures how well the negative instances are classified [12]

$$Specificity = \frac{TN}{TN + FP}$$

- **F-1 Measure:** It considers both precision and recall [12]

$$F - 1 Measure = 2.Precision.\frac{Recall}{Precision + Recall}$$

## 4.5 Results

Our algorithm was run for 21 iterations in the Enron dataset and the results were analyzed by benchmarking against the Nowell's algorithm, which uses the proximity measures directly for predicting the future edges. Detailed explanation of the Nowell's algorithm is given the 'Related Work' chapter of this thesis. Nowell's algorithm showed very good improvement in link prediction over the random predictor.

Main aim of our algorithm is to improve upon the previous algorithm, i.e., to make more number of correct predictions. To benchmark the results for both the algorithms, the four relevancy measures precision, recall, specificity, and f-1 measure are computed for each of the four proximity measures Common Neighbors, Jaccards, Preferential Attachment, and Adamic Adar, the graphs were plotted.

### 4.5.1 Enron email dataset - Comparing our algorithm results with results of Nowell's algorithm

The graphs given below clearly show that our algorithm fares better in precision, recall, and f-1 measure than the Nowell's algorithm. The reason for the previous algorithm being better than our algorithm in specificity is that our algorithm

predicts more number of false positives since we predict some percentage of edges as the future edges whereas the previous algorithm predicts a certain number of edges as future predictions.

### **Terminology used in Results:**

- Top 5%: Algorithm in which possible edges with top 5% of posterior probabilities predicted as edges
- Similarly, Our algorithm is executed for Top 10%,15%,20%, and 25%
- Core value 2: Nowell's algorithm is executed by keeping core threshold as 2
- Similarly, Nowell's algorithm is executed with 3 and 4 core thresholds
- Core threshold: degree of the node in the graph

For each of the four proximity measures (Common Neighbors, Jaccards, Preferential Attachment, and Adamic Adar), respective graph is plotted for 3 relevancy measures (precision, recall, and specificity). Please find them below.

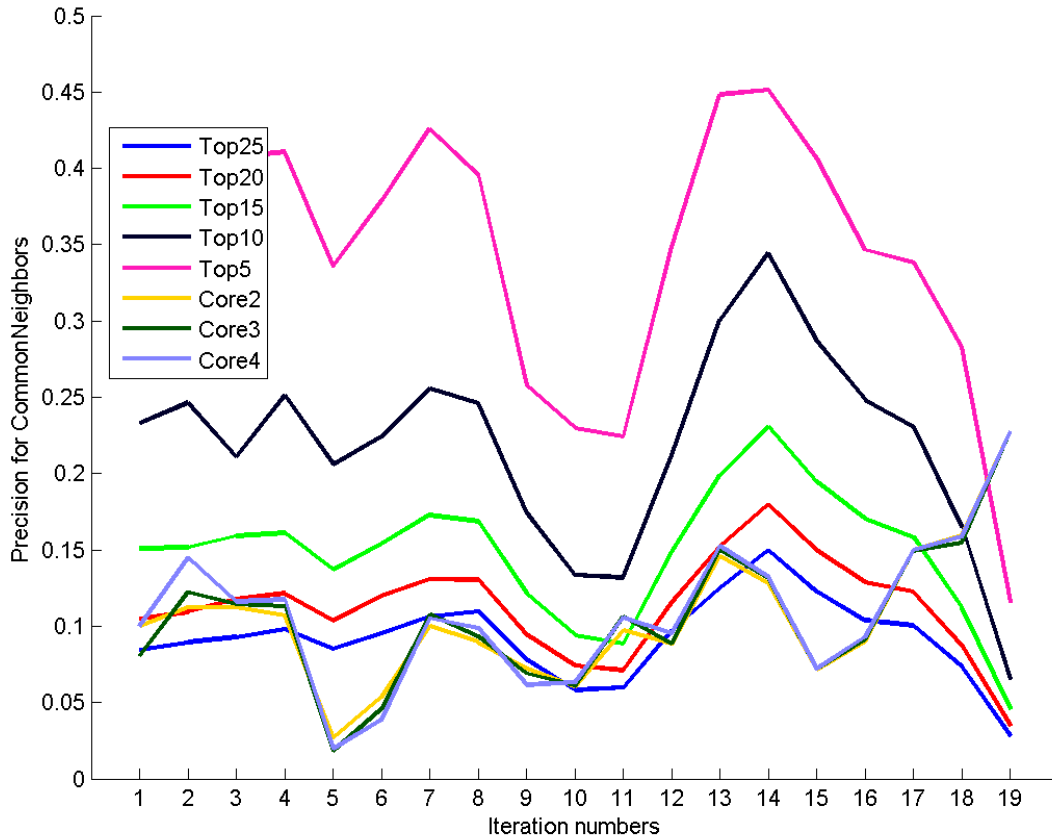
### **Common Neighbors:**

#### **Precision:**

Inferring from the graph,

1. The Precision values are better than those of Nowell's algorithm

- Top 5% > 10% > 15% > 20% > =25% > =Core 2, 3, and 4. The reason for precision of Top 5% version to be greater than other versions is that generally as you predict more number of edges as possible edges to be formed in the future, the precision value decreases.



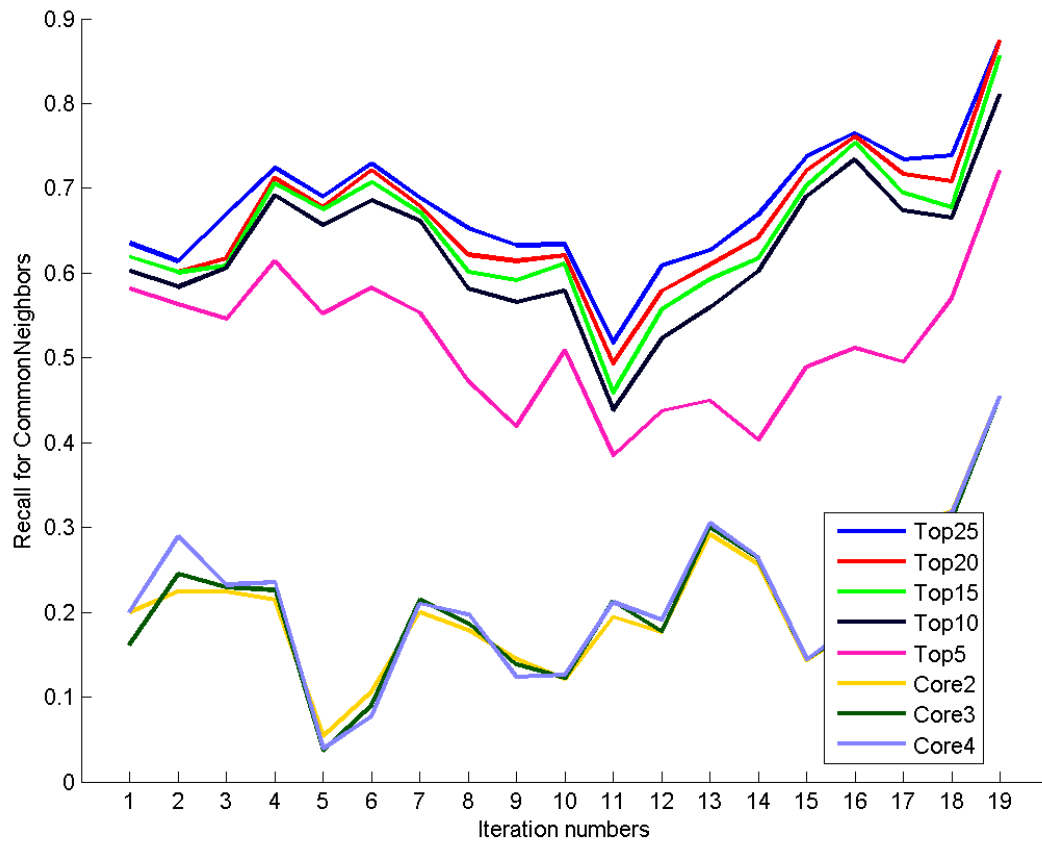
Recall:

Inferring from the graph,

- Recall values are better than those of Nowell's algorithm



- Top 25% > 20% > 15% > 10% > 5% > = Core 2, 3, and 4. The reason for recall value of the Top 25% version to be more than the other versions is that it gets to predict more number of edges than the others.

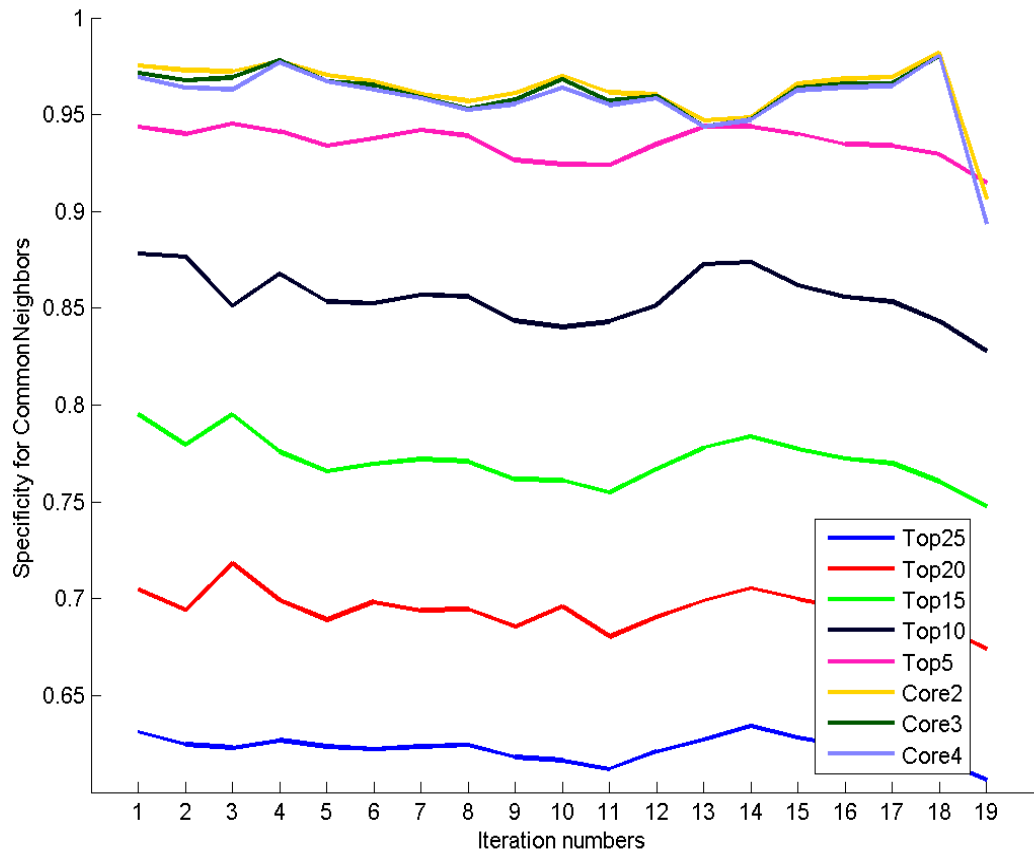


### Specificity:

Inferring from the graph,

- Specificity values of our algorithm are not that good when compared to those of Nowell's algorithm

2. Core 2, 3, and 4 > Top 5% > 10% > 15% > 20% > 25%. The reason for the decrease of the value of specificity as the number of edges predicted increases is that number of false positives increases as we predict more number of edges as possible links.



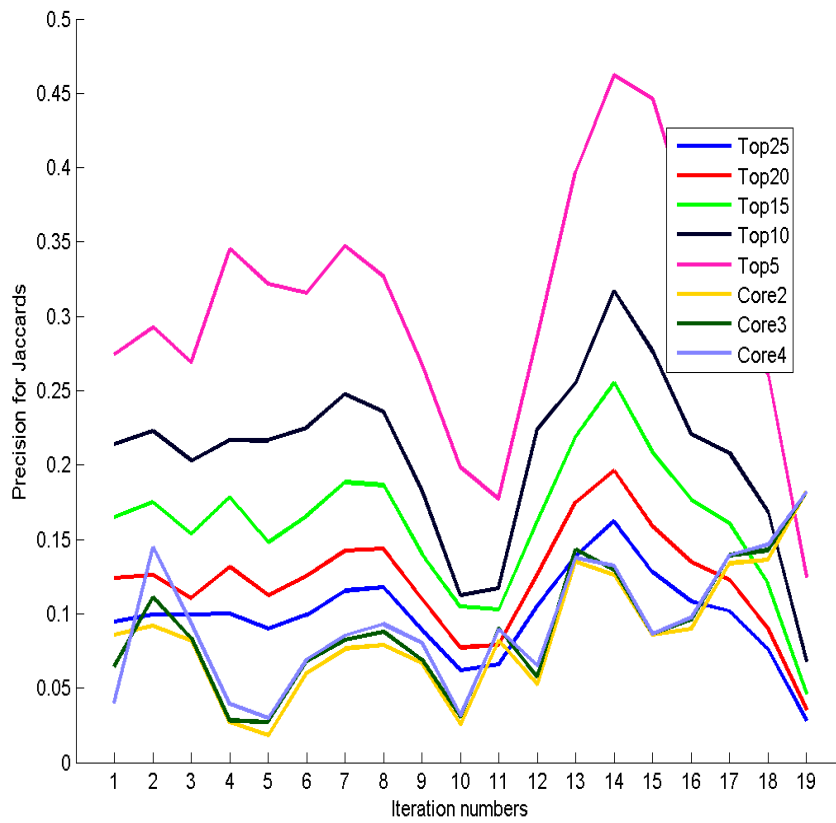
Experimental results have proved that all the remaining three features follow a similar pattern to that of Common Neighbors.

**Jaccards:**

**Precision:**

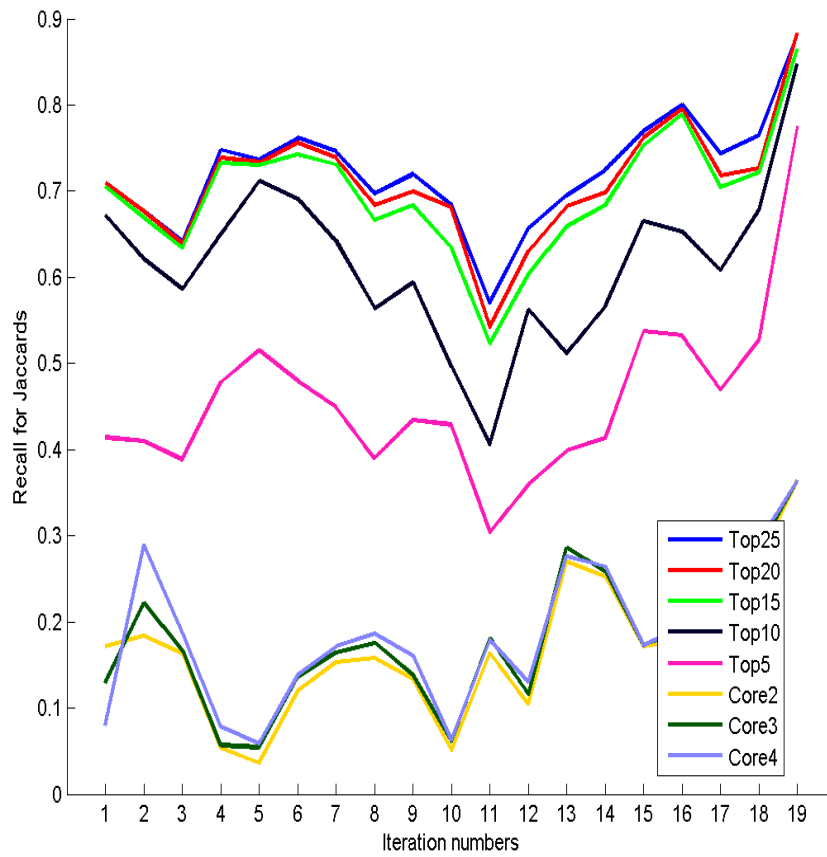
Inferring from the graph,

1. The precision values are better than those of Nowell's algorithm



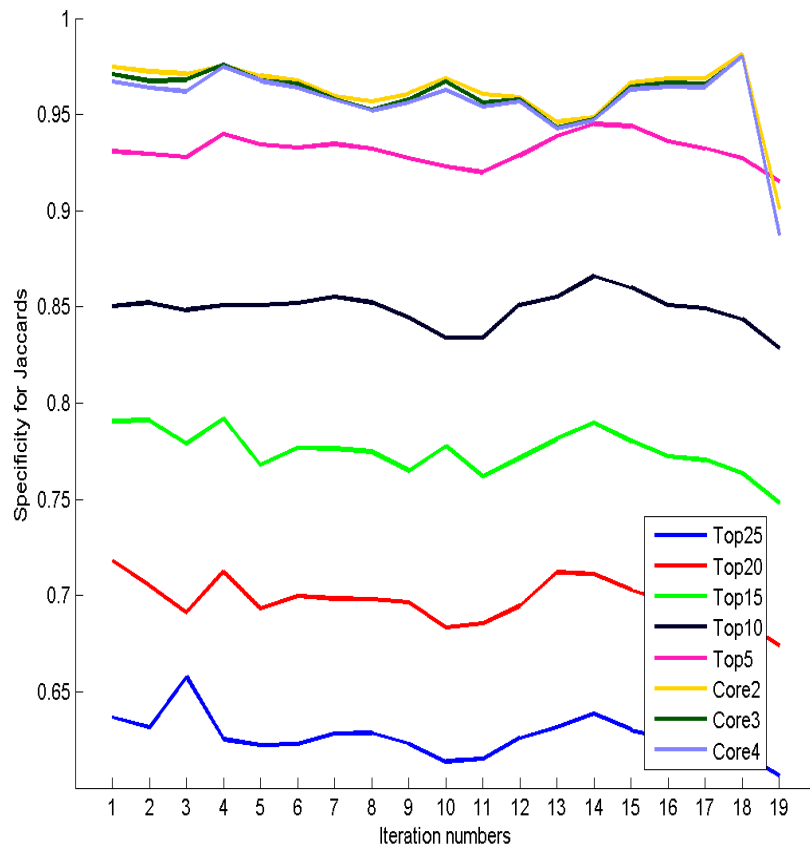
Recall:

Inference: Top 25% > 20% > 15% > 10% > 5% > Core 2, 3, and 4



Specificity:

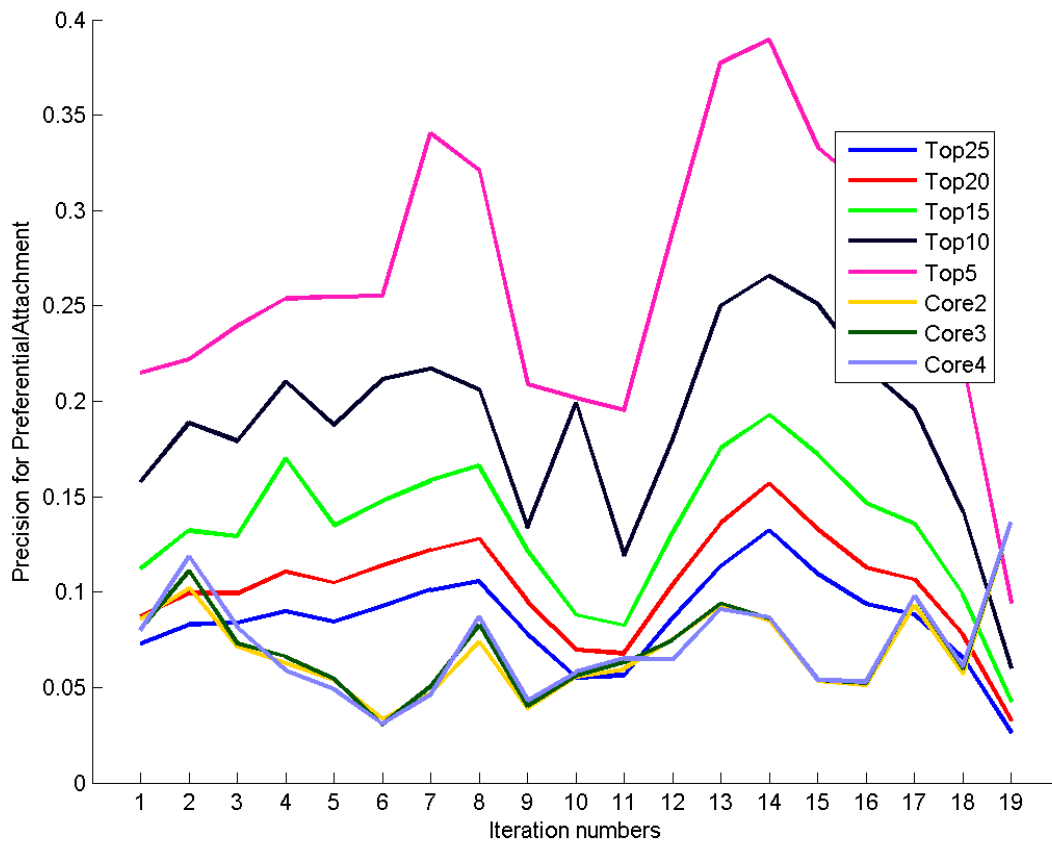
Inference: Core 2, 3, and 4 > Top 5% > 10% > 15% > 20% > 25%



**Preferential Attachment:**

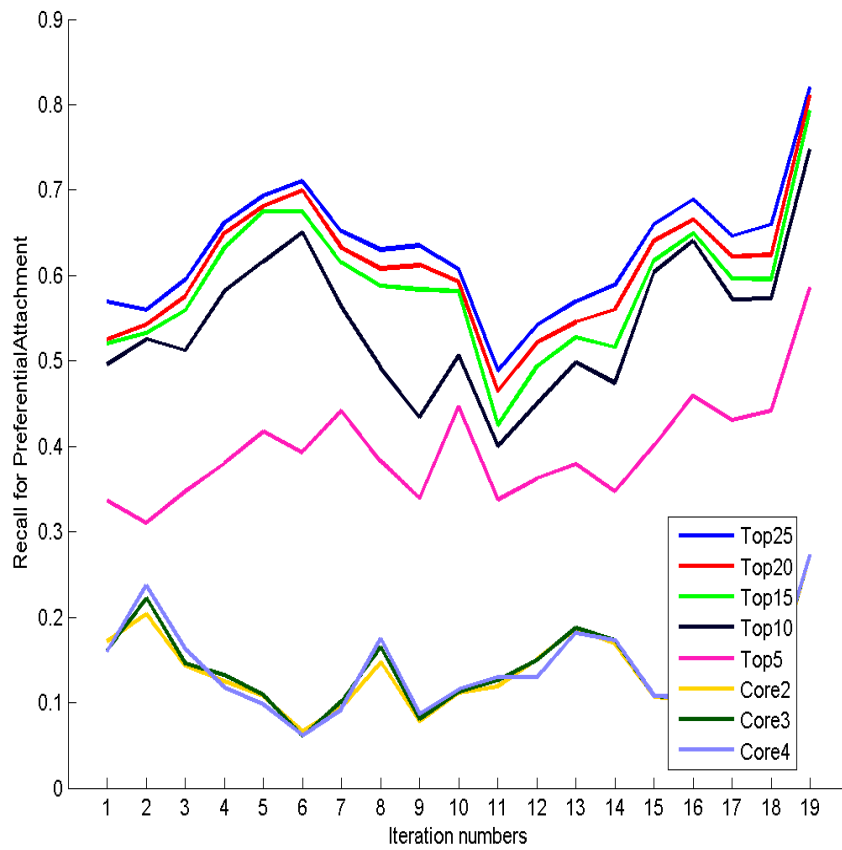
**Precision:**

Inference: Top 5%>10%>15%>20%>=25%>=Core 2, 3, and 4



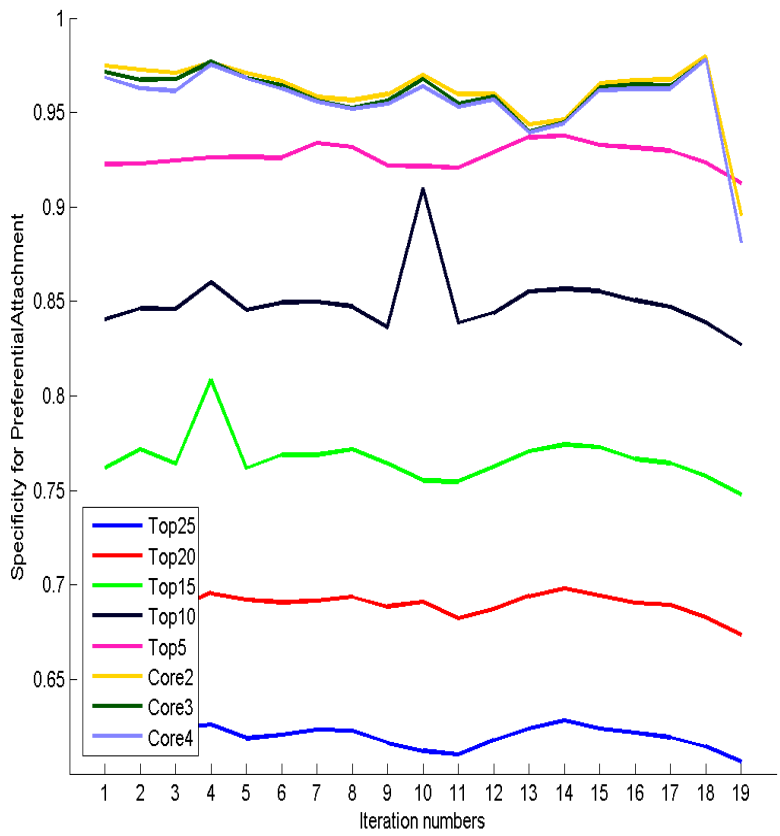
Recall:

Inference: Top 25%>20%>15%>10%>5%>=Core 2, 3, and 4



Specificity:

Inference: Core 2, 3, and 4 > Top 5 > 10% > 15% > 20% > 25%

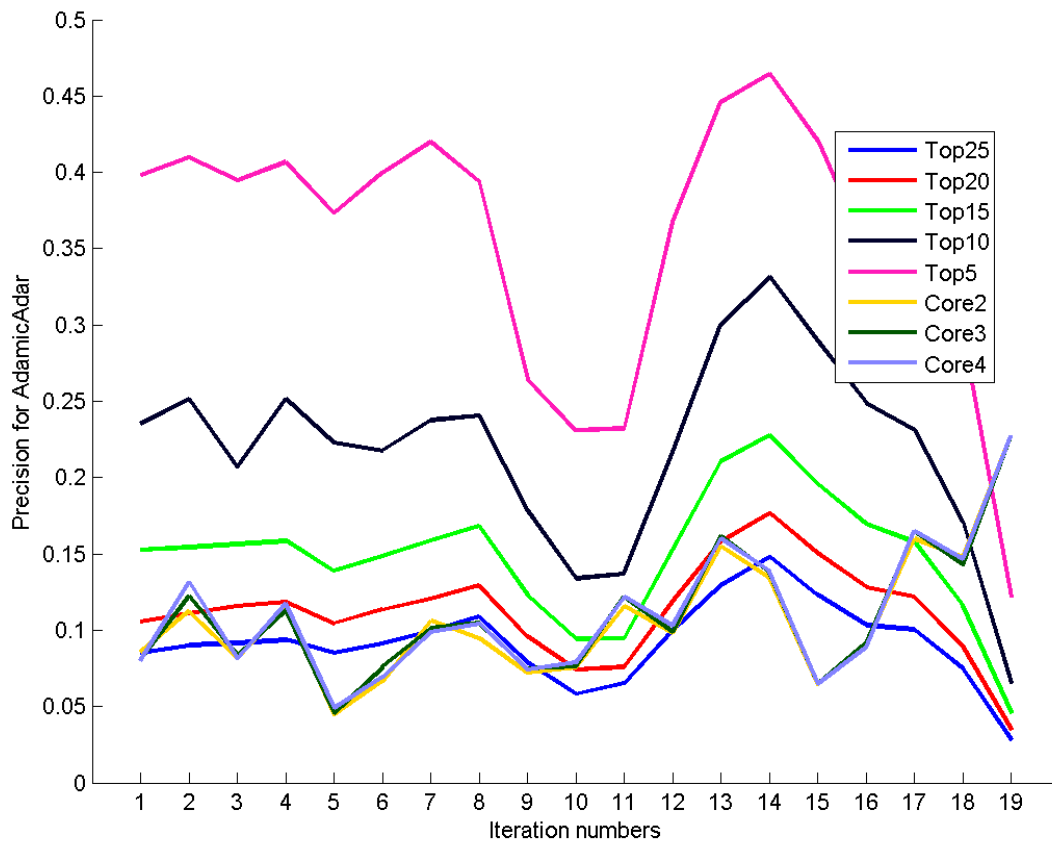


**Adamic Adar:**

**Precision:**

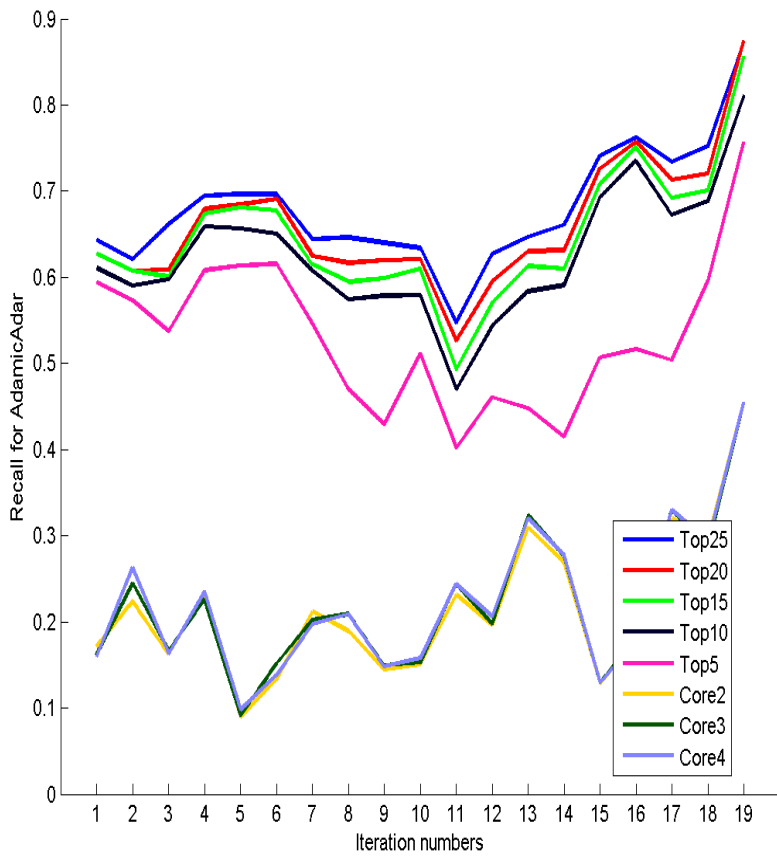
Inference: Top 5% > 10% > 15% > 20% > =25% > =Core 2, 3, and 4





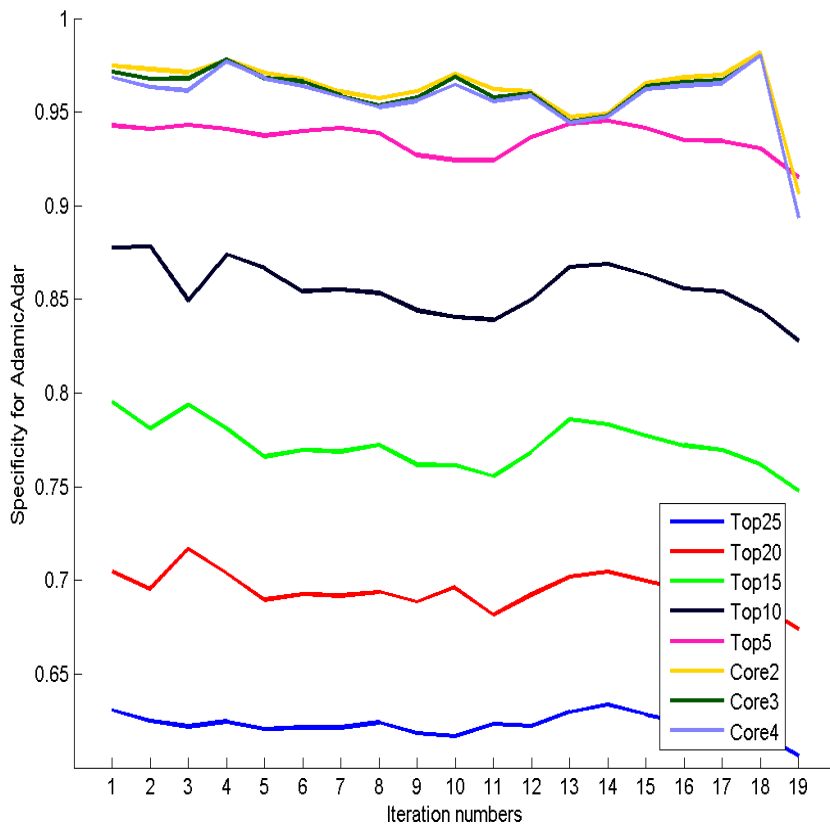
Recall:

Inference: Top 25%>20%>15%>10%>5%>=Core 2, 3, and 4



Specificity:

Inference: Core 2, 3, and 4>Top 5>10%>15%>20%>25%



#### 4.5.2 Facebook WallPosts dataset - Comparing our algorithm results with the results of Nowell’s algorithm

Three random subsets are generated from the Facebook WallPosts dataset by randomly selecting 400, 600, and 800 nodes from each iteration. All other parameters for executing the algorithm were unchanged.

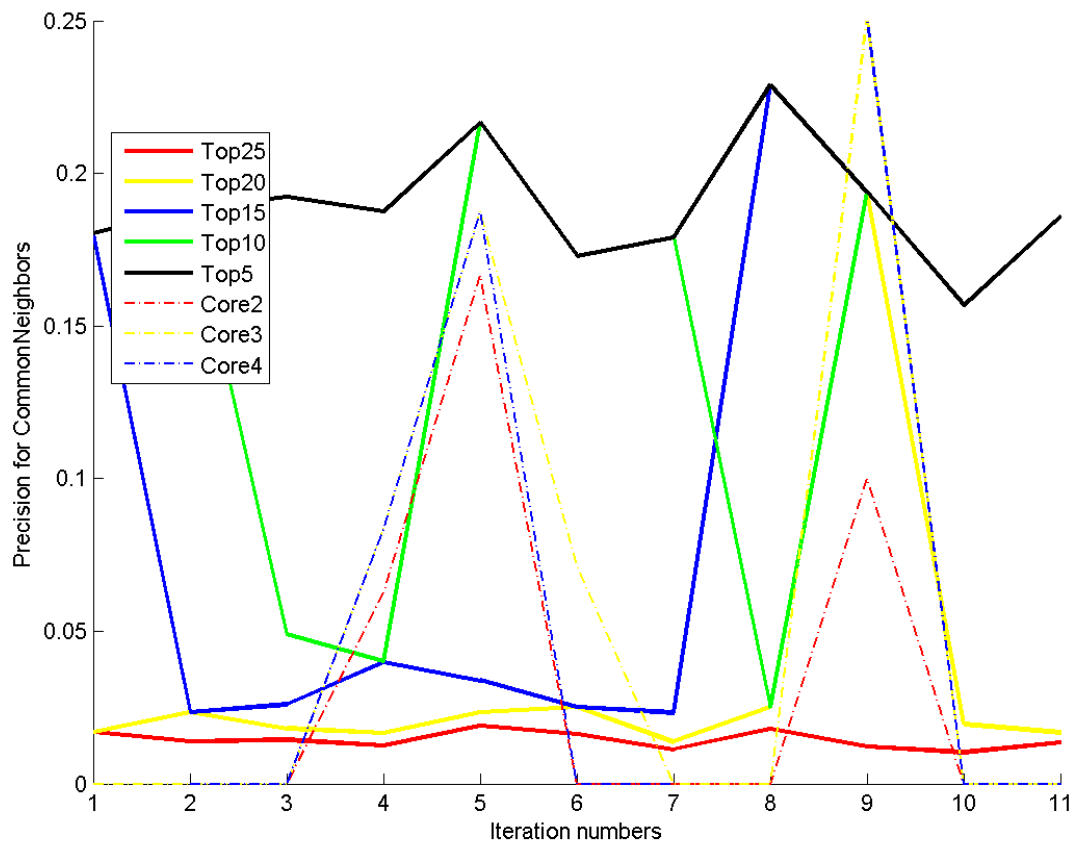
##### Results with Sub-dataset-1 - 400 Nodes:

##### Common Neighbors:

##### Precision:

Inferring from the graph,

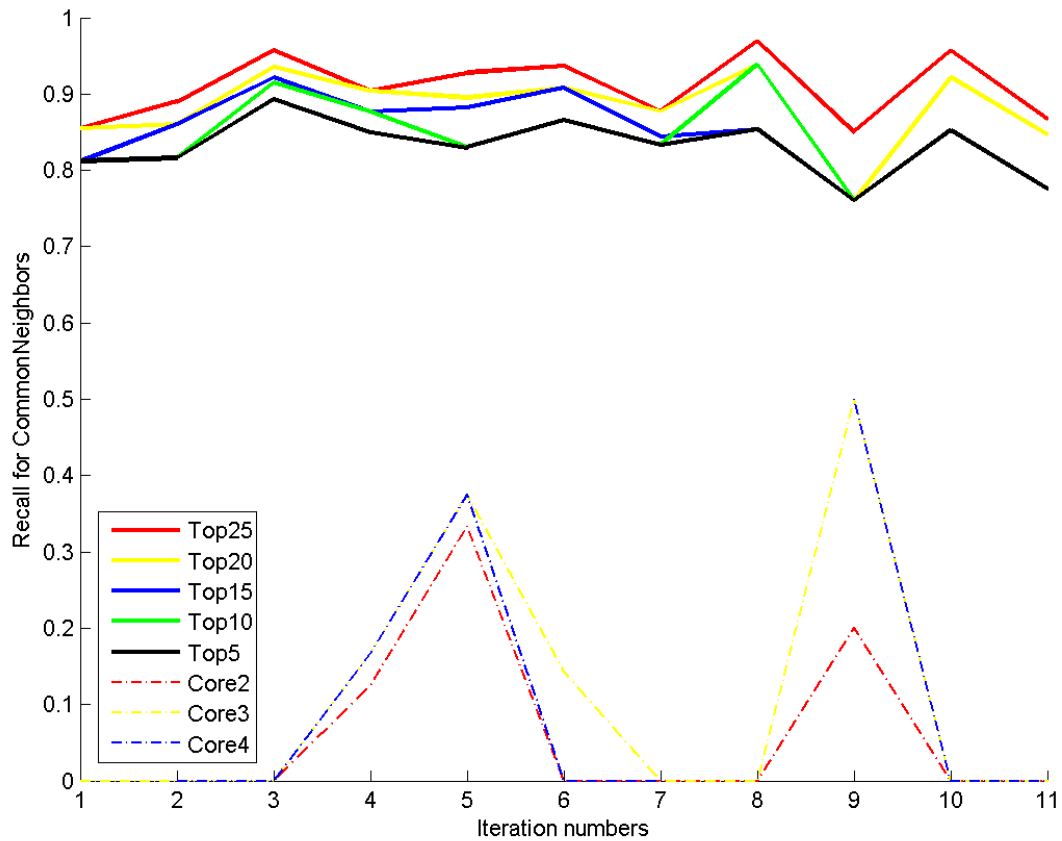
1. Results with top 5% as predicted edges has better precision than Nowell's algorithm for most periods
2. However, as number of edges predicted increases, Nowell's algorithm performs equally good.
3. Top 25% > 20% > 15% > 10% > 5% > = Core 2, 3, and 4



Recall:

Inferring from the graph,

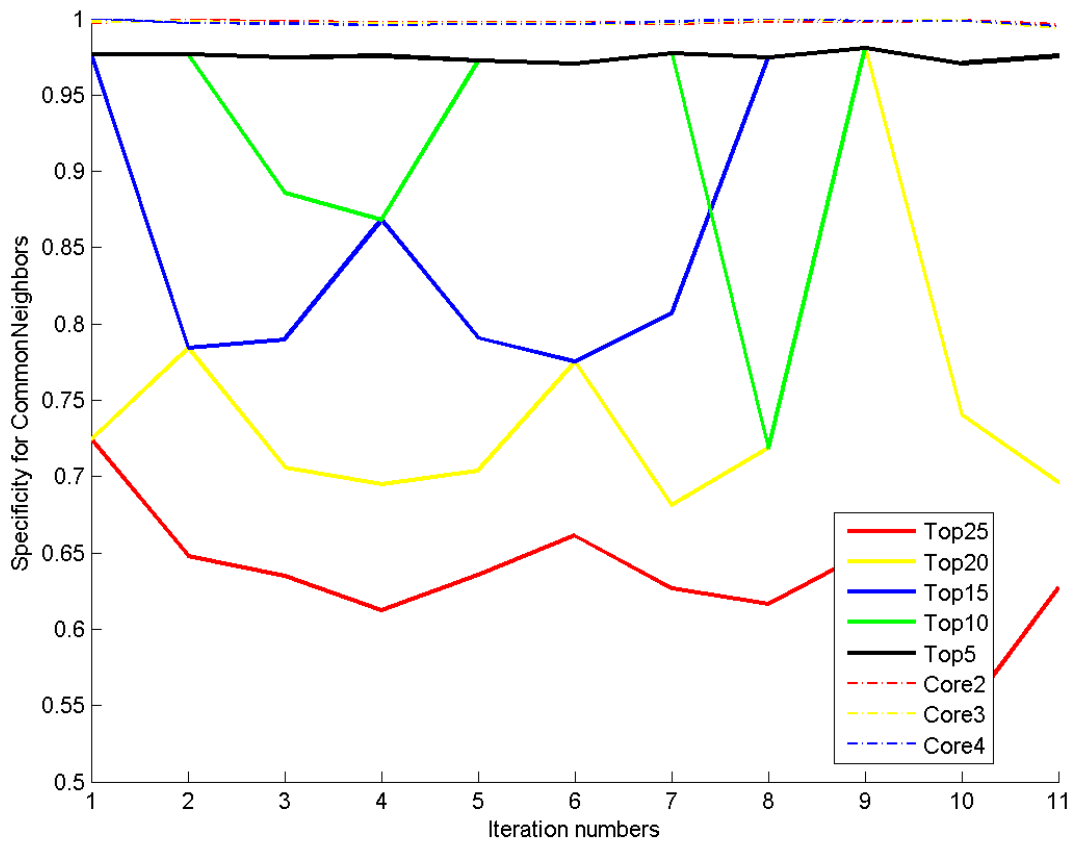
1. Recall values for our algorithm are significantly better than those of Nowell's algorithm
2. Top 25% > 20% > 15% > 10% > 5% = Core 2, 3, and 4



Specificity:

Inferring from the graph,

1. Specificity values of our algorithm are not that good when compared to those of Nowell's algorithm
2. Follows a similar pattern to that of Enron dataset
3. Core 2, 3, and 4 > Top 5% > 10% > 15% > 20% > 25%



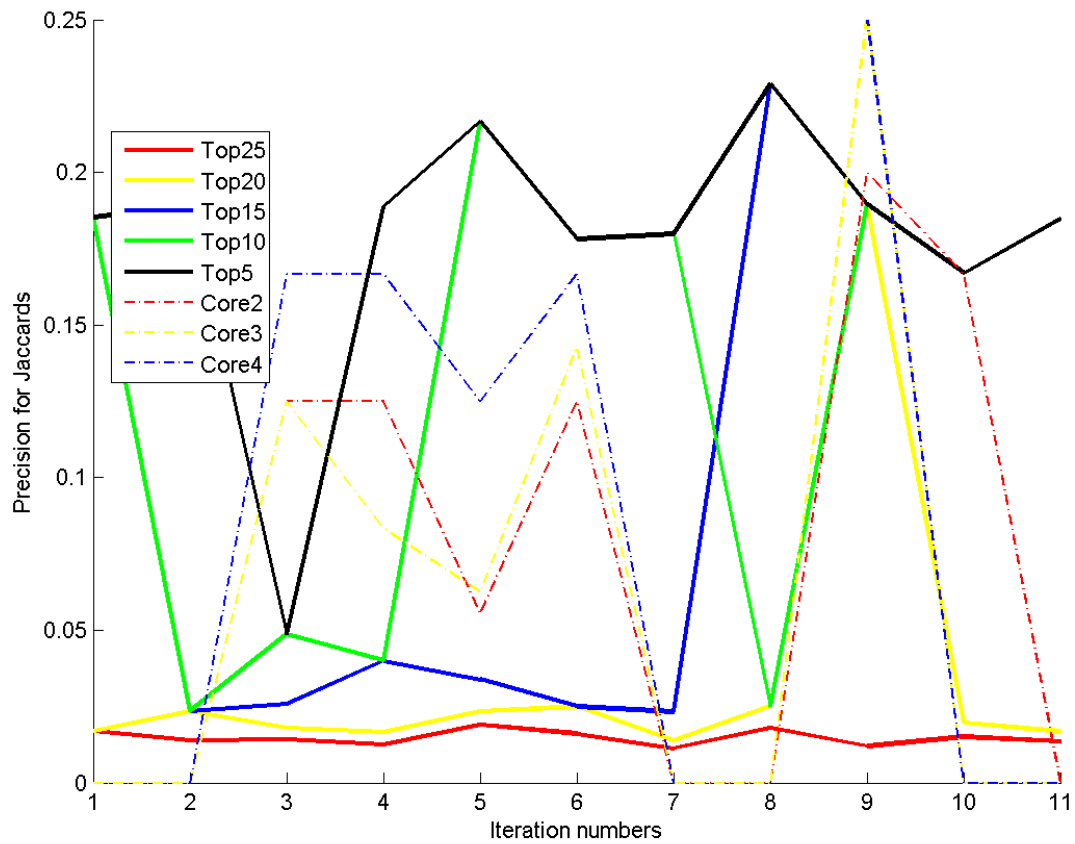
Experimental results have shown that all the remaining 3 features performed similar to that of Common Neighbors in terms of precision, recall, and specificity. The graphs given below reveal the same:

**Jaccards:**

**Precision:**

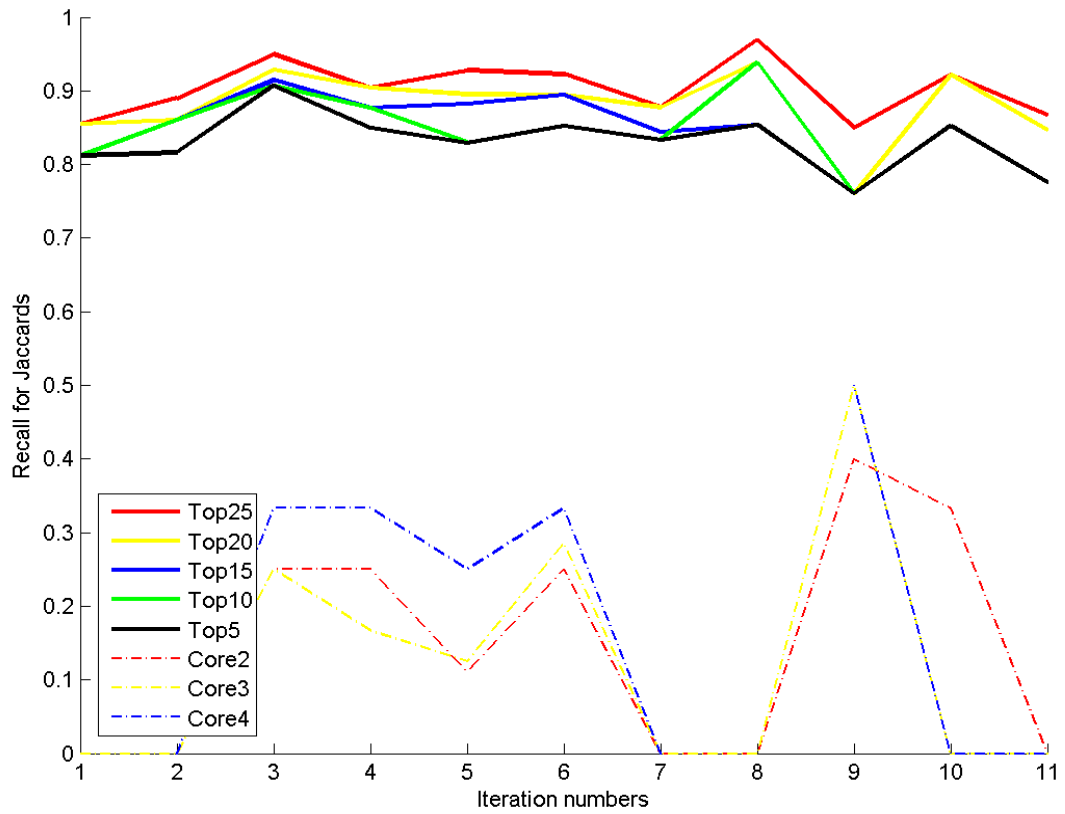
Inferring from the graph,

1. Similar behavior to that of Common Neighbors, as number of edges predicted increases, Nowell's algorithm performs equally good
2. For most periods, precision values of Top 5% are better than those of Nowell's algorithm
3. Top 5% > 10% > 15% > 20% > 25%



Recall:

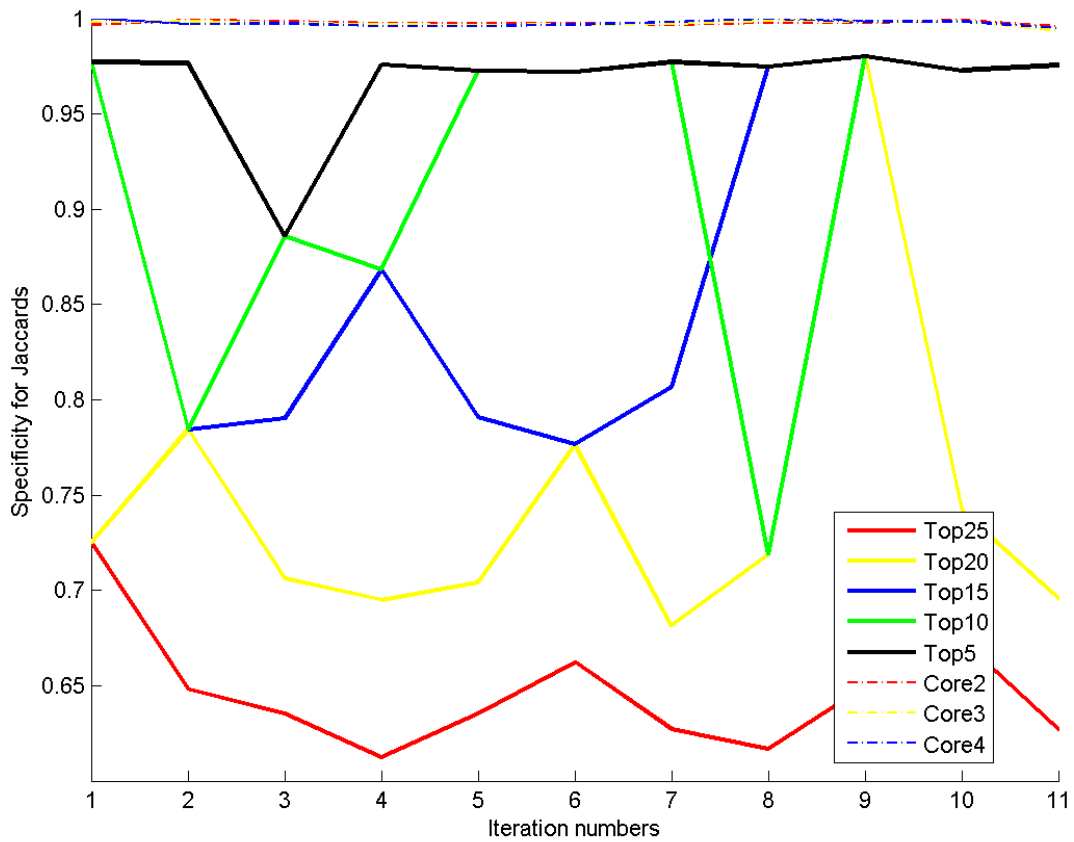
Inference from the graph: Top 25% > 20% > 15% > 10% > 5% > Core 2, 3, and 4



Specificity:

Inference from the graph: Core 2, 3, and 4 > Top 5% > 10% > 15% > 20% > 25%





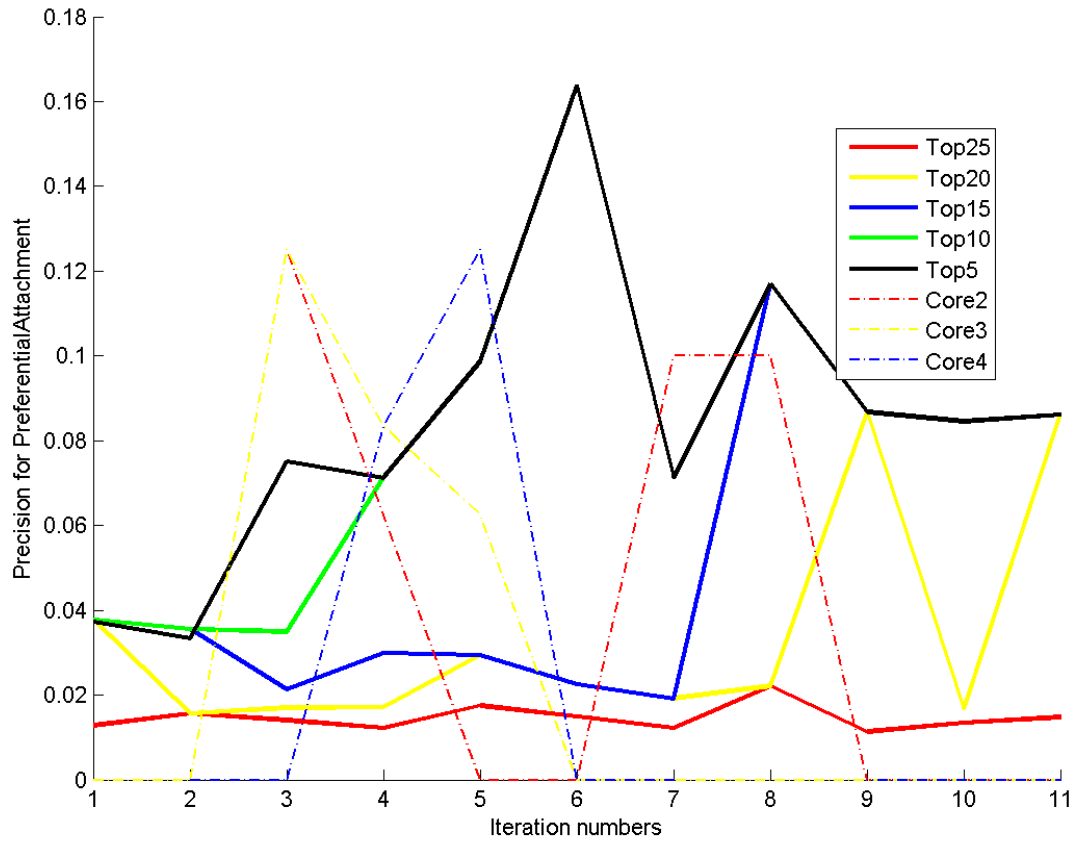
### Preferential Attachment:

#### Precision:

Inferring from the graph,

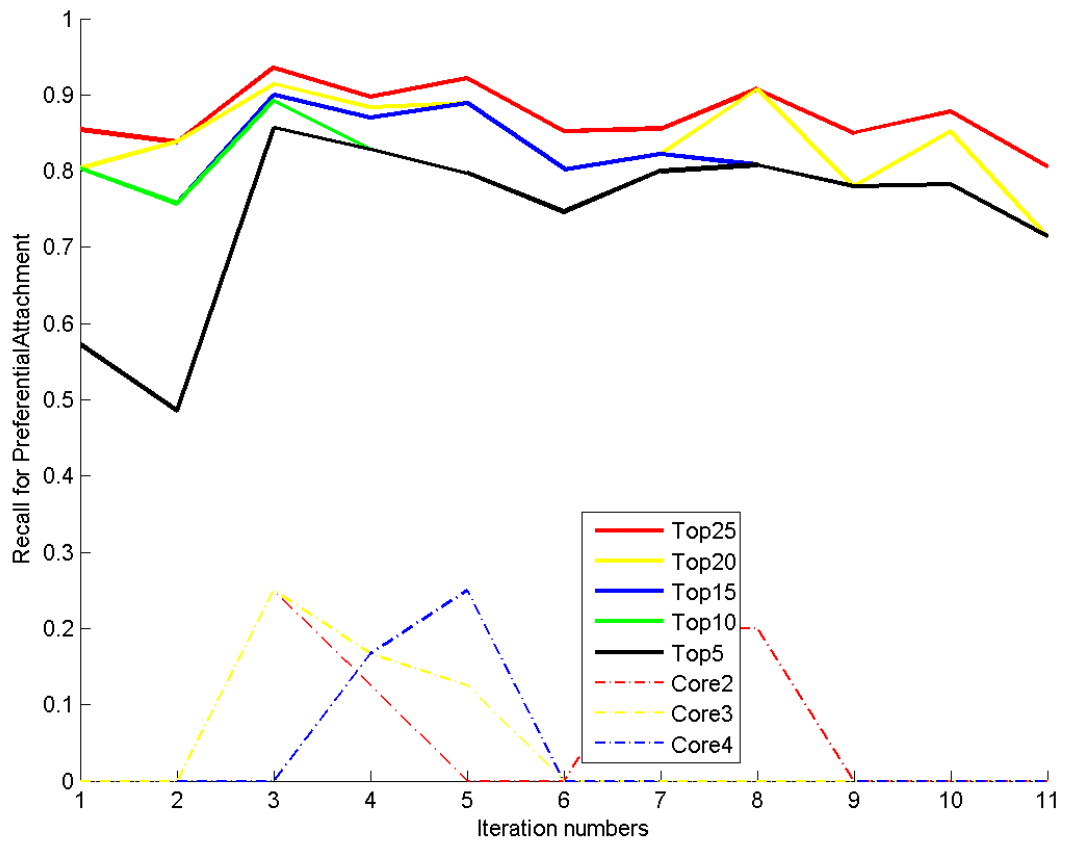
1. Results with top 5% as predicted edges has better precision than Nowell's algorithm for most periods
2. However, as number of edges predicted increases, Nowell's algorithm performs equally good.
3. Similar to Common Neighbors and Jaccards

4. Recall values of Top 5%>10%15%>20%>25%



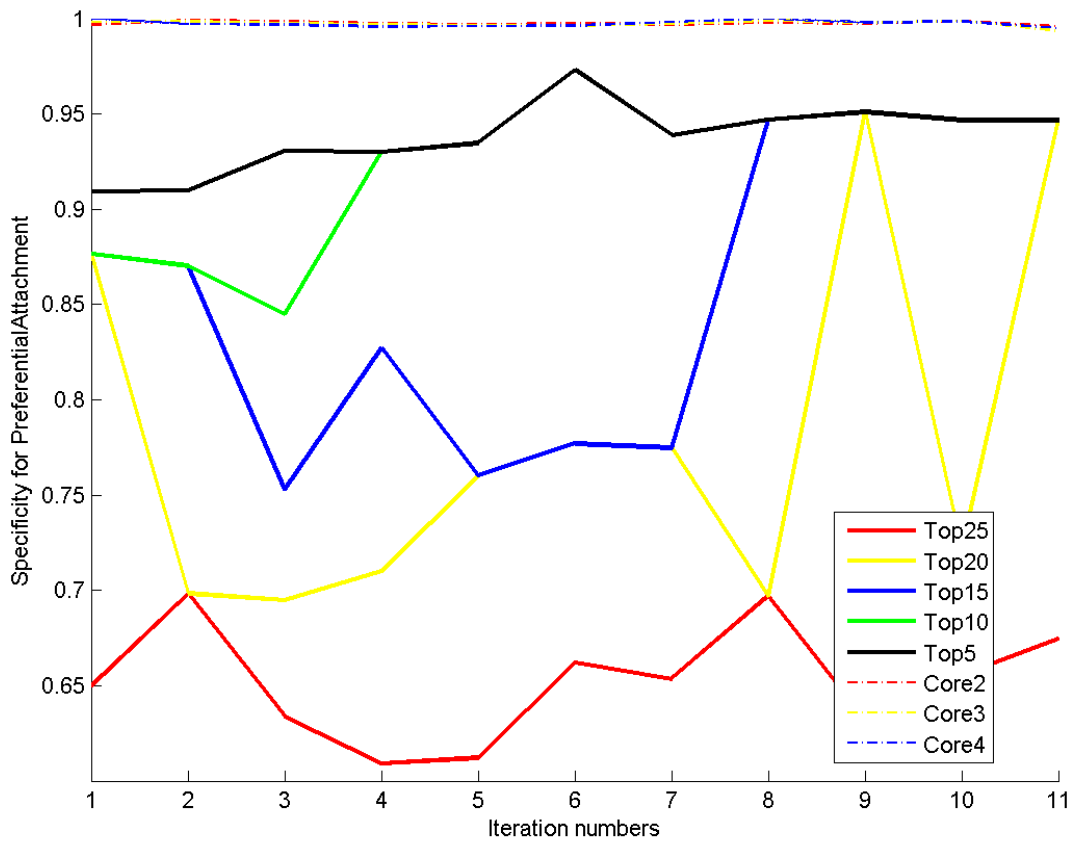
Recall:

Inference from the graph: Top 25%>20%>15%>10%>5%>=Core 2, 3, and 4



Specificity:

Inference from the graph: Core 2, 3, and 4 > Top 5 > 10 > 15 > 20 > 25%



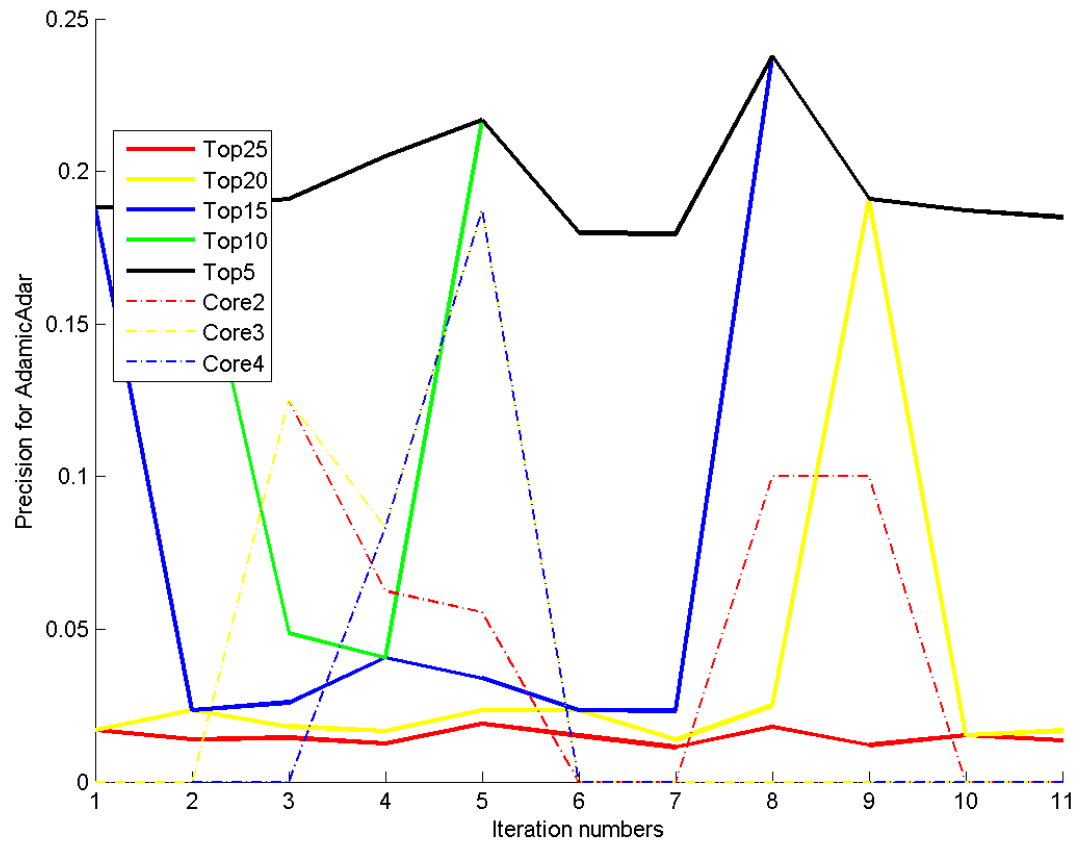
**Adamic Adar:**

**Precision:**

Inferring from the graph,

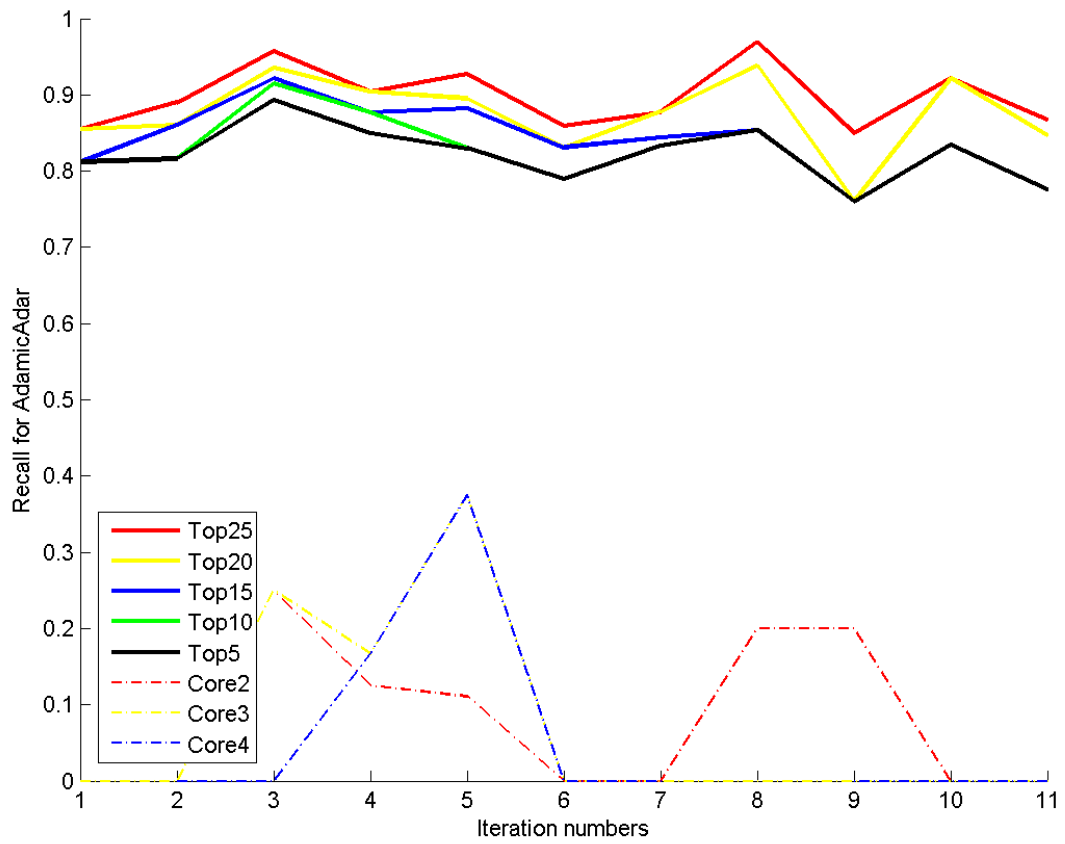
1. Results with top 5% as predicted edges has better precision than Nowell’s algorithm for most periods
2. However, as number of edges predicted increases, Nowell’s algorithm performs equally good.
3. Similar to Common Neighbors, Jaccards, and Preferential Attachment

#### 4. Recall values for Top 5%>10%15%>20%>25%



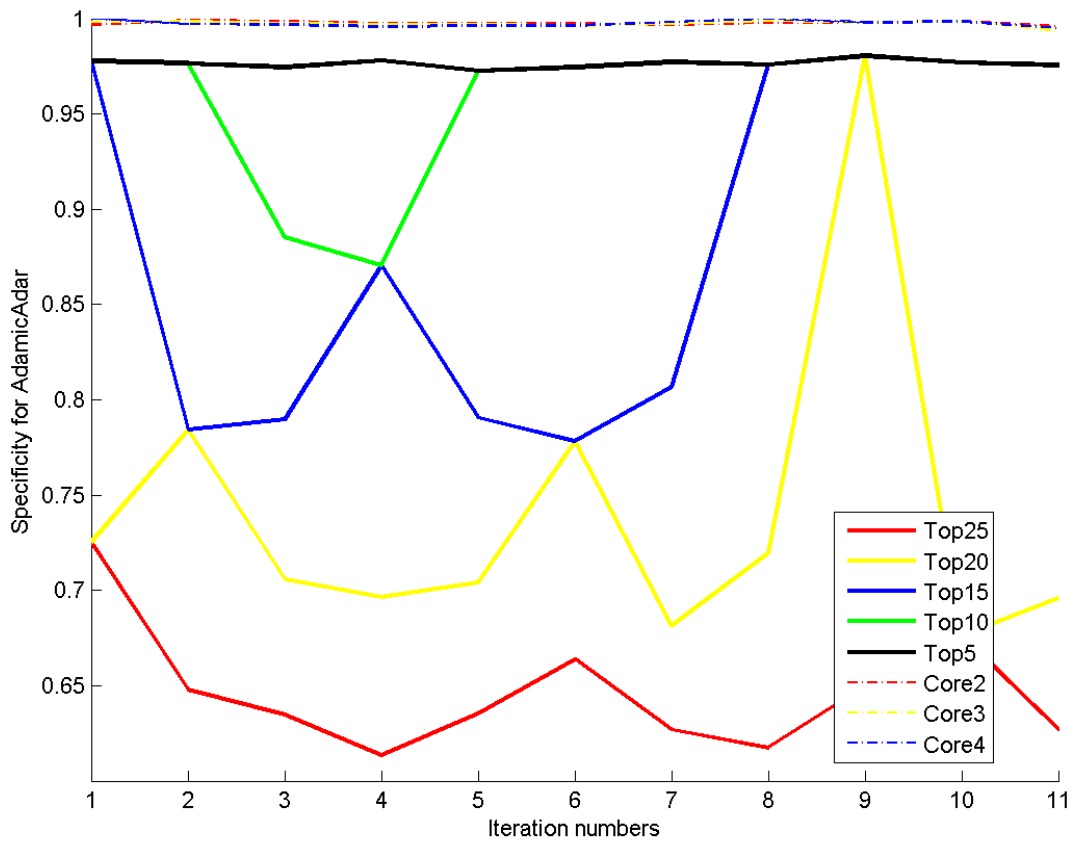
#### Recall:

Inference from the graph: Recall values for Top 25%>20%>15%>10%>5%>=Core 2, 3, and 4



Specificity:

Inference from the graph: Core 2, 3, and 4 > Top 5 > 10% > 15% > 20% > 25%

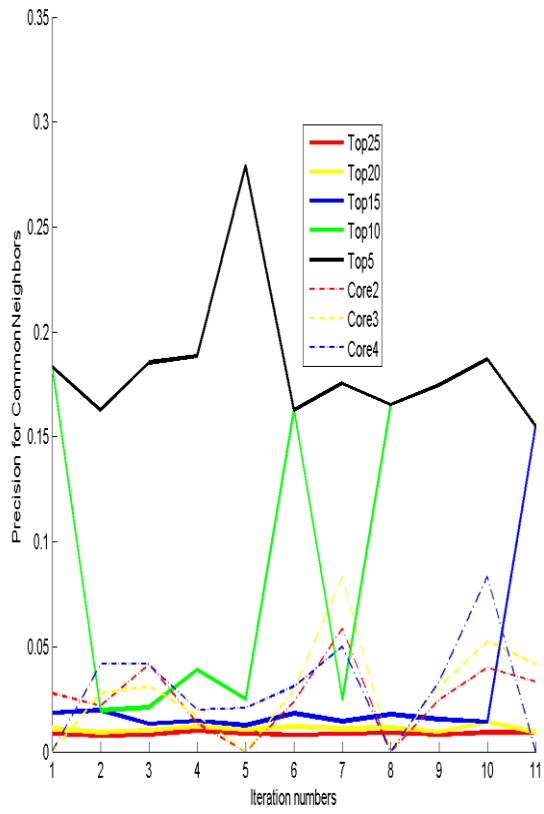


### Results with Sub-dataset-2 - 600 Nodes:

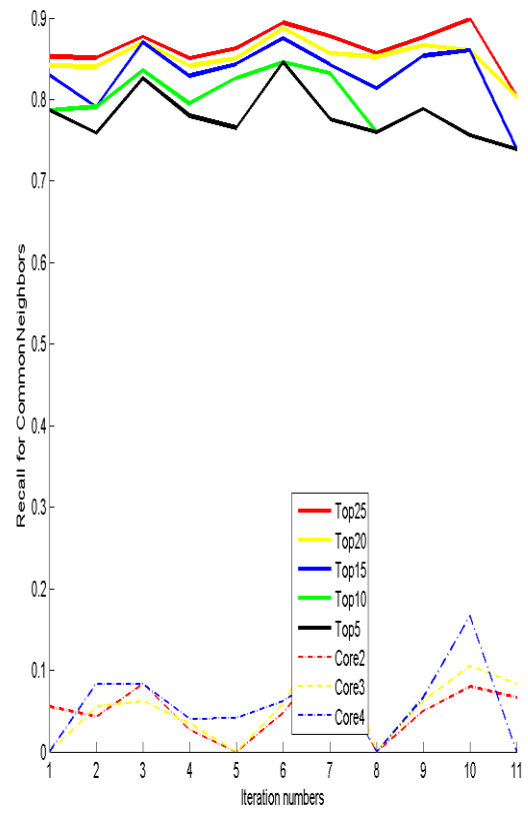
The experiment was repeated for the second subset randomly selected from the Facebook WallPosts dataset by selecting random 600 nodes from each iteration. The results looked similar to results of 400 nodes.

### Common Neighbors:

Top 5%>10%15%>20%>25%

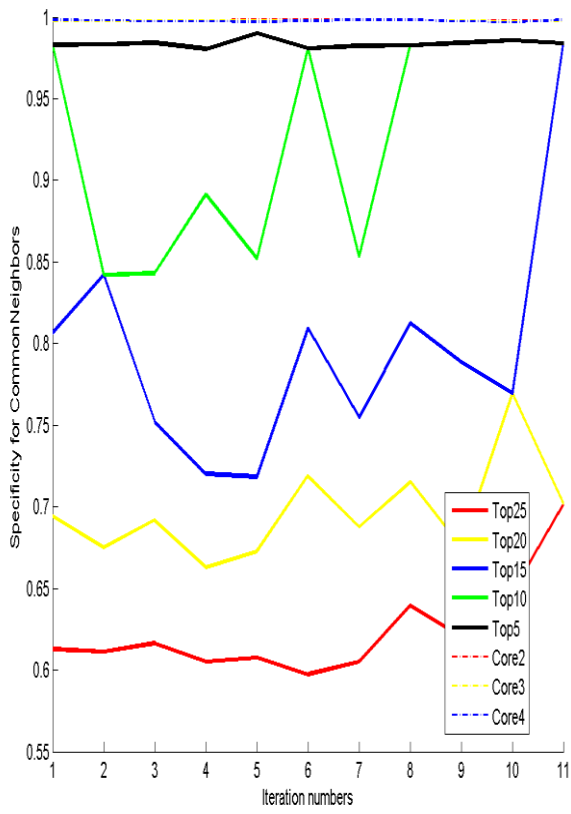


Top 25%>20%>15%>10%>5%>=Core 2, 3, and 4



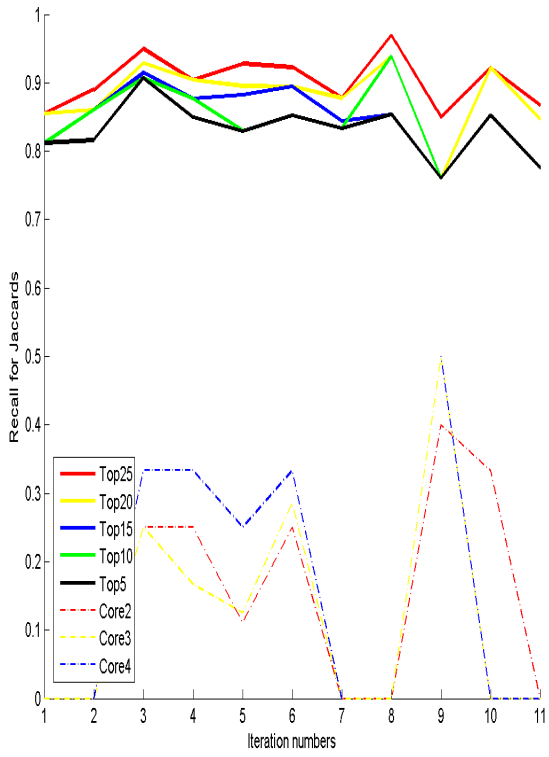


### Core 2, 3, and 4>Top 5%>10%>15%>20%>25%

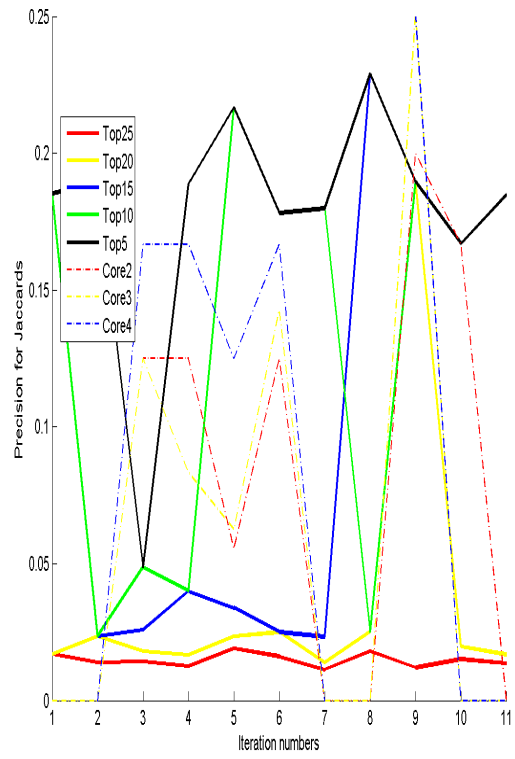


### Jaccards:

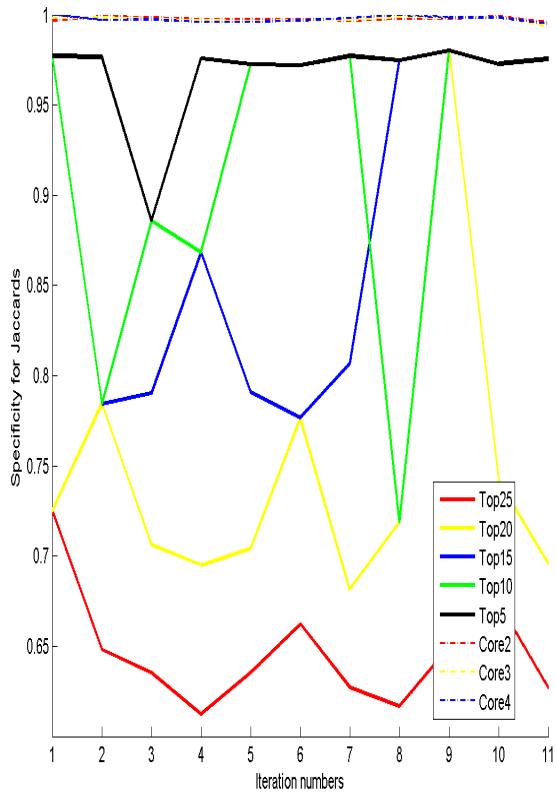
Top 5% > 10% > 15% > 20% > 25%



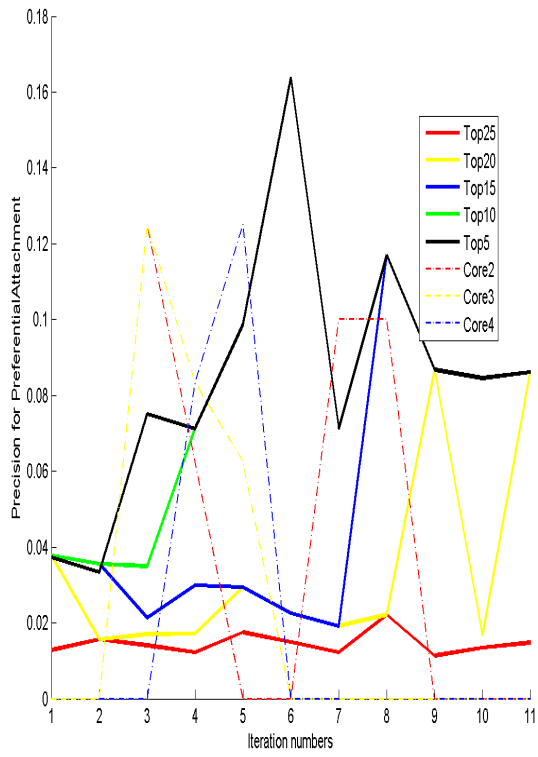
Top 25% > 20% > 15% > 10% > 5% > =Core 2, 3, and 4



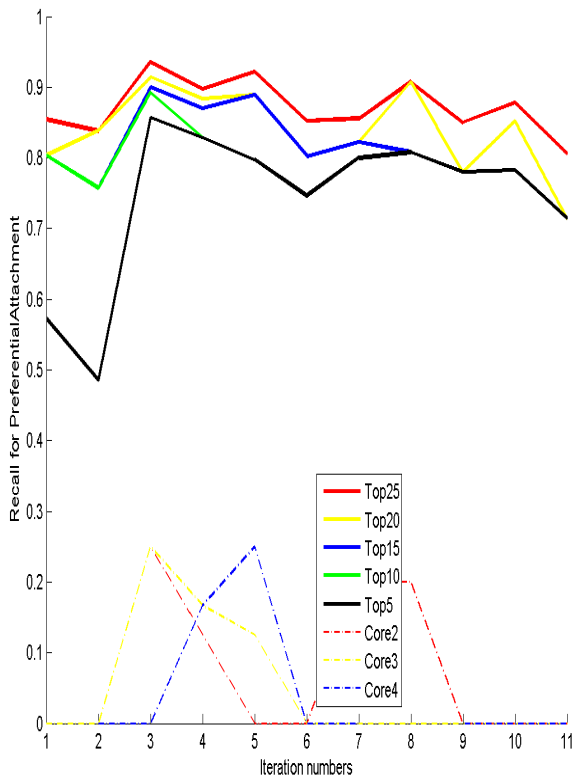
Core 2, 3, and 4>Top 5%>10%>15%>20%>25%



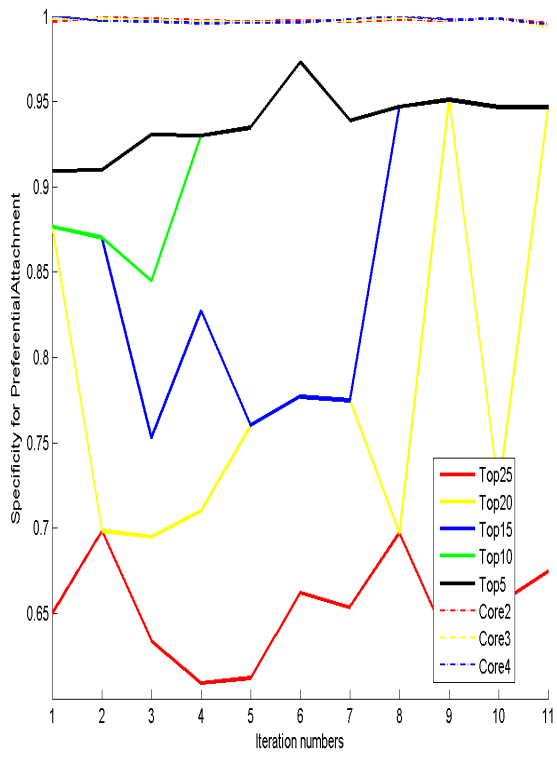
**Preferential Attachment:**



Top 5%>10%15%>20%>25%

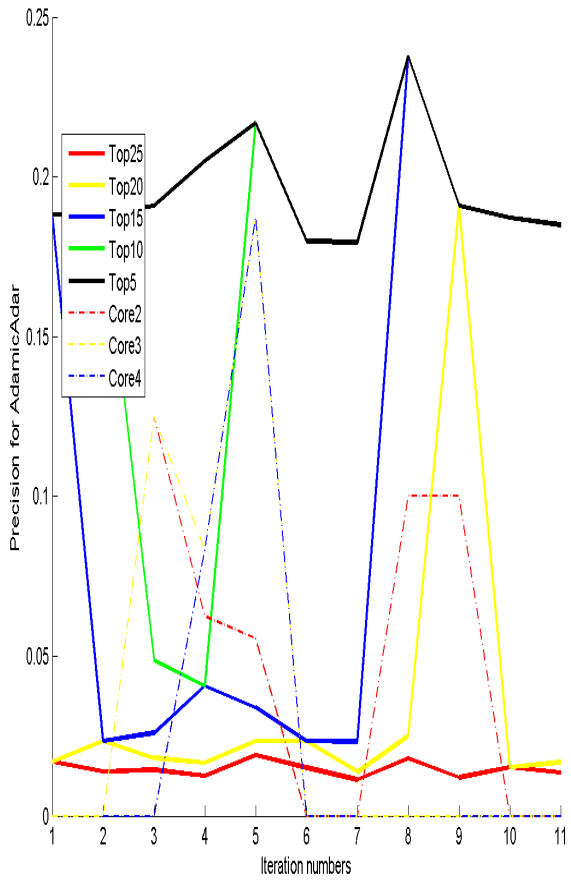


Top 25% > 20% > 15% > 10% > 5% > = Core 2, 3, and 4

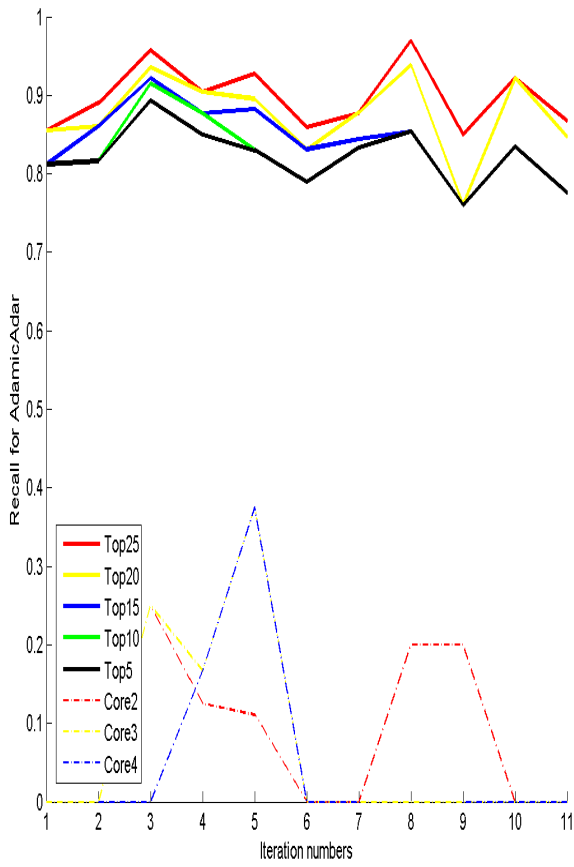


Core 2, 3, and 4  
 >Top5%>10%>15%>20%>25%

**Adamic Adar:**

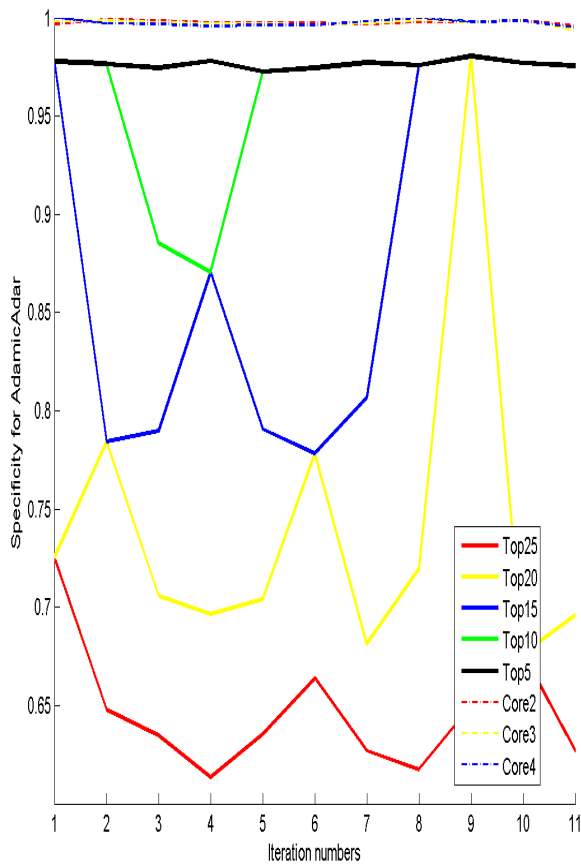


Top 5%>10%15%>20%>25%



Top 25%>20%>15%>10%>5%  
 >=Core 2, 3, and 4



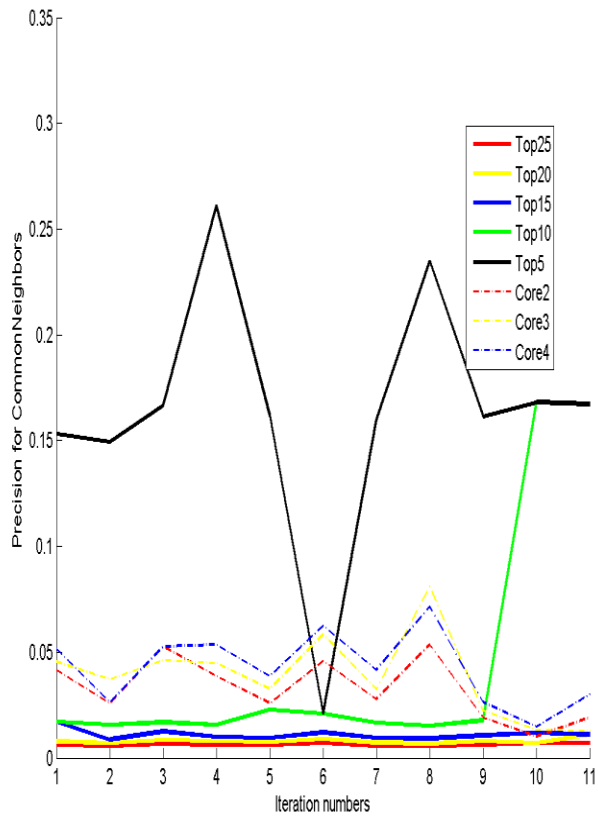


Core 2, 3, and 4  
>Top 5%>10%>15%>20%>25%

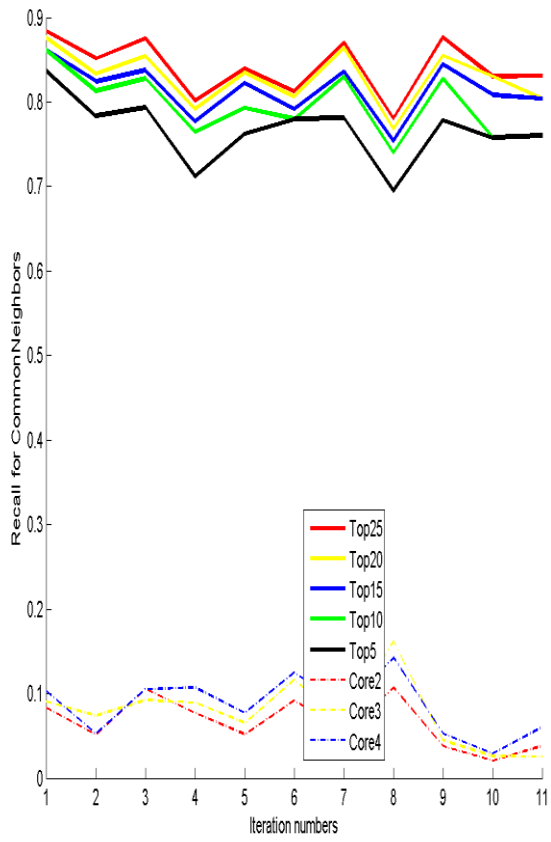
### Results with Sub-dataset-3 - 800 Nodes:

The experiment was again repeated for randomly selected 800 nodes, and the results followed the similar pattern as that of randomly selected 400 and 600 nodes.

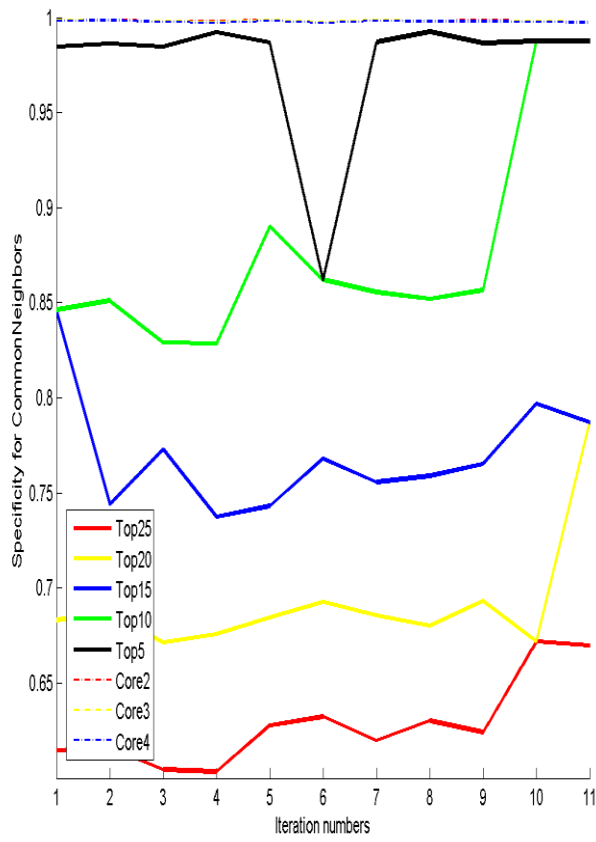
### Common Neighbors:



Top 5%>10%15%>20%>25%

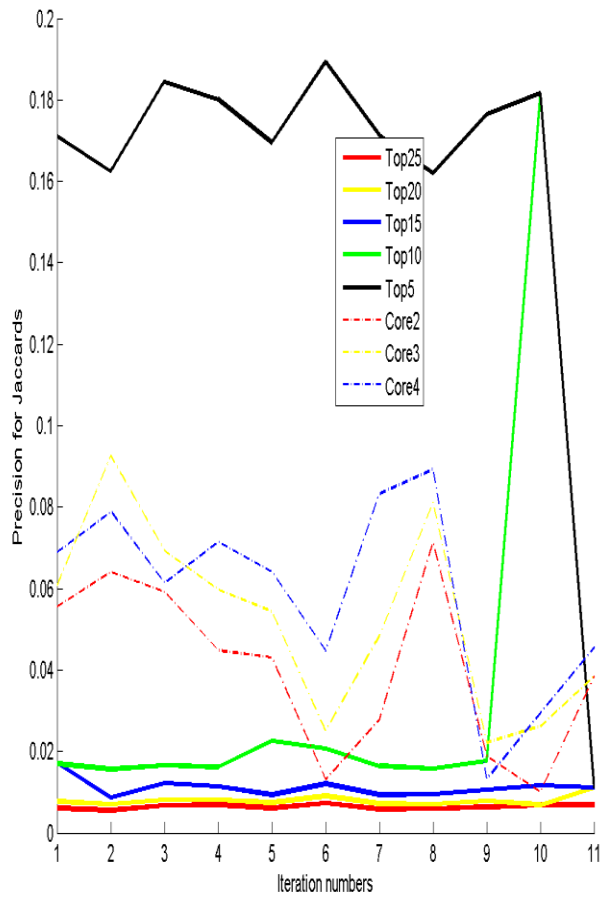


Top 25%>20%>15%>10%>5%  
 >=Core 2, 3, and 4

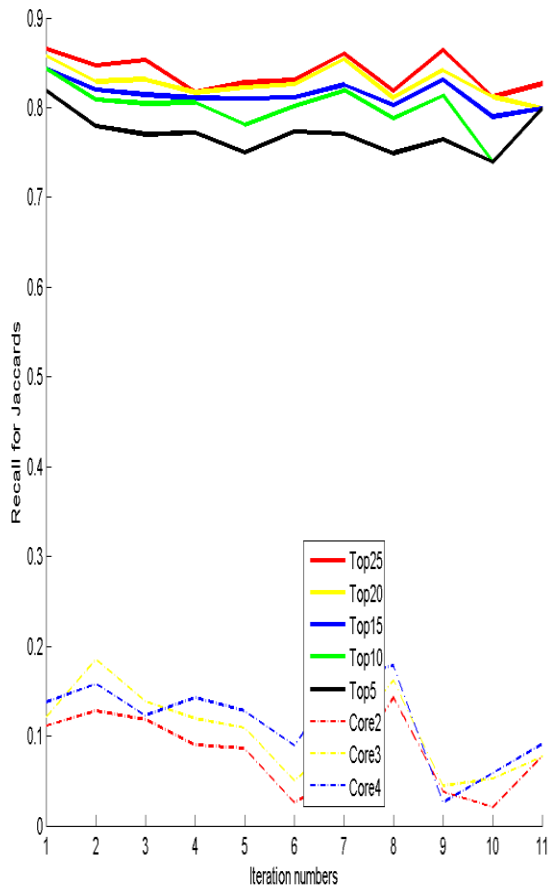


Core 2, 3, and 4  
 >Top 5%>10%>15%>20%>25%

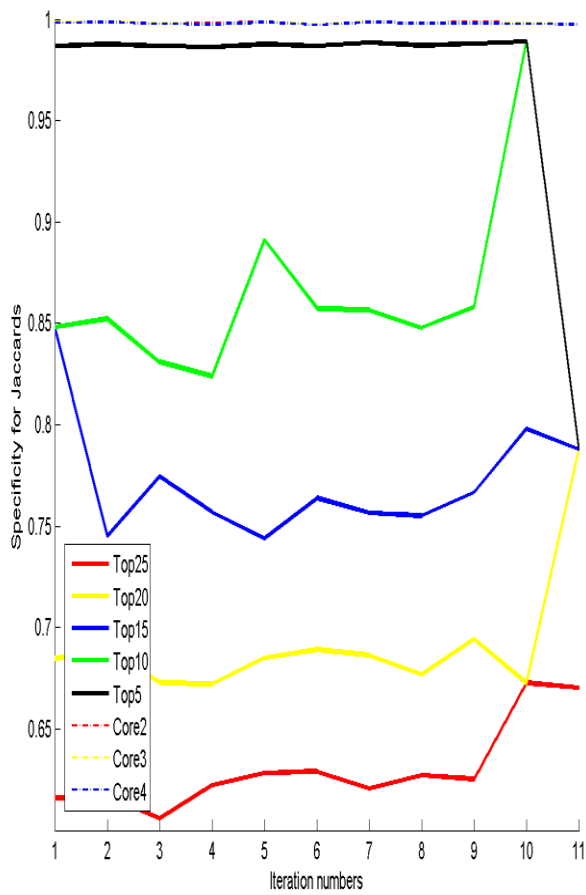
**Jaccards:**



Top 5% > 10% > 15% > 20% > 25%

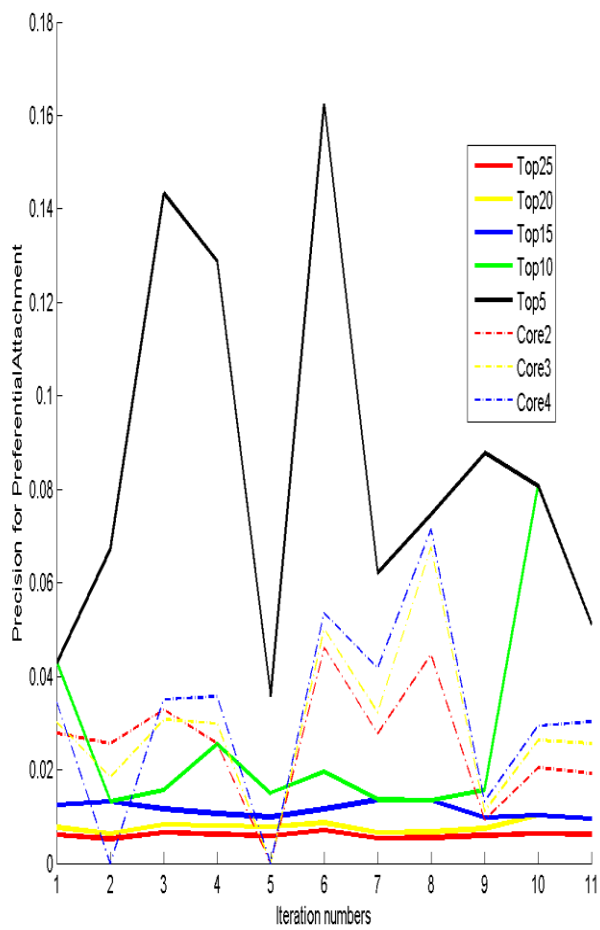


Top 25%>20%>15%>10%>5%  
 >=Core 2, 3, and 4



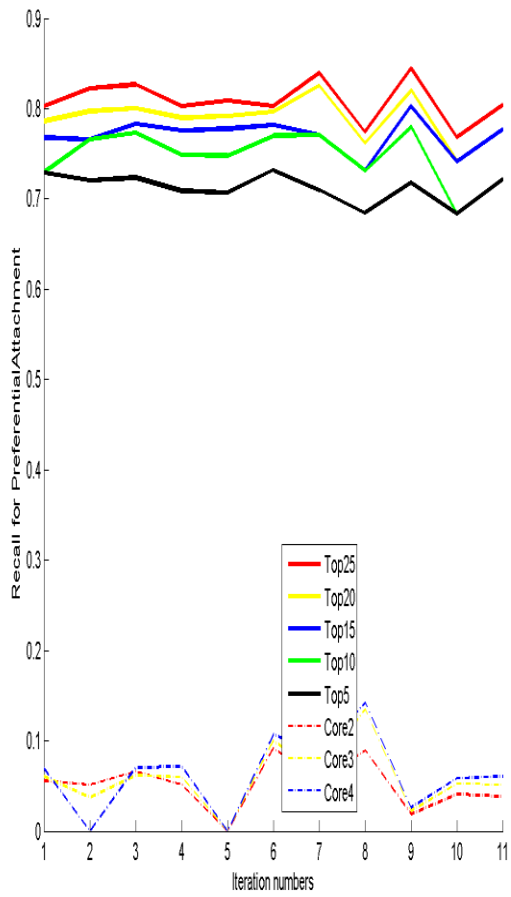
Core 2, 3, and 4  
 >Top 5%>10%>15%>20%>25%

**Preferential Attachment:**

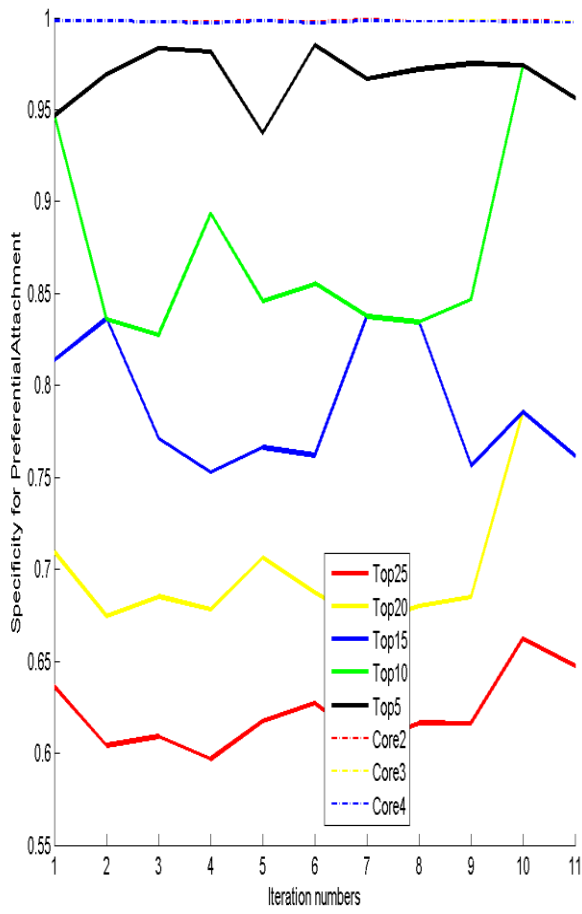


Top 5% > 10% > 15% > 20% > 25%



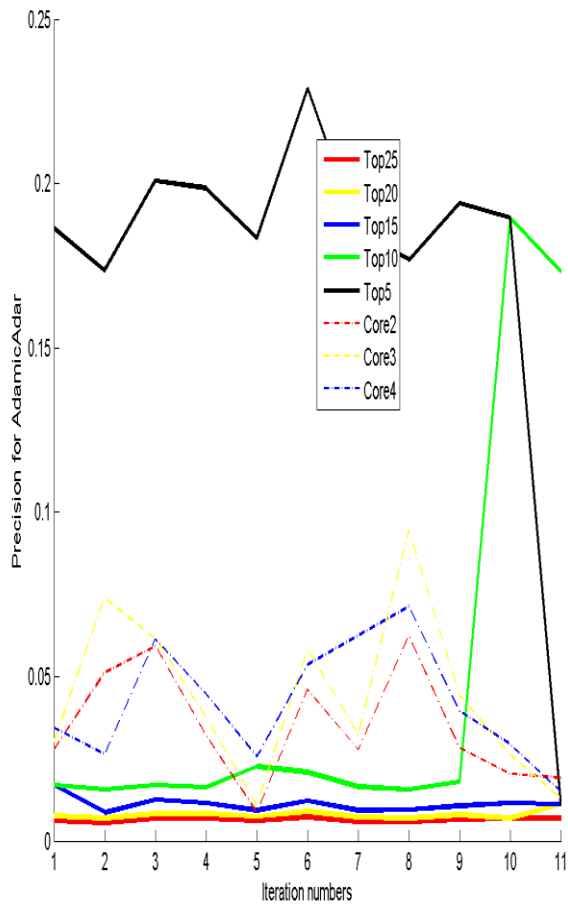


Top 25>20%>15%>10%>5%  
 >=Core 2, 3, and 4

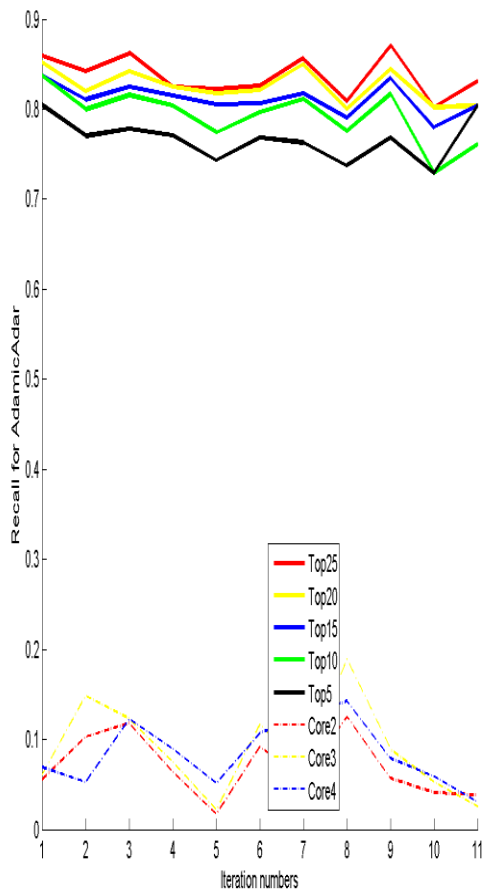


Core 2, 3, and 4  
>Top 5%>10%>15%>20%>25%

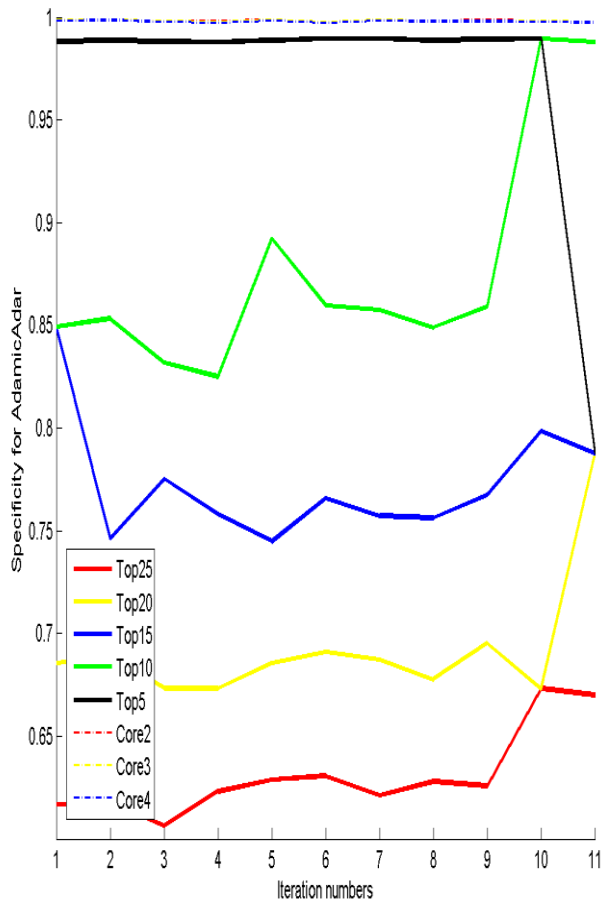
**Adamic Adar:**



Top 5%>10%15%>20%>25%



Top 25% > 20% > 15% > 10% > 5%  
 >= Core 2, 3, and 4



Core 2, 3, and 4  
>Top 5>10>15>20>25%

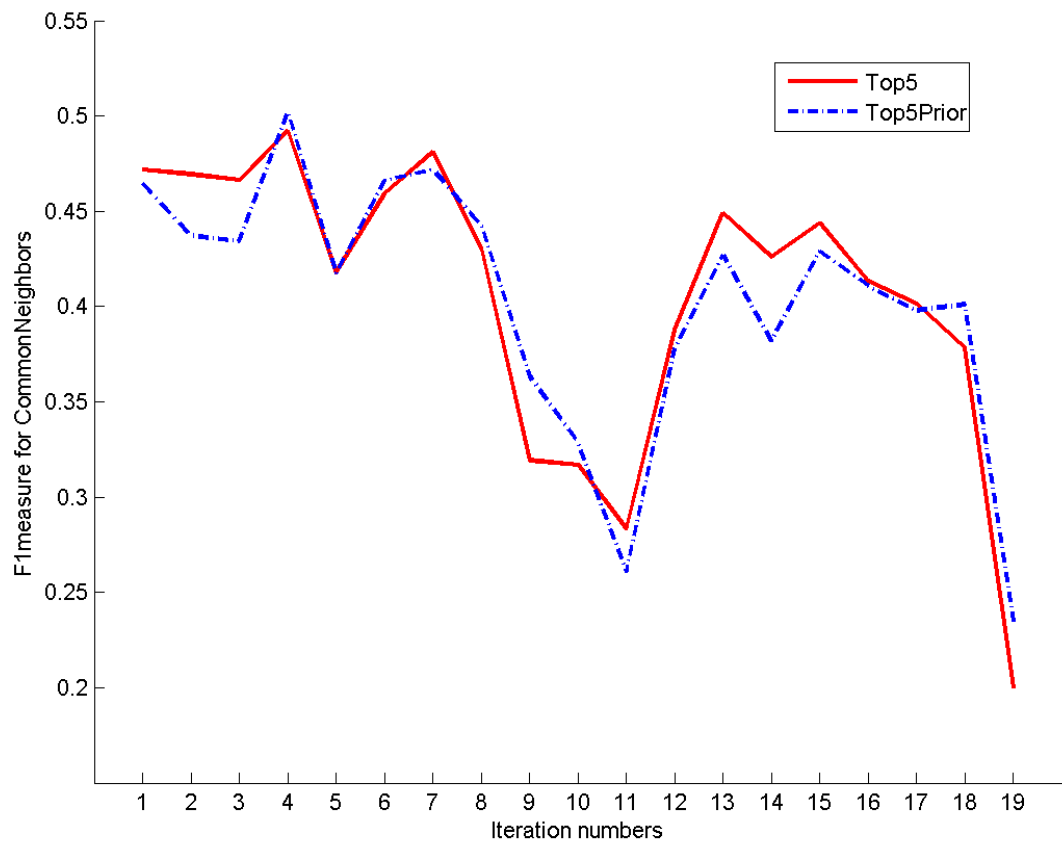
### 4.5.3 Enron dataset – Comparing results of our algorithm with different version of Prior Probability with the first version

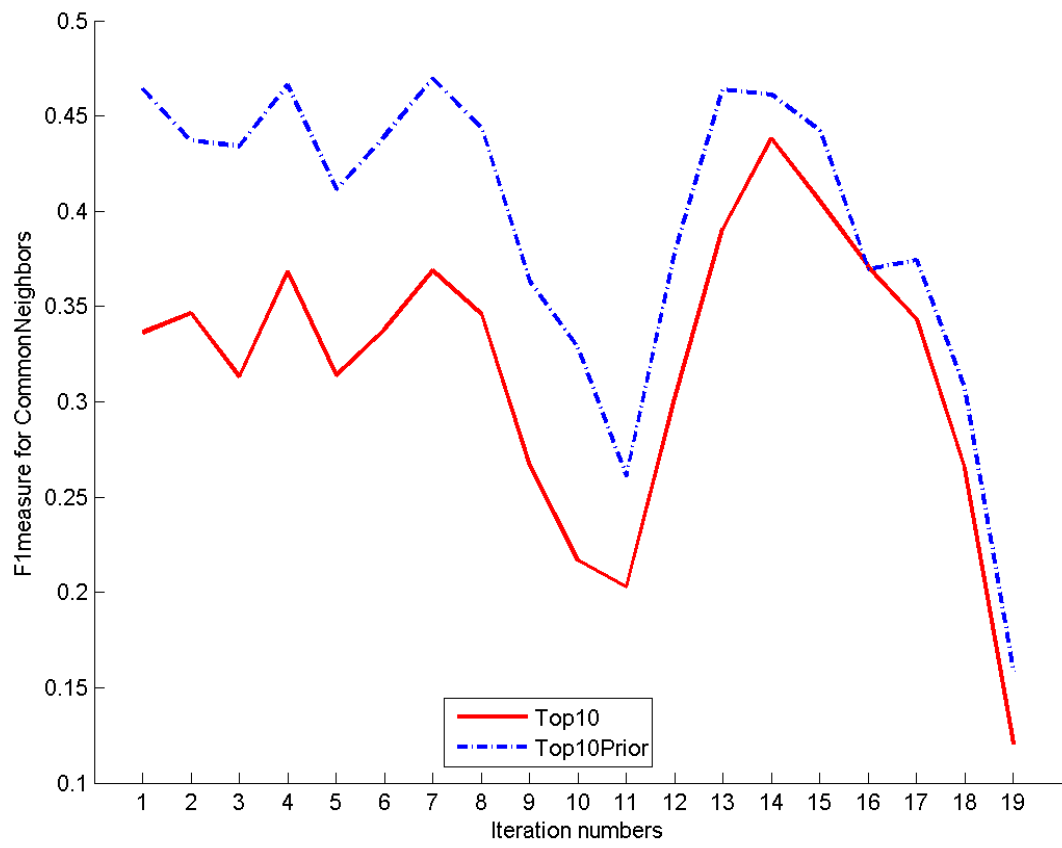
- As mentioned earlier in the ‘approach’ part, we experimented with the way prior probability was calculated
- Detailed analysis of results by running algorithm with this different version of prior probability proved that these results are comparable to the first version results in precision and F-1 Measure, but it did not have good recall percentage
- And it performed better than the Nowell’s algorithm in precision and recall percentage

## **Common Neighbors:**

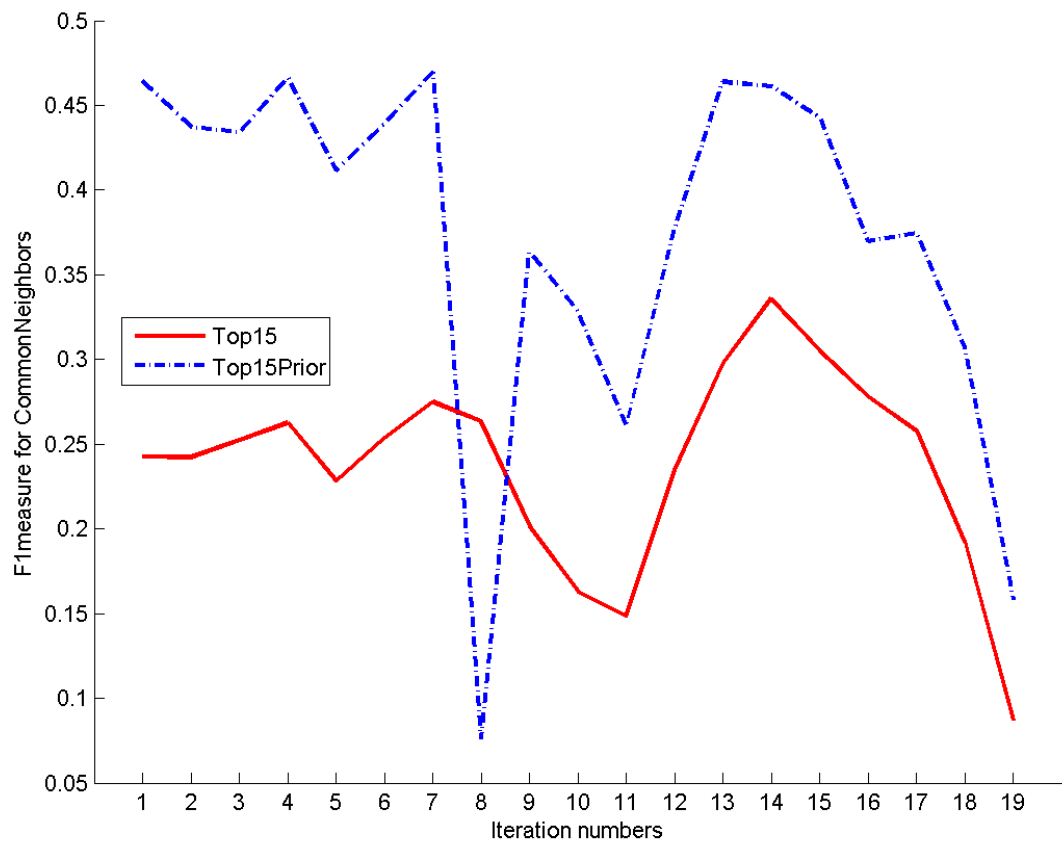
### F-1 Measure:

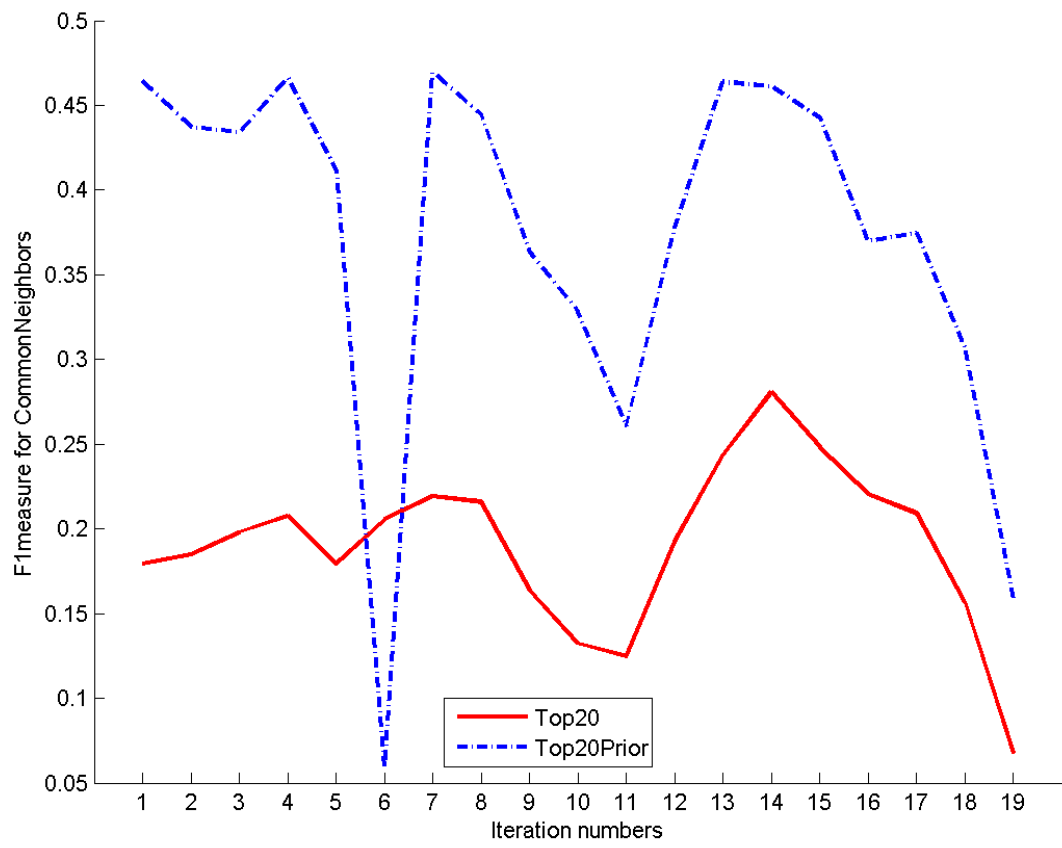
1. Original version performs better than the second version looking at the graphs
2. However, as the number of edges that are predicted(top %) increases, the F-1 Measure values of the original versions come closer to the second version's values

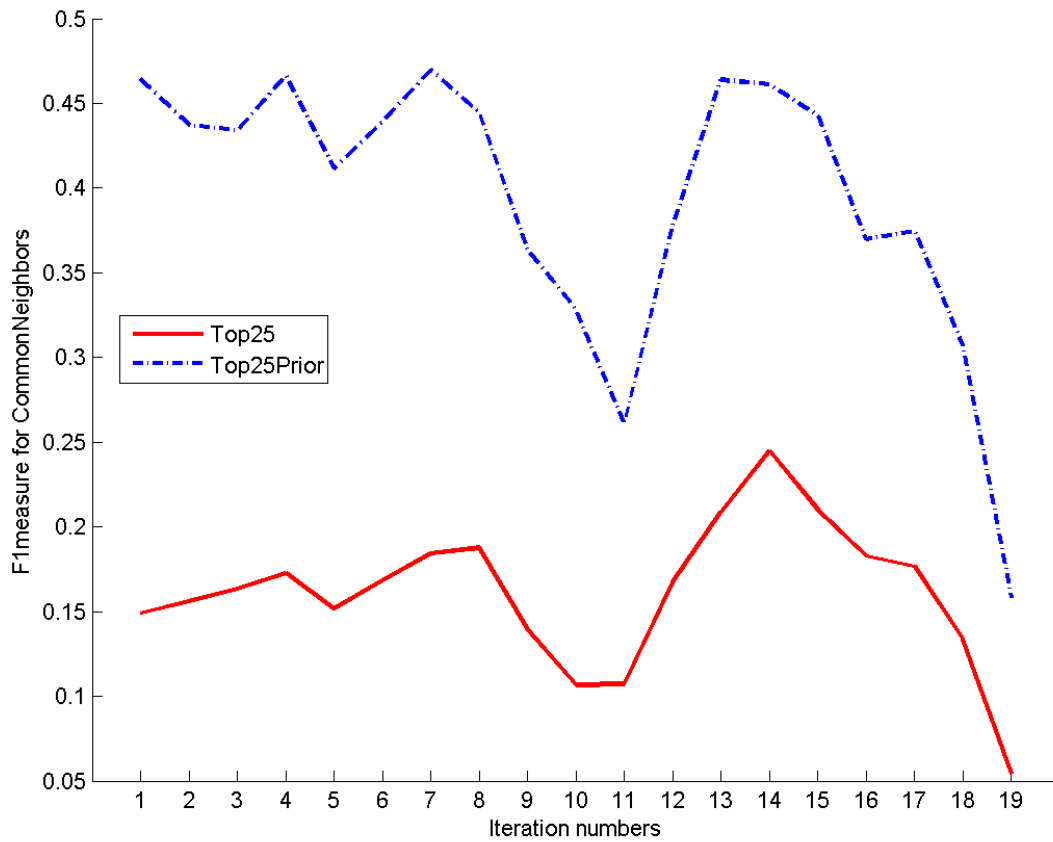






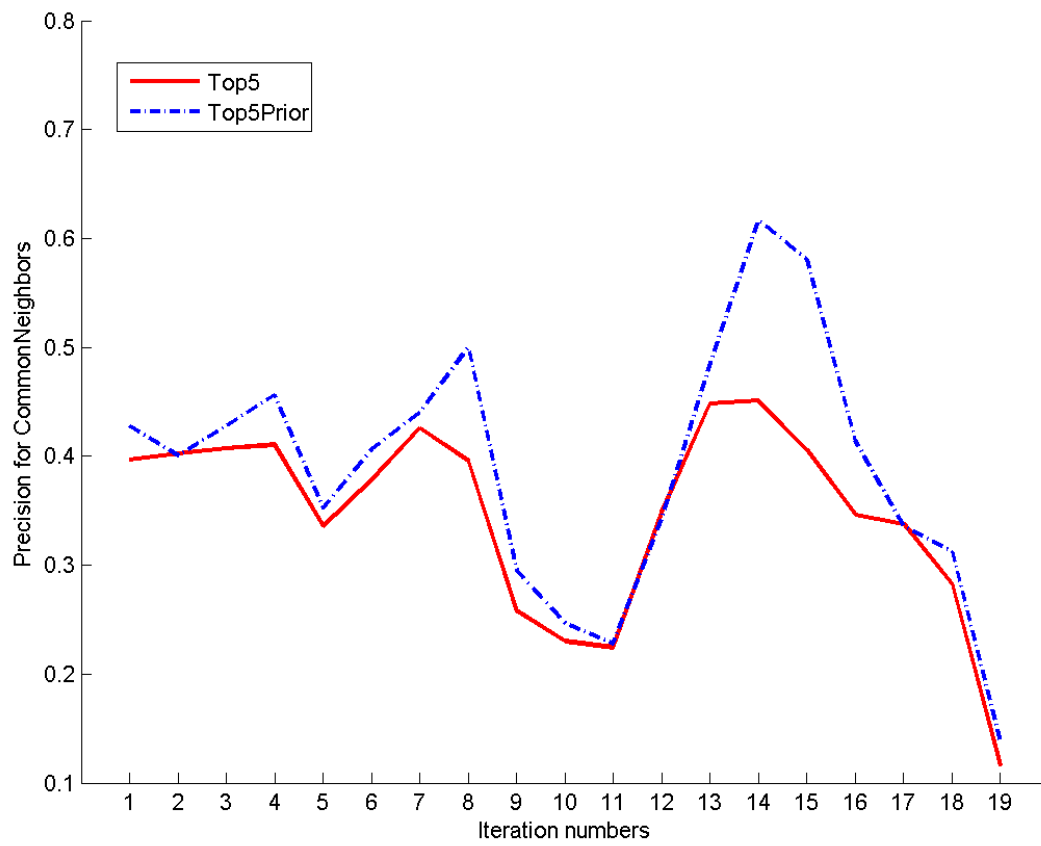


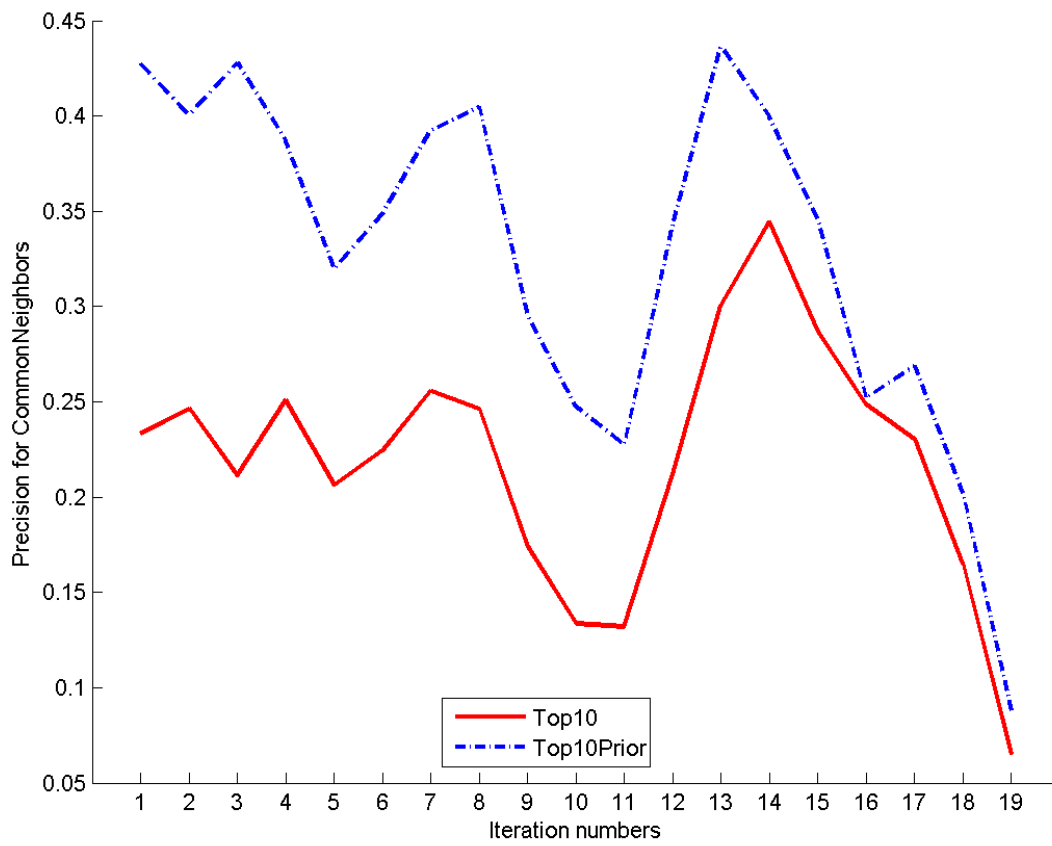


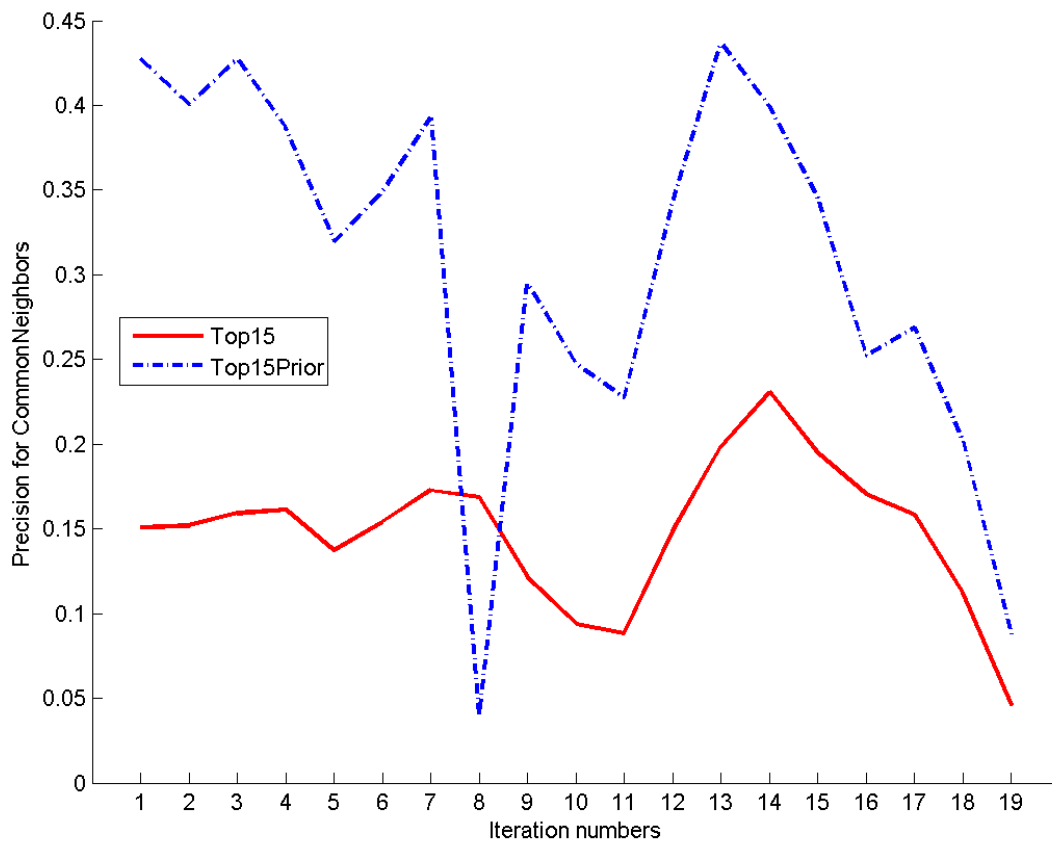


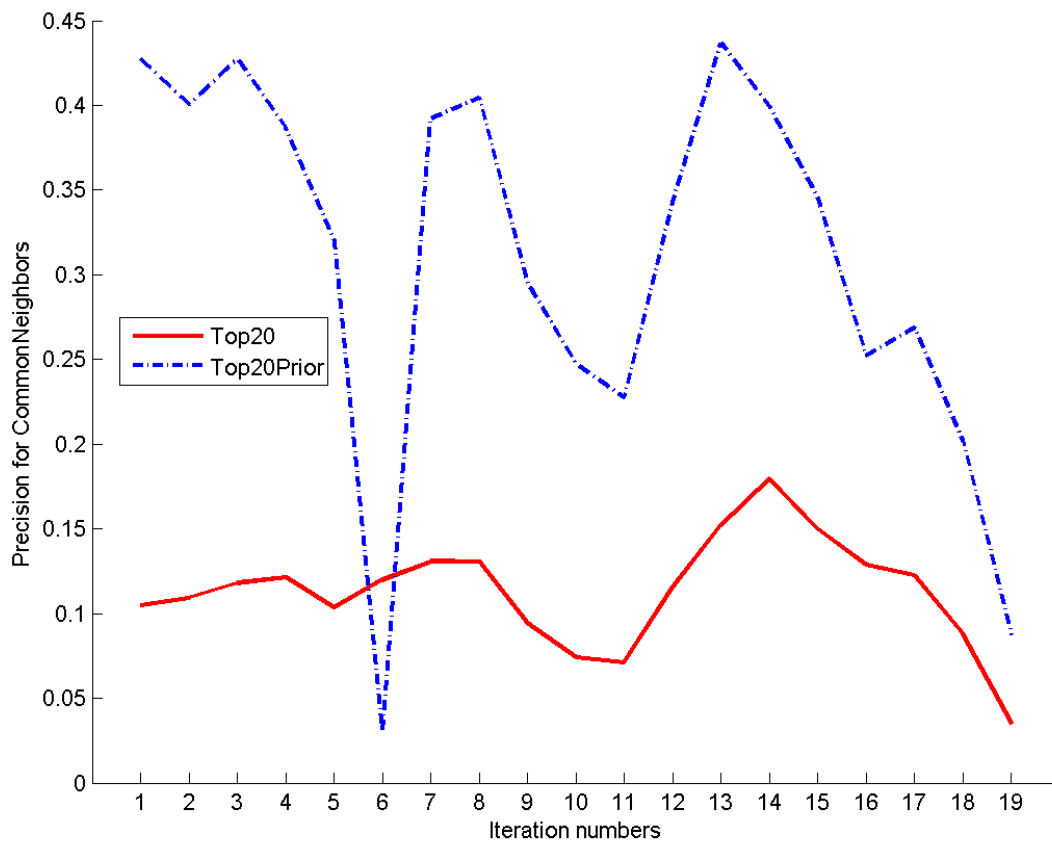
### Precision:

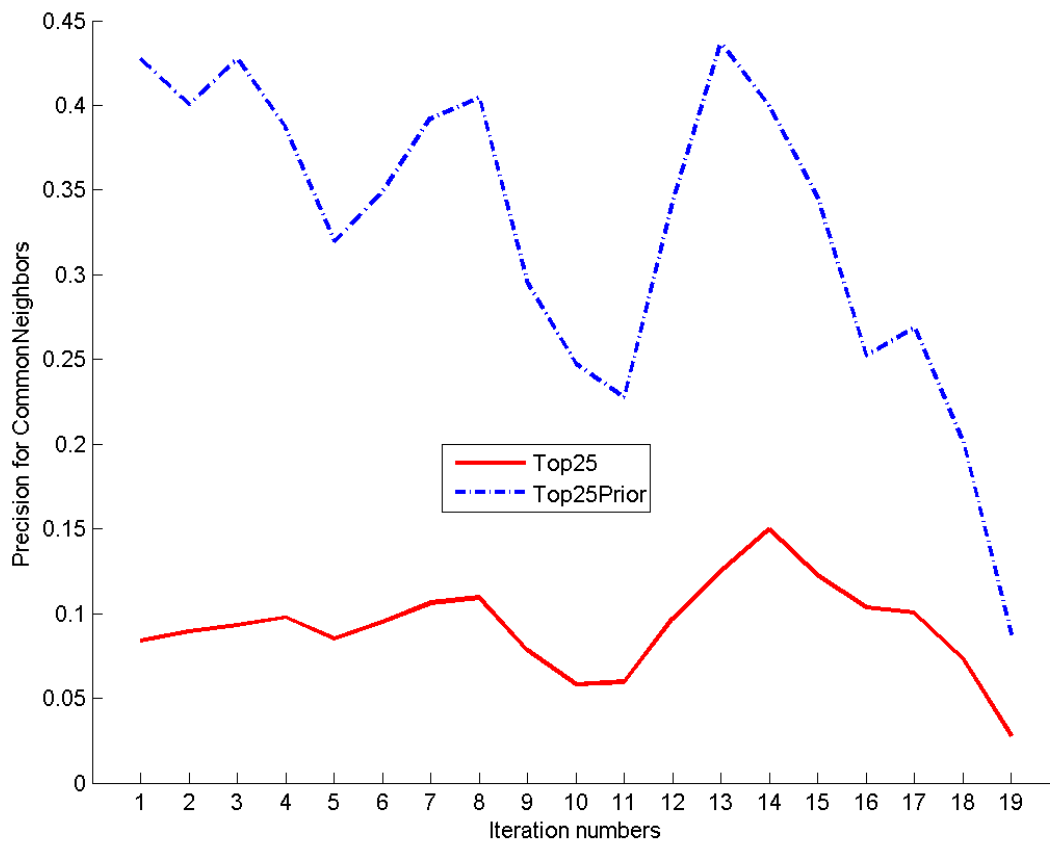
Precision values follow the same pattern as that of F-1 Measure. While the F-1 Measure of second version remains constant as more top % of edges are predicted, the measures of original version decrease.







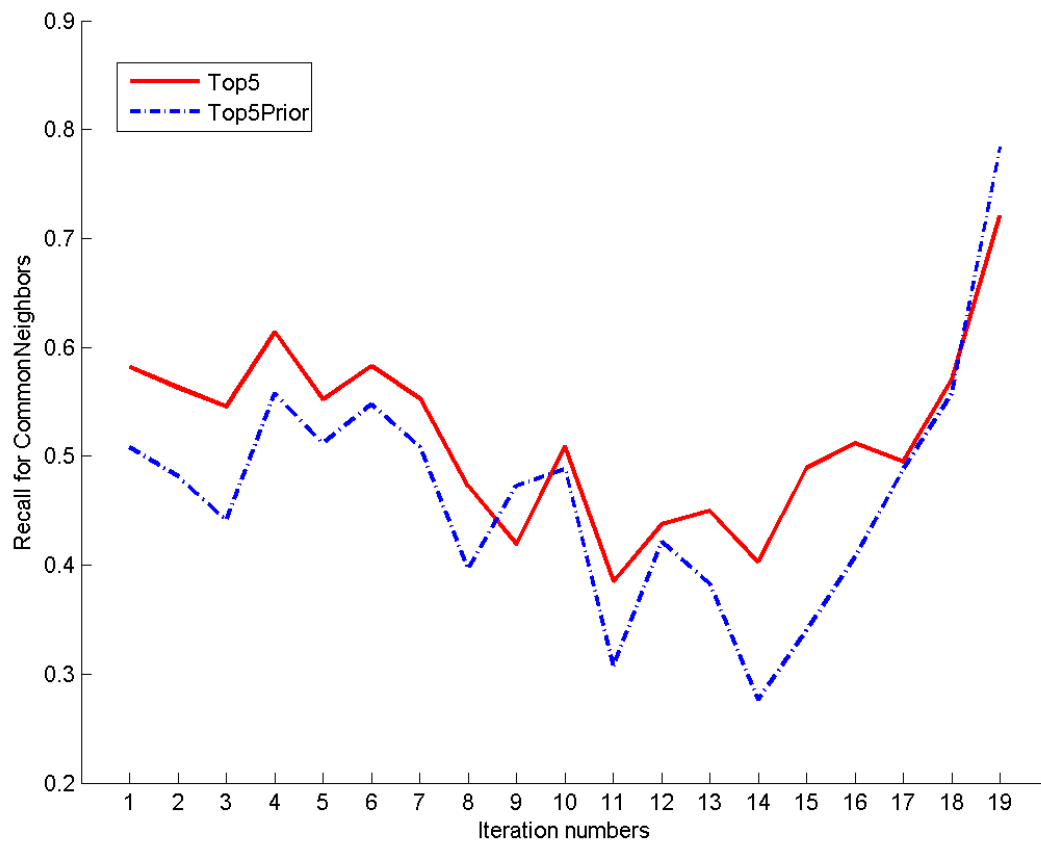


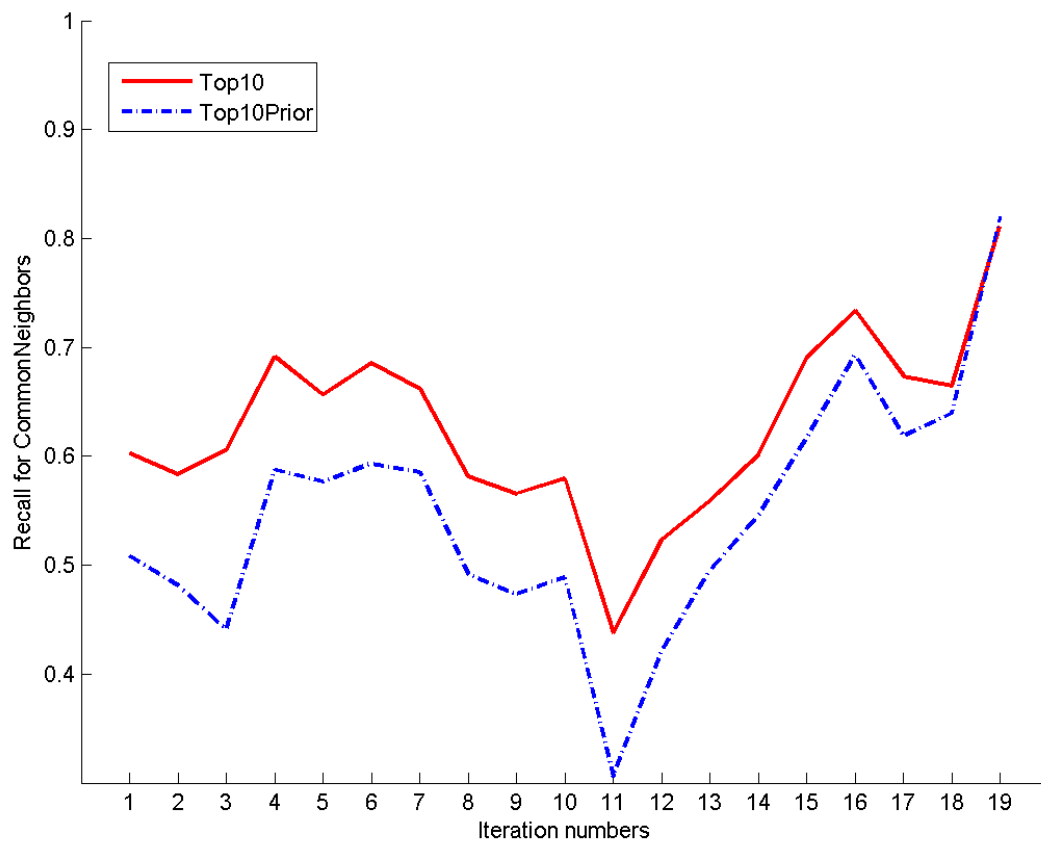


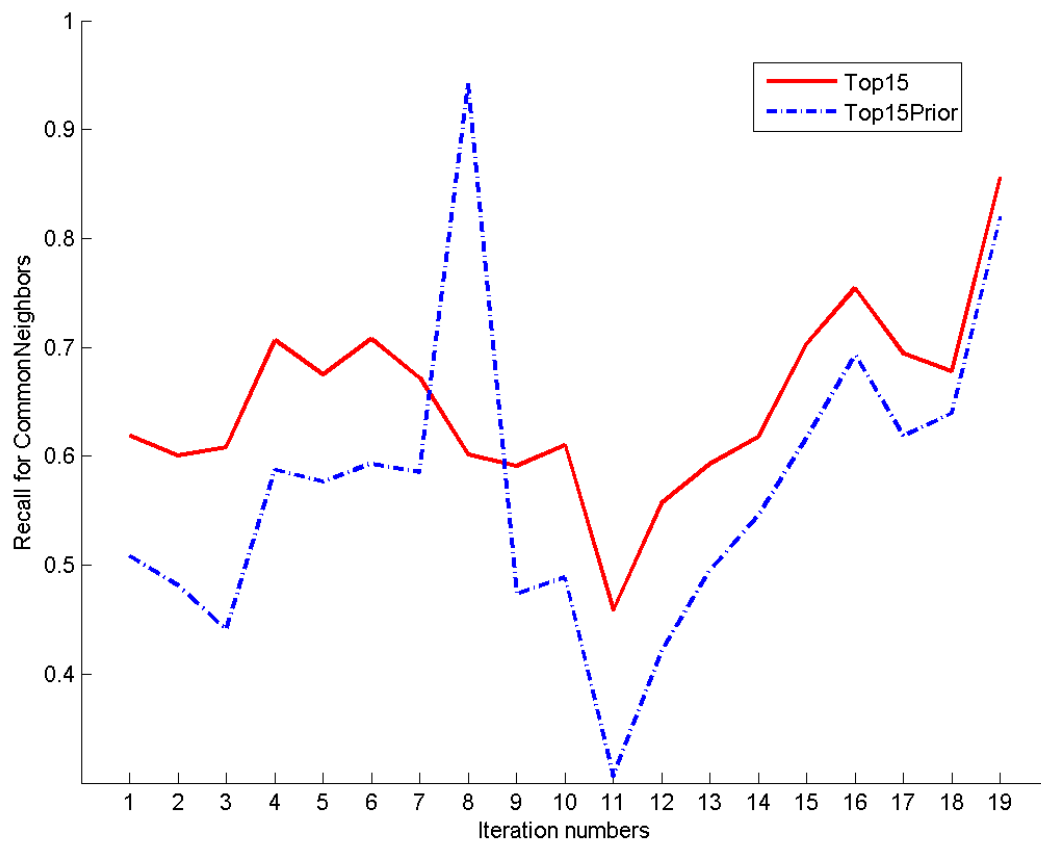
Recall:

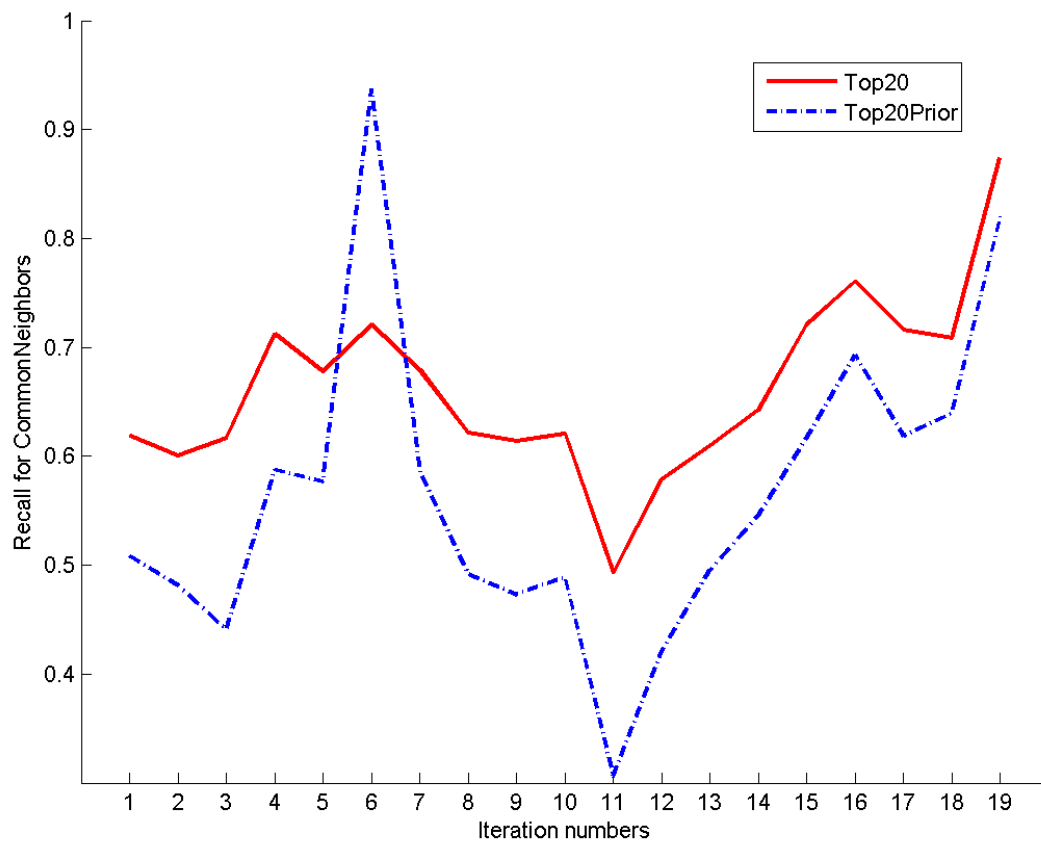
For all 5 variants of top % of original version, recall value of the original version outperforms the second version.

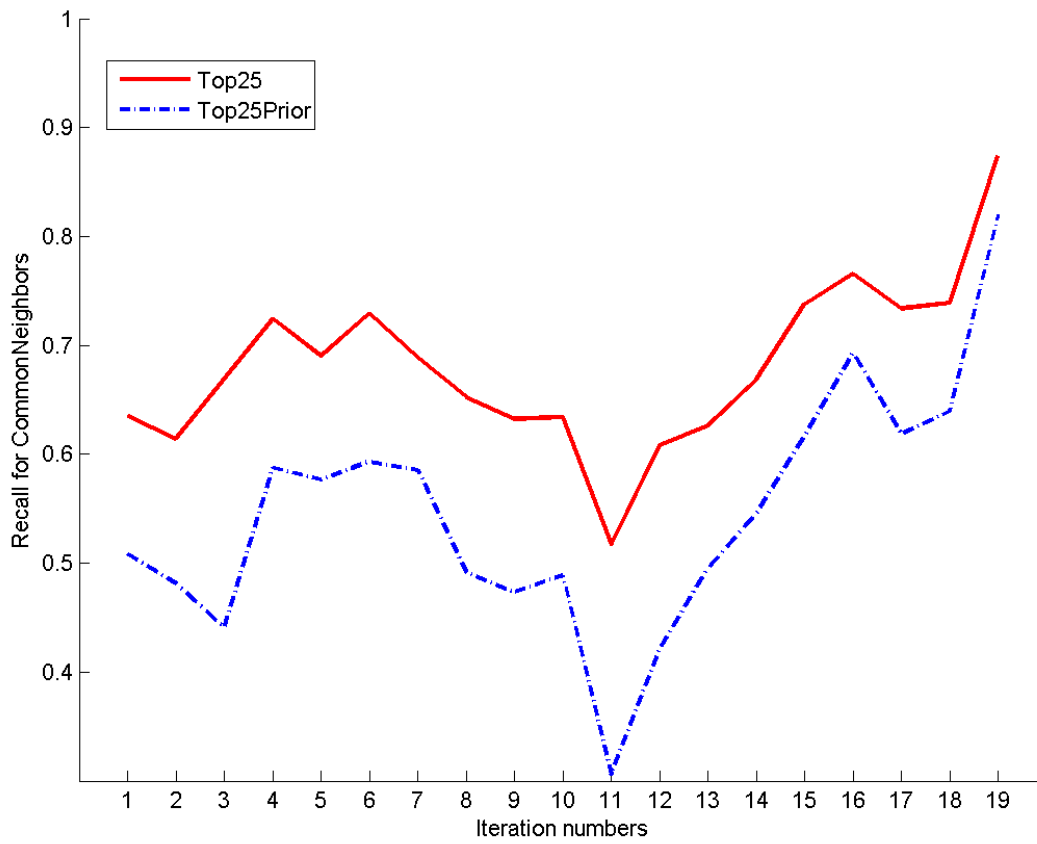






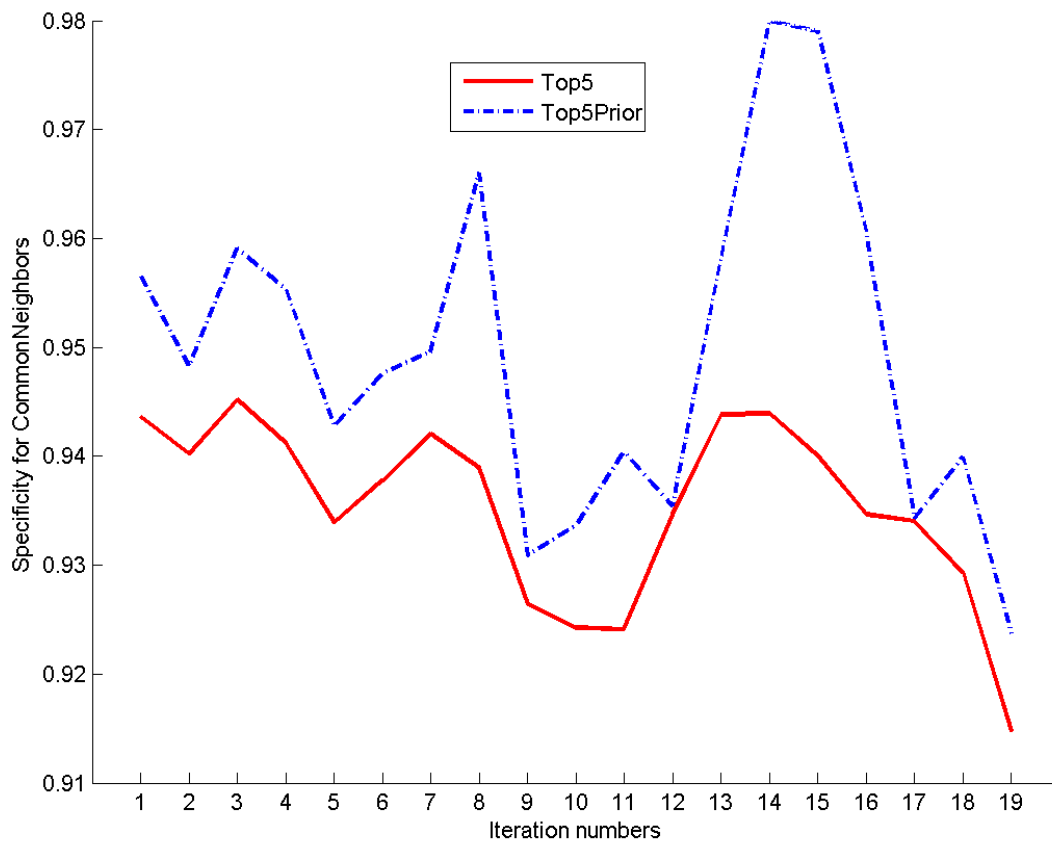


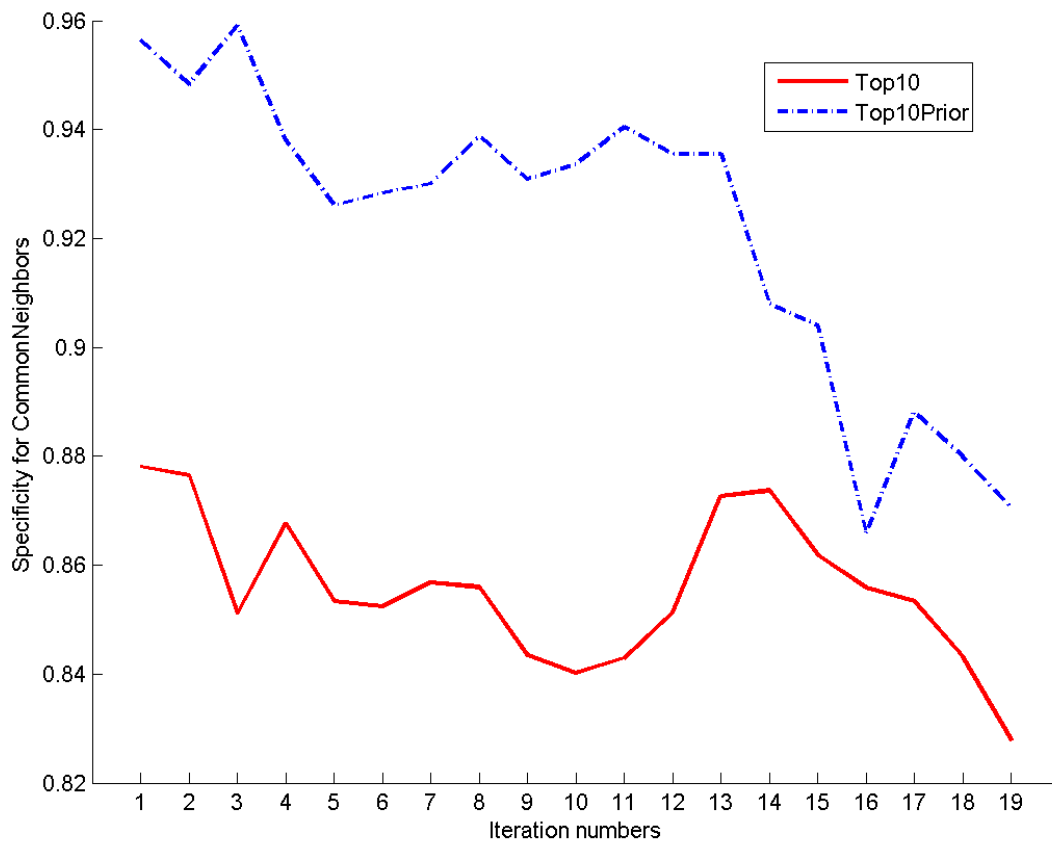


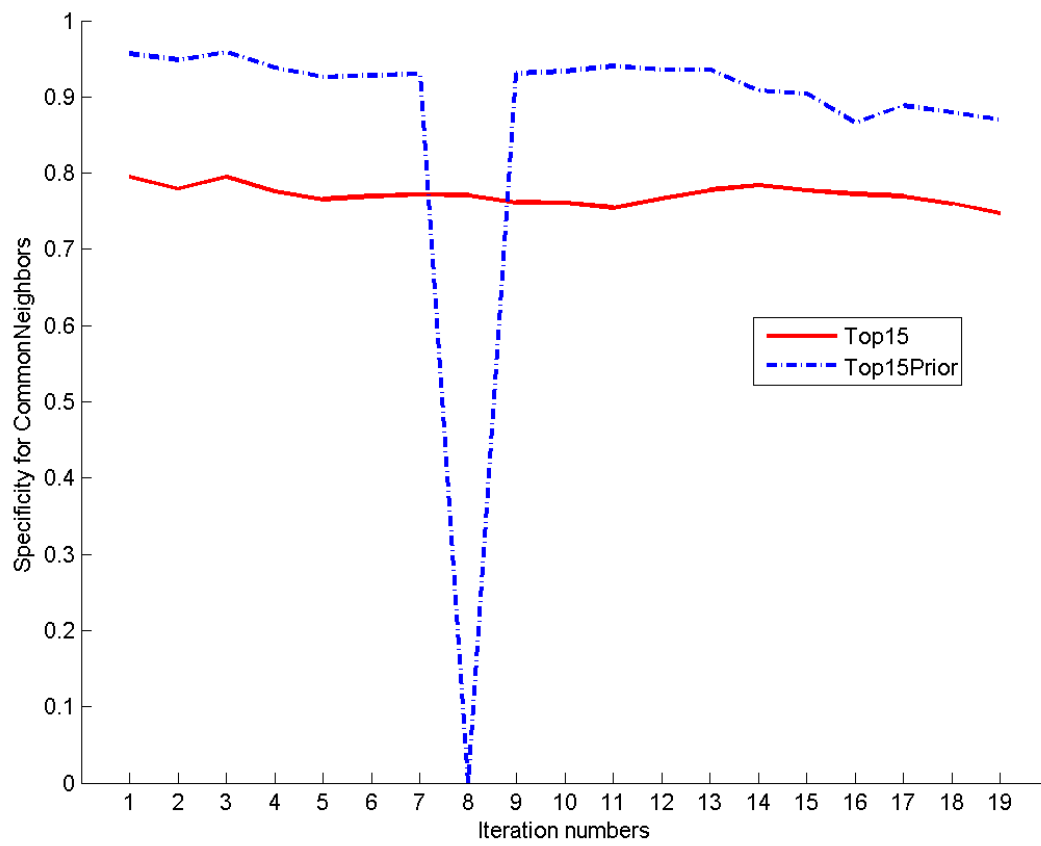


Specificity:

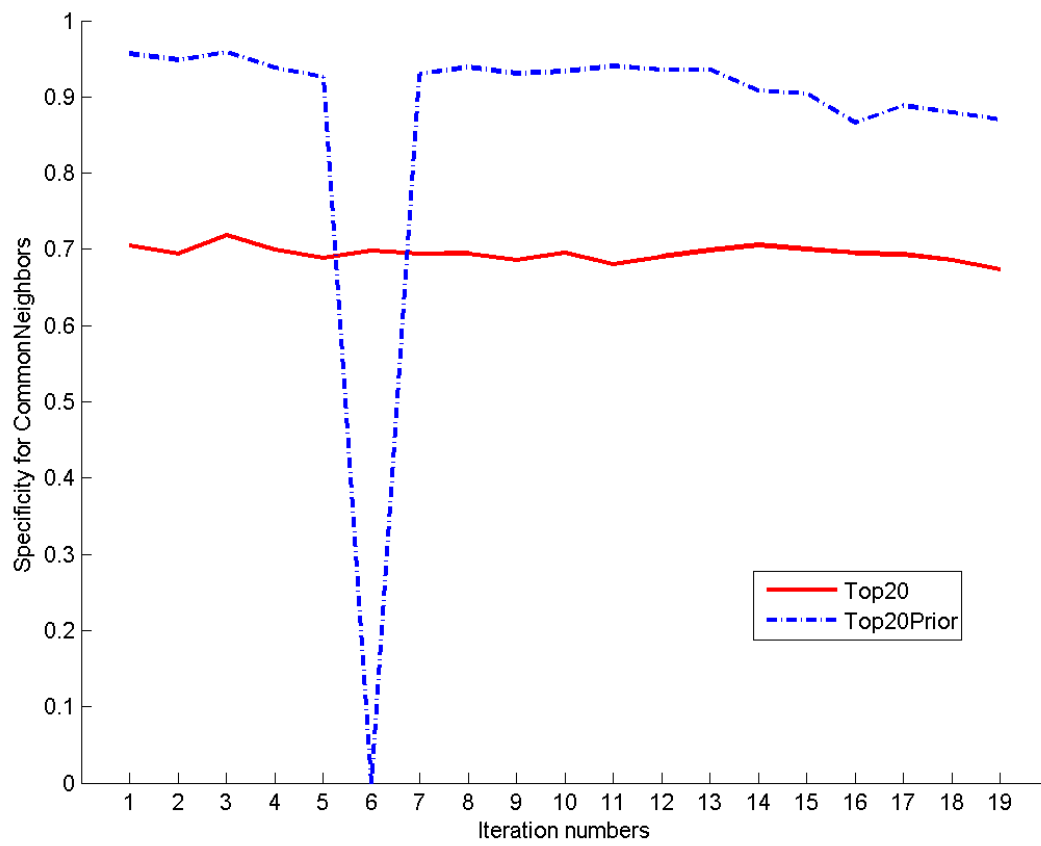
Comparatively, specificity values of the second version are better than the first one.

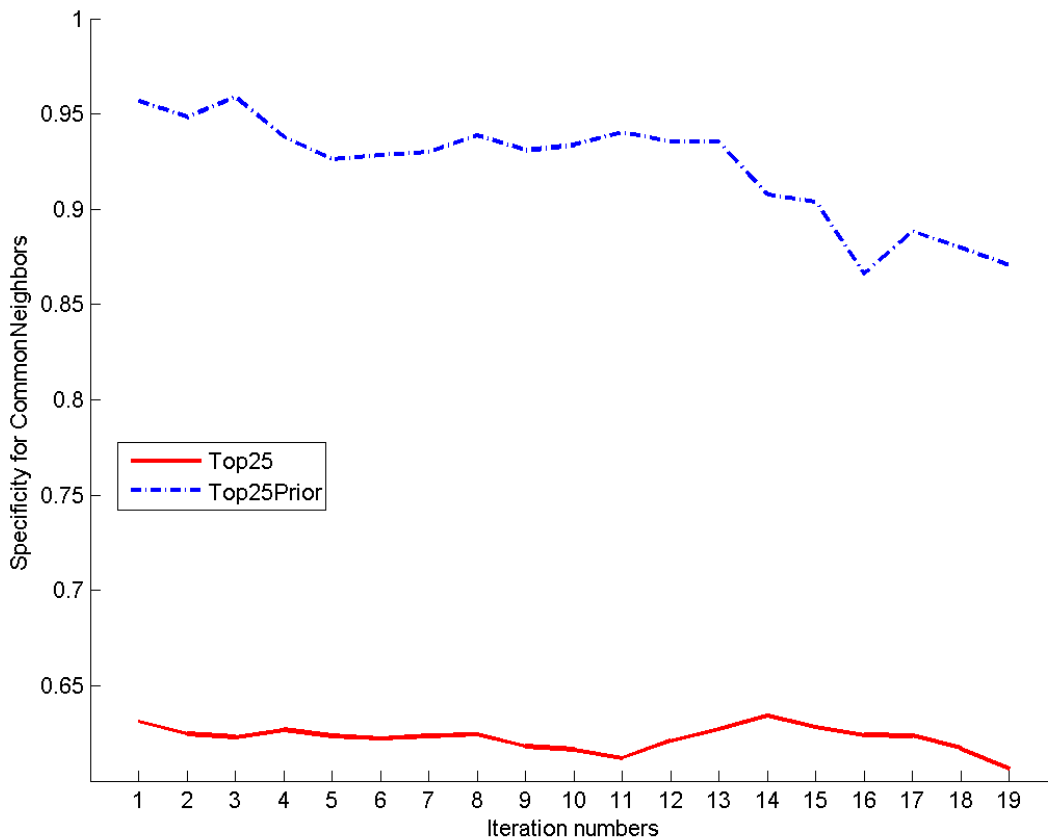












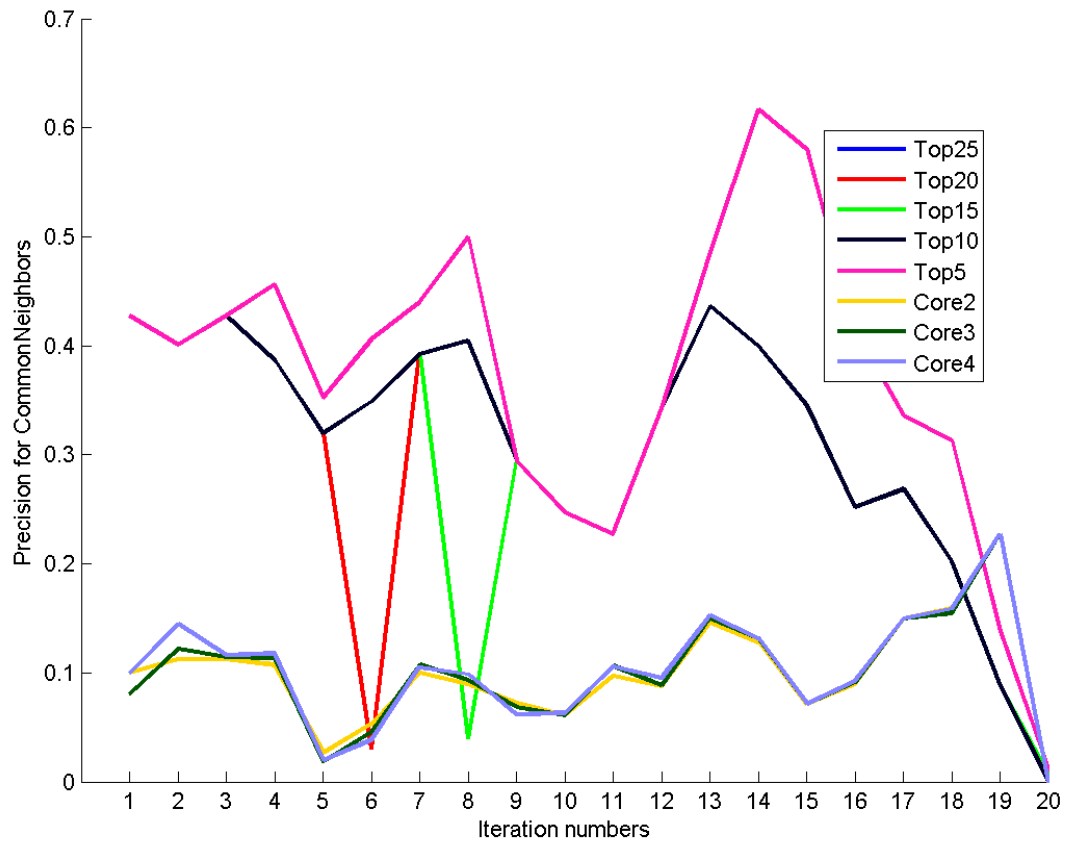
All the remaining 3 proximity measures have shown similar performance when tested against second version of our algorithm with this dataset. Please find their graphs below.

#### 4.5.4. Comparing Results of second version of our algorithm with Nowell's Algorithm

Common Neighbors:

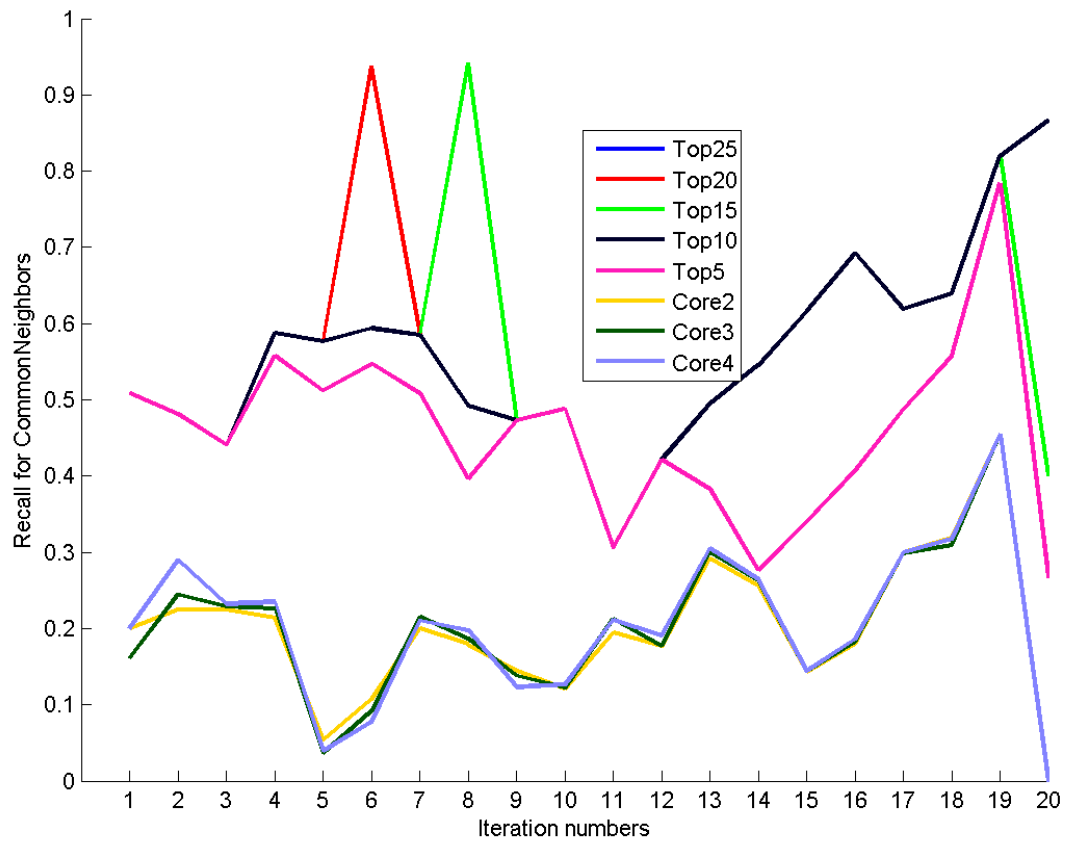
Precision:

1. Precision values of the second version with different way to computer Prior Probability are better than the Nowell's algorithm, but are less than the values of the first version of the algorithm
2. Remaining 3 features exhibit almost similar behavior



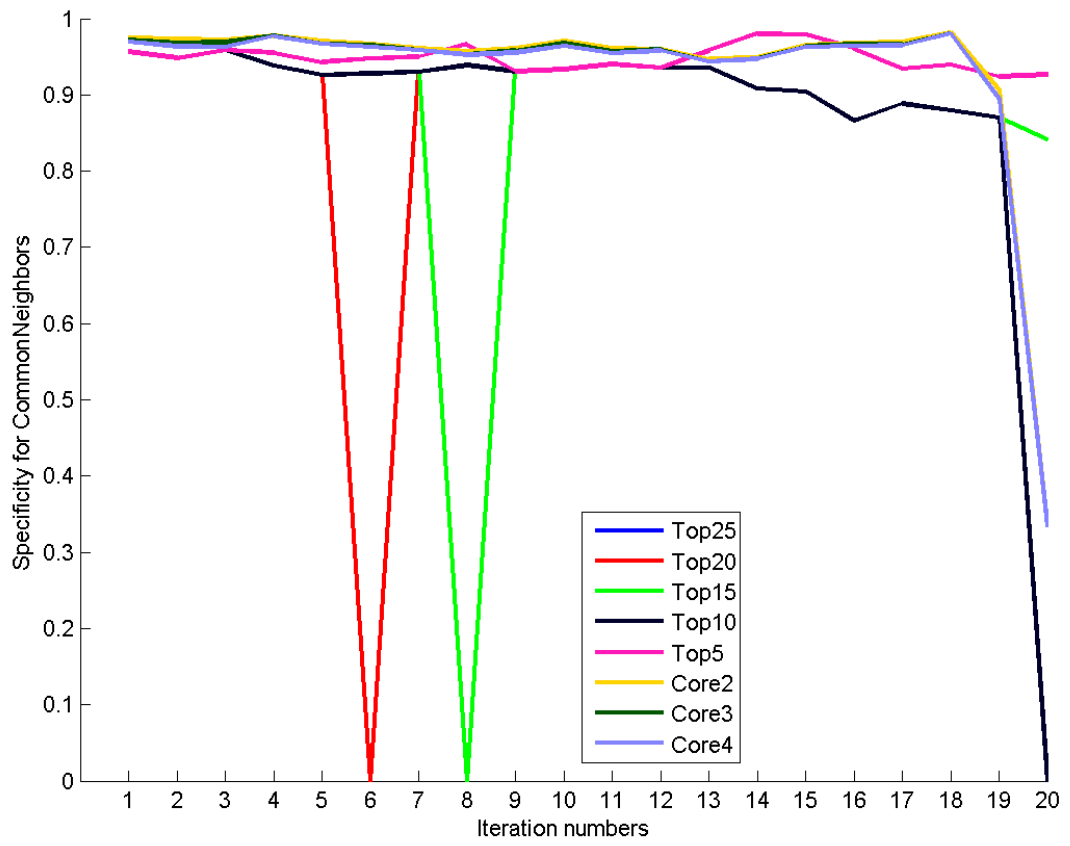
Recall:

1. Recall values of the second version are better than the previous version algorithm, but are less than the values of the original version of the algorithm
2. Remaining 3 features exhibit almost similar behavior

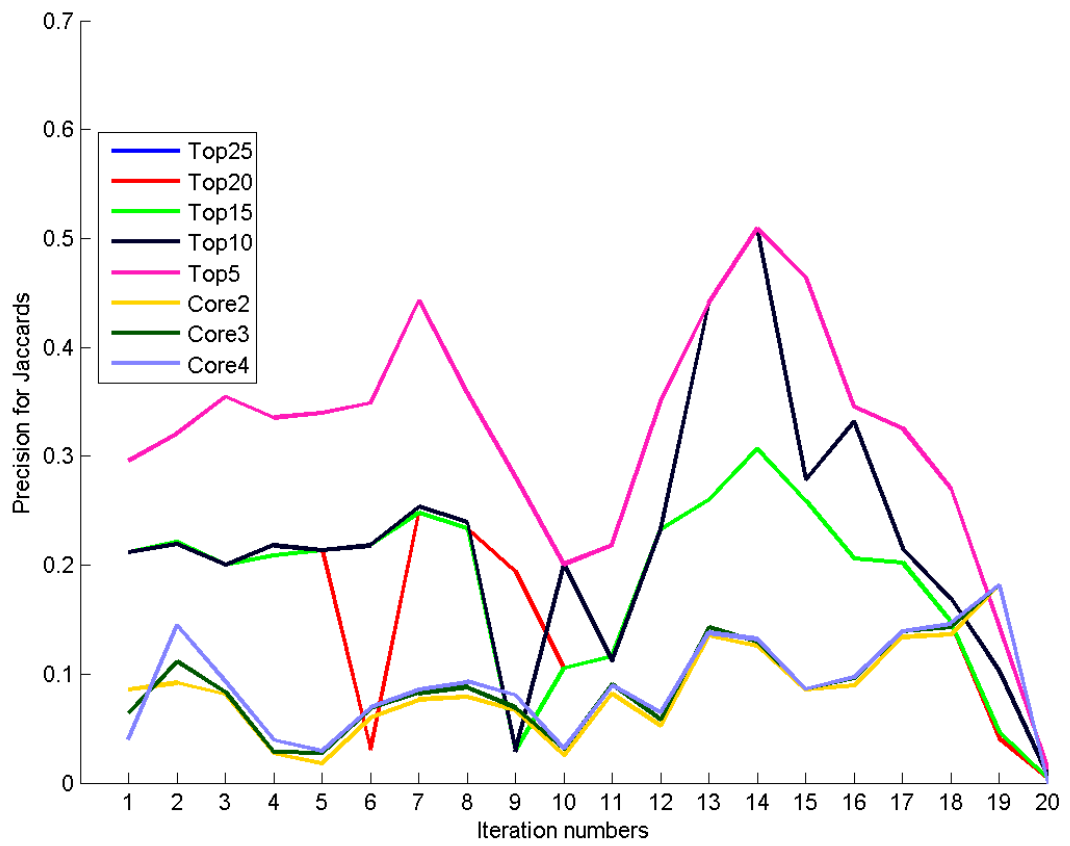


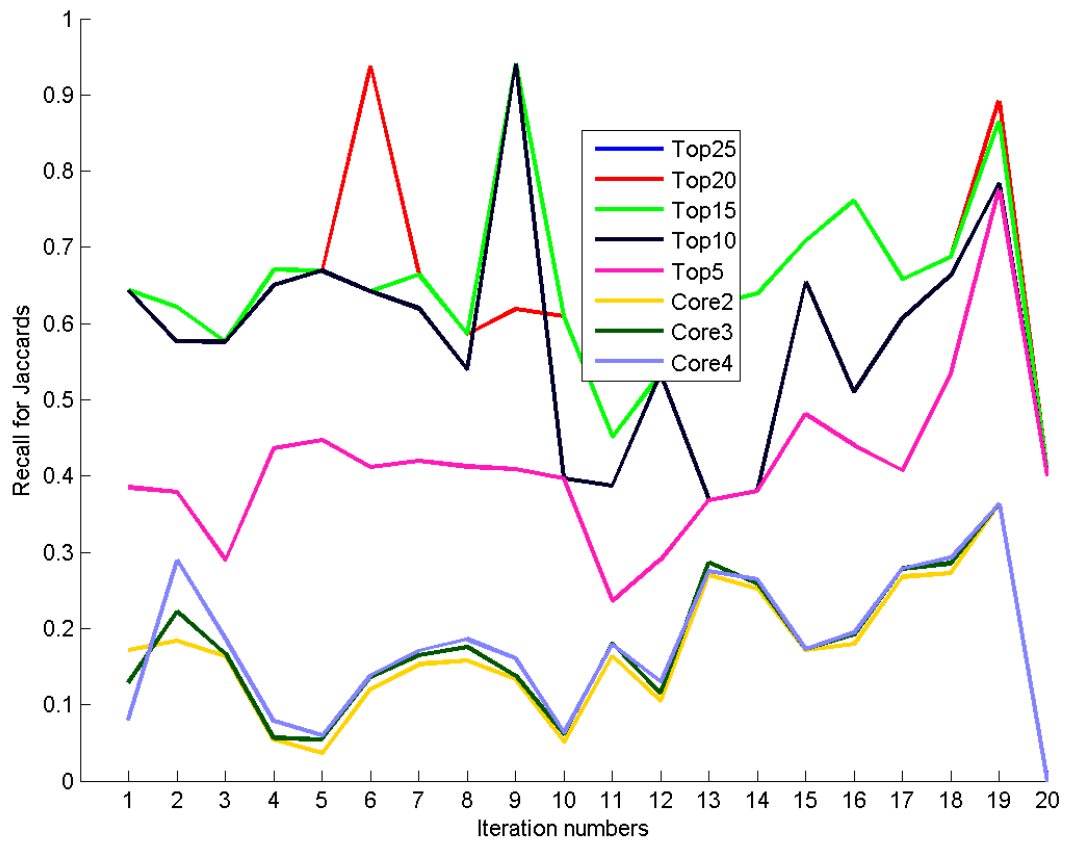
Specificity:

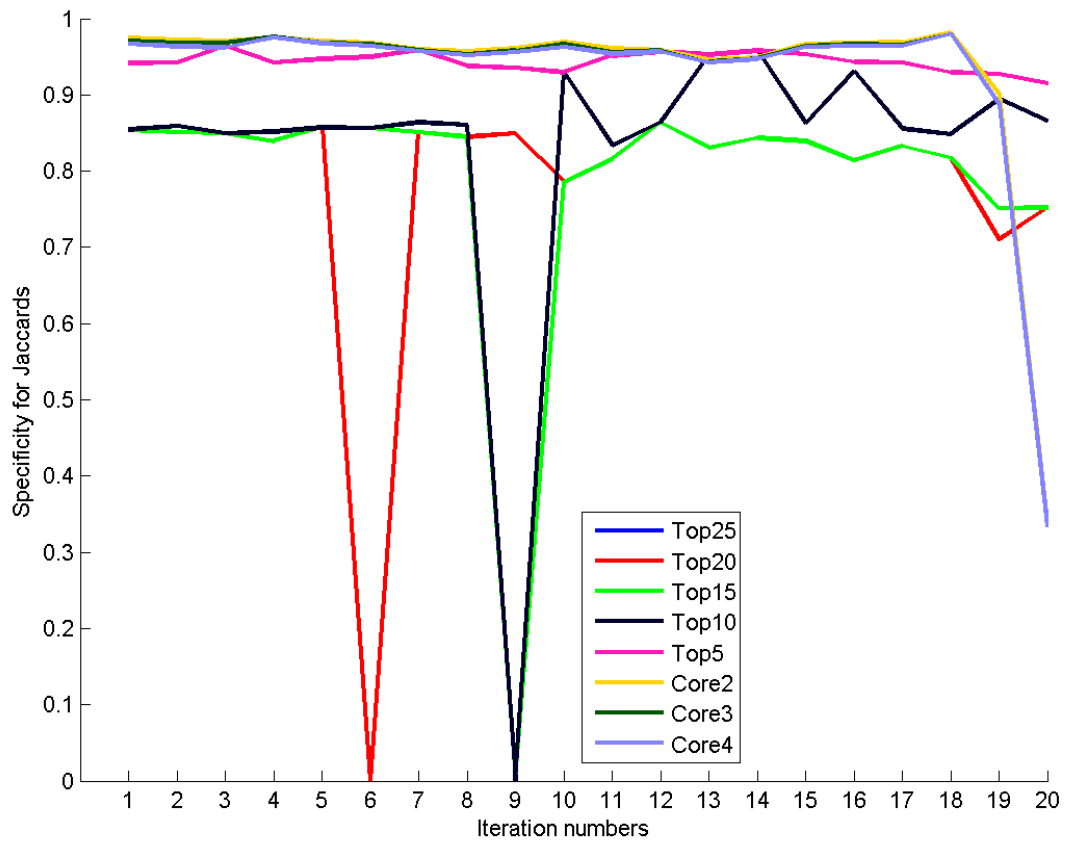
1. Specificity values of the second version of our algorithm are comparatively less than the previous algorithm, but are greater than the values of the original version of the algorithm
2. Remaining 3 features exhibit almost similar behavior



**Jaccards:**

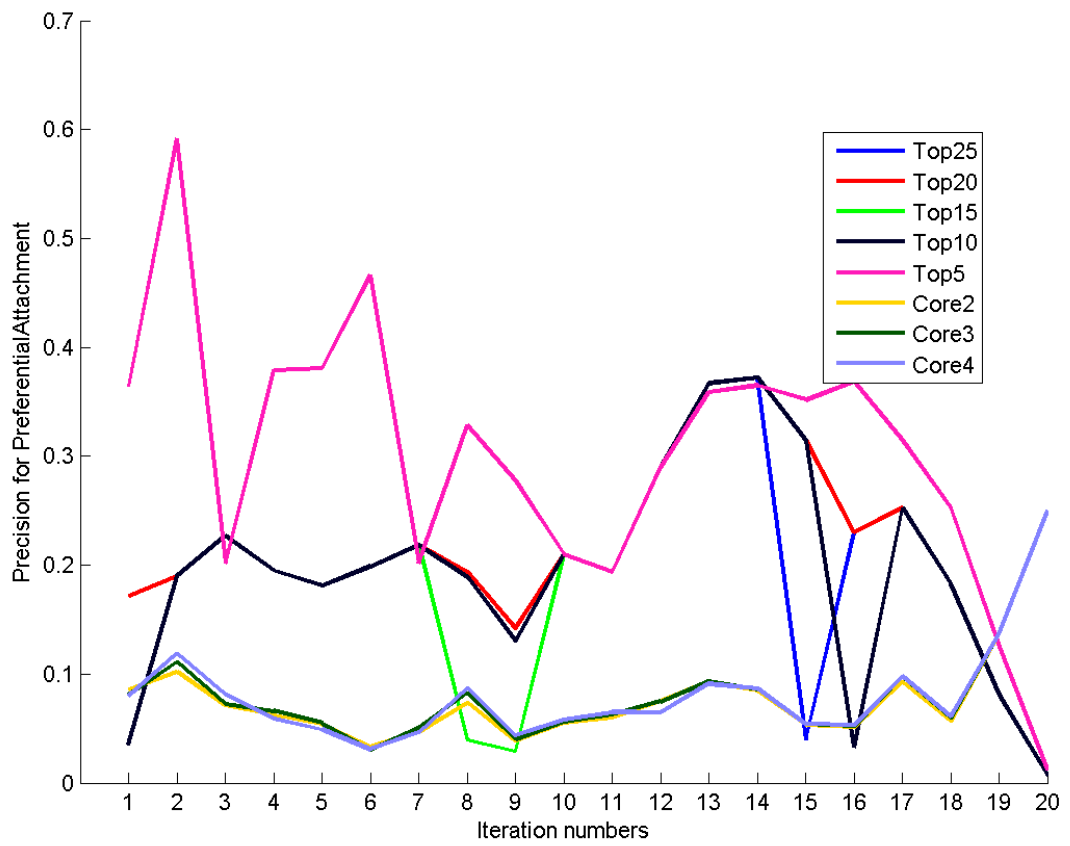


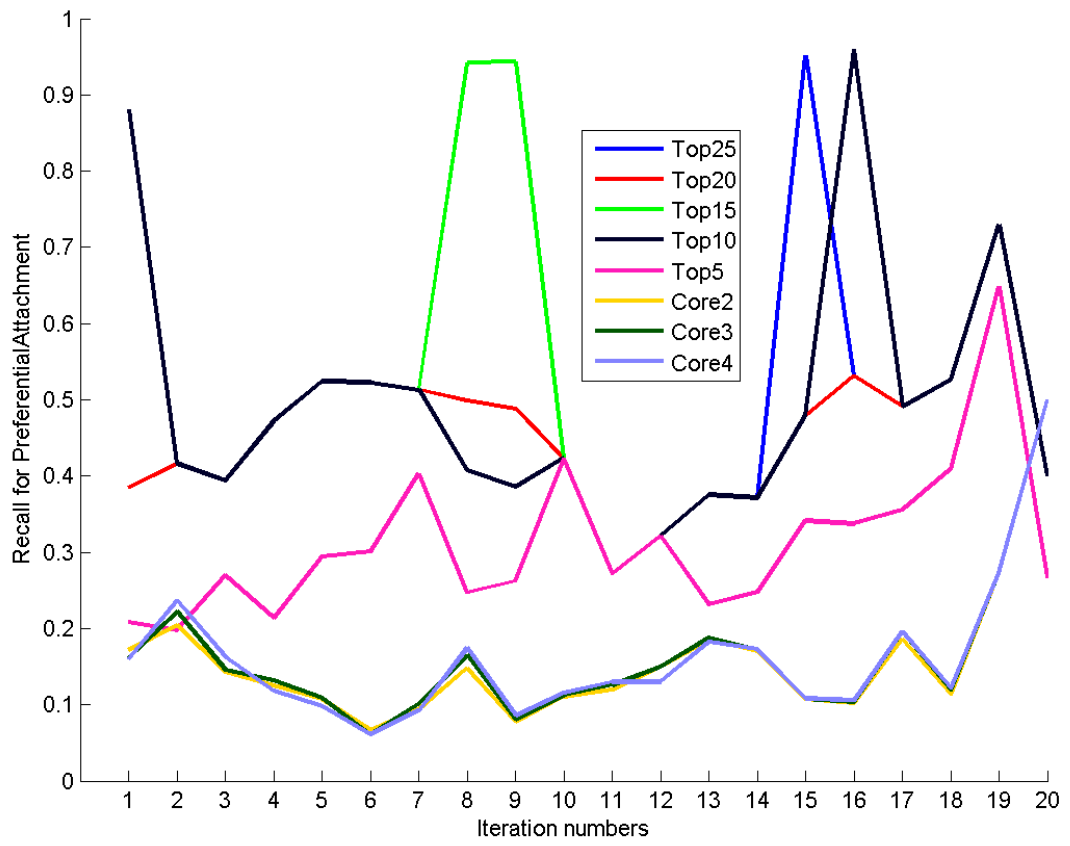


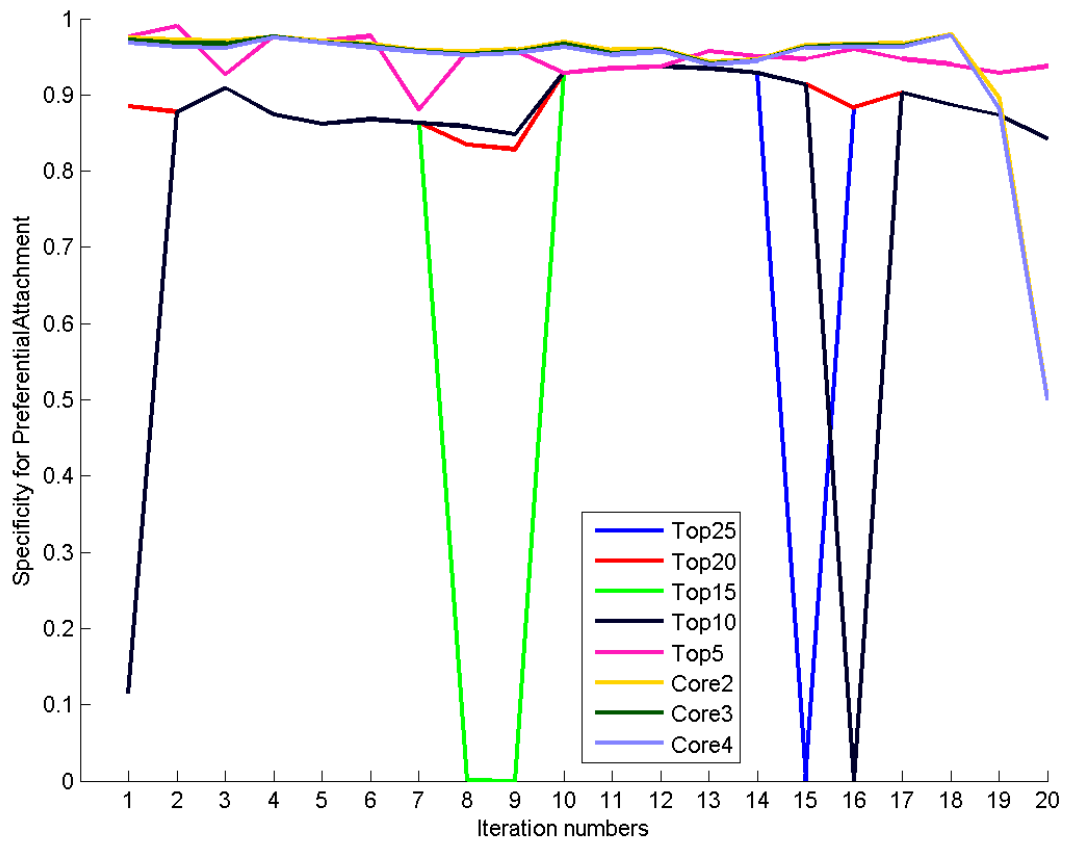


**Preferential Attachment:**

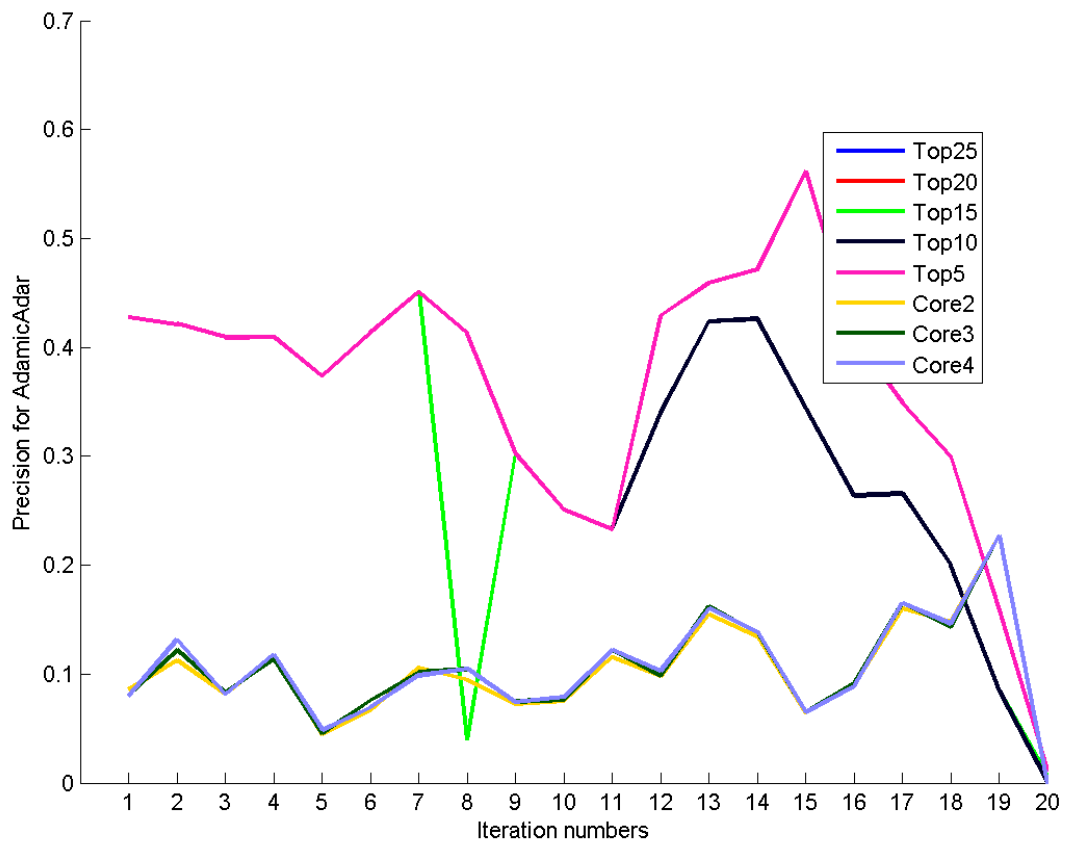


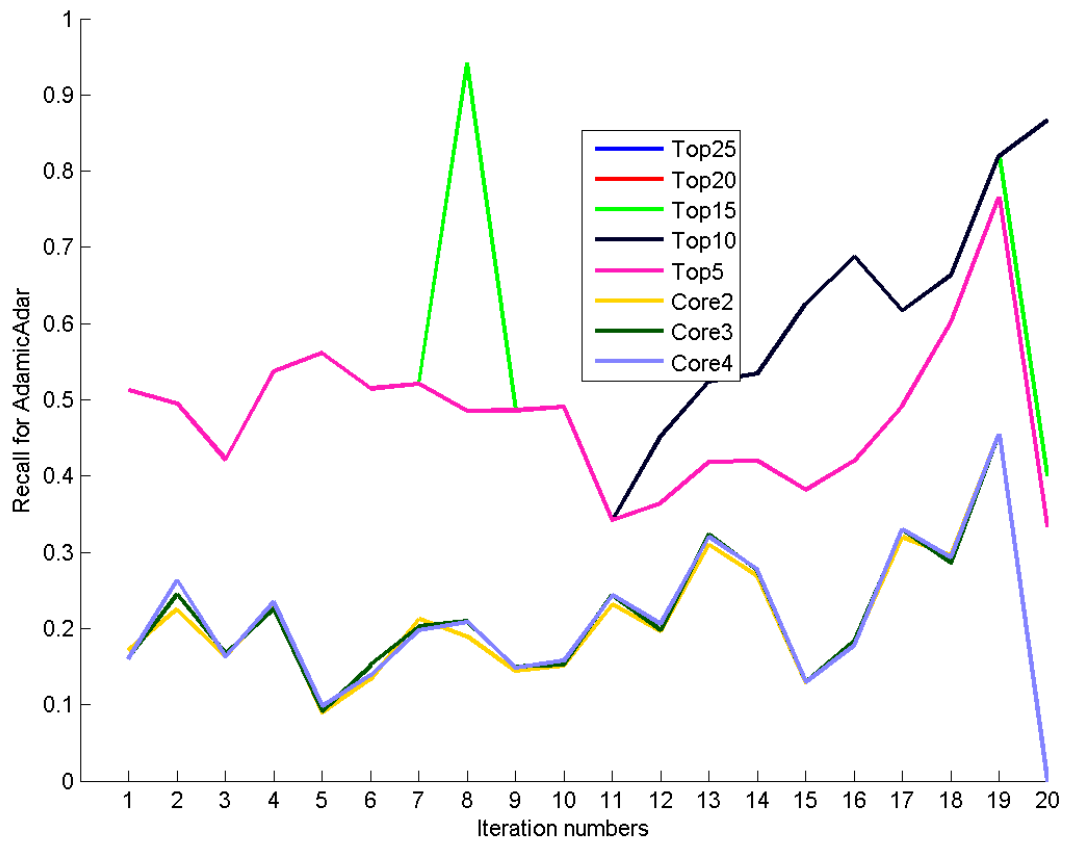


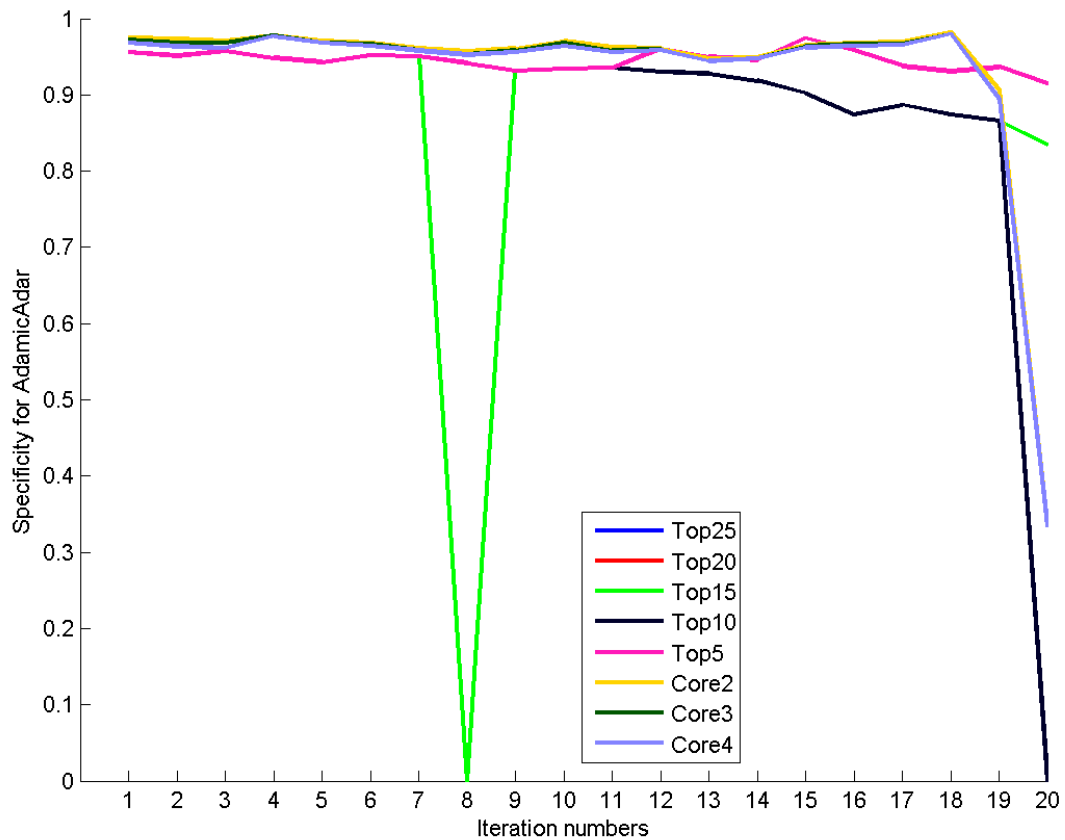




**Adamic Adar:**







#### 4.5.5. Facebook WallPosts dataset - Results of our algorithm with different version of Prior Probability

We also tested the second version of our algorithm which calculates prior probability differently with Facebook WallPosts dataset, and a careful analysis of results proved that the results are similar to that of the Enron dataset.

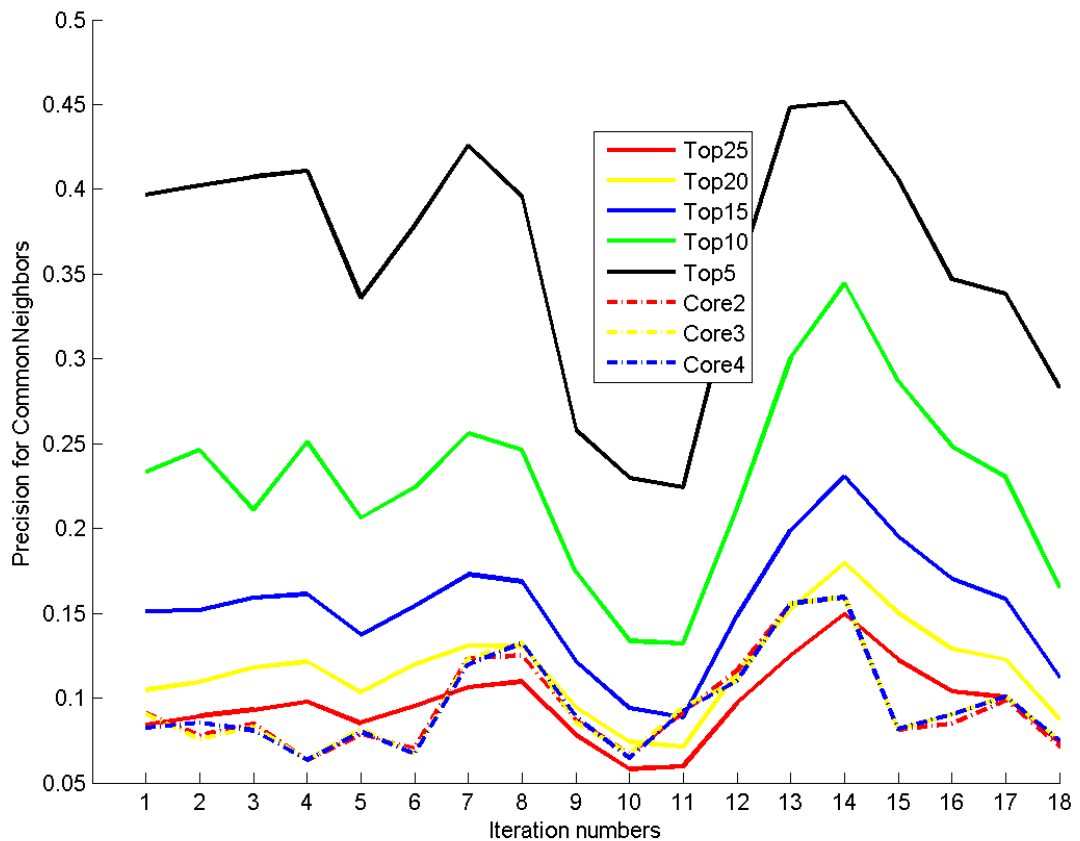
#### 4.5.6. Our Algorithm vs Second variant of Nowell's

Second variant of Nowell's is almost similar to the original variant except that in the second variant the number of edges predicted by the Nowell's was made equal to the number of edges predicted by Top 5% variant of our algorithm.

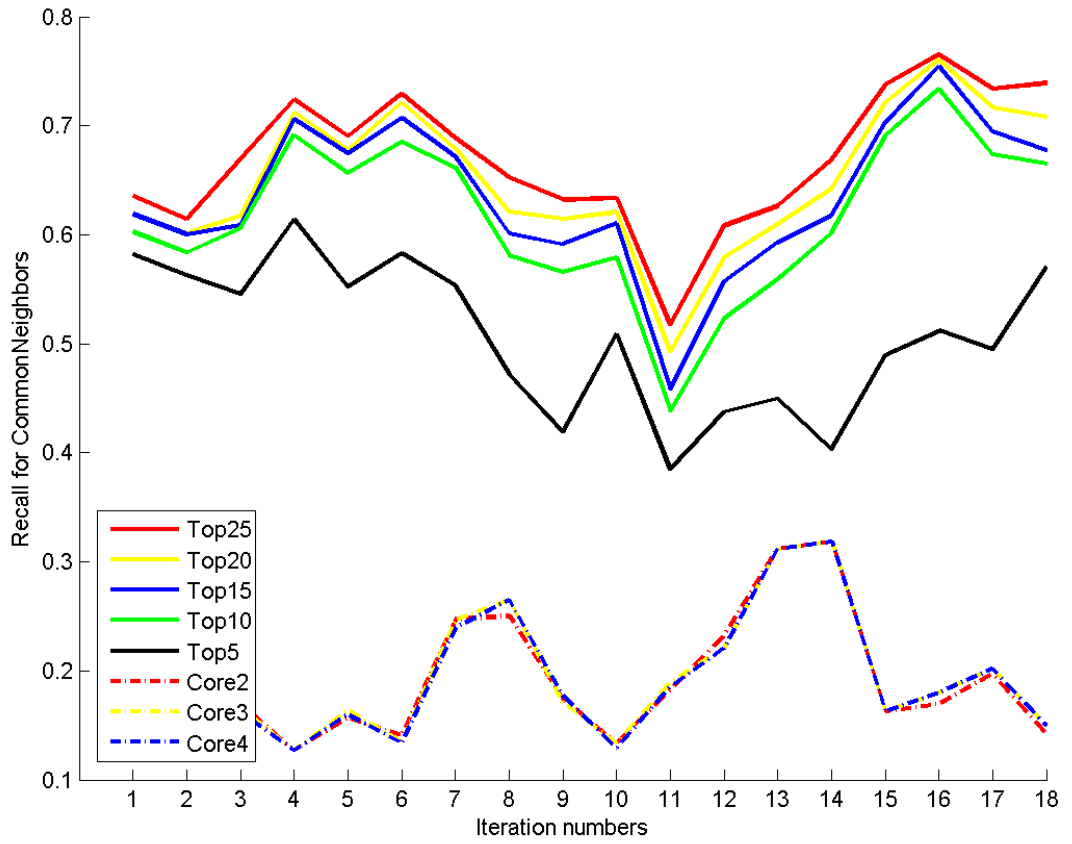
Careful analysis of the results revealed that our algorithm also fares better than the second variant of Nowell's and surprisingly, specificity for the Top 5% fares better than that of Nowell's.

Graphs comparing the results for the feature 'Common Neighbors' is given below. All the remaining three features showed a similar behavior.

**Precision using Common Neighbors:**

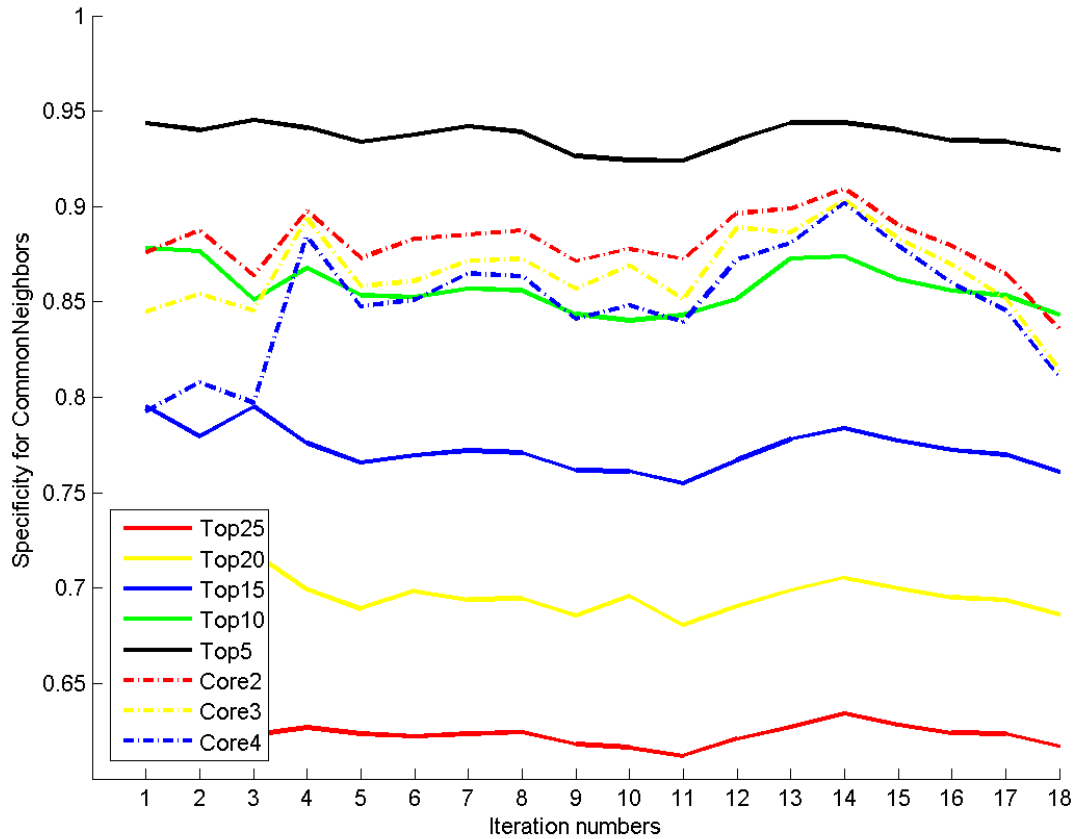


**Recall using Common Neighbors:**





### Specificity using Common Neighbors:



#### 4.5.7. Our Algorithm vs Third variant of Nowell's

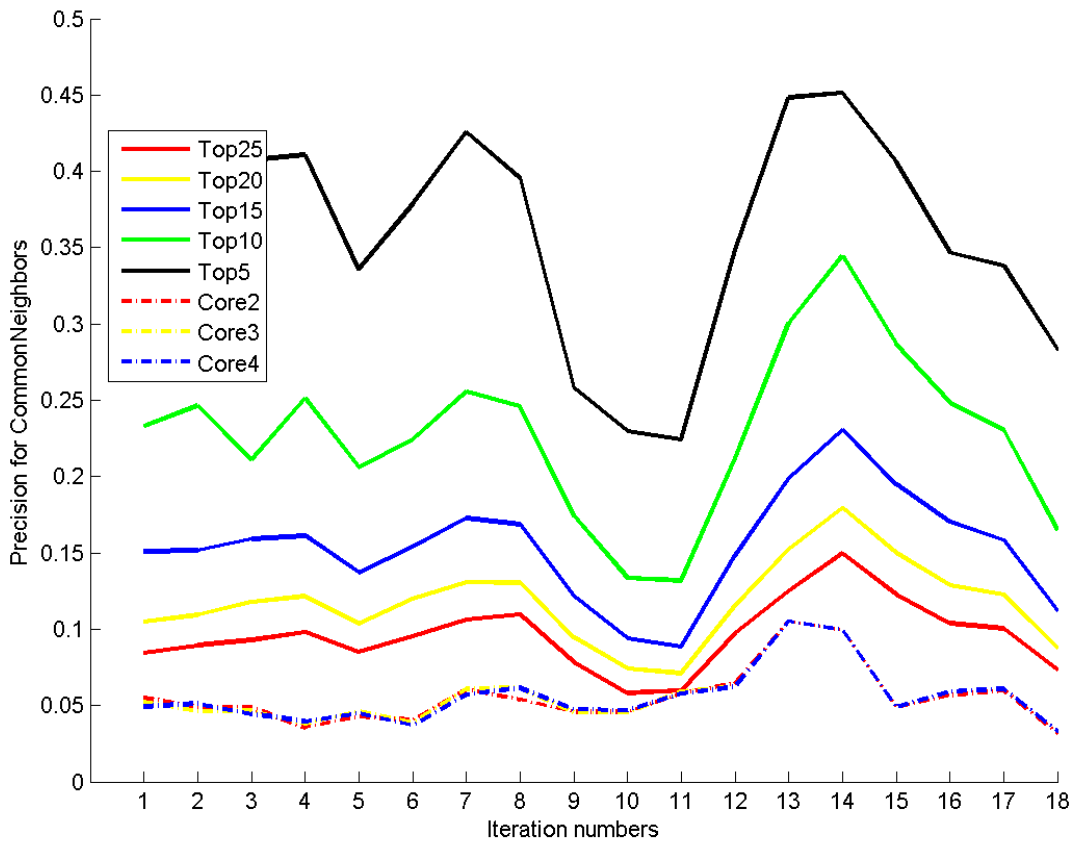
Third variant of Nowell's is almost similar to the original variant except that in the second variant the number of edges predicted by the Nowell's is made equal to the number of edges predicted by Top 5% variant of our algorithm, and also it is allowed to

predict the edges that are present now and may appear again in future unlike the original and the second variant of Nowell's algorithm.

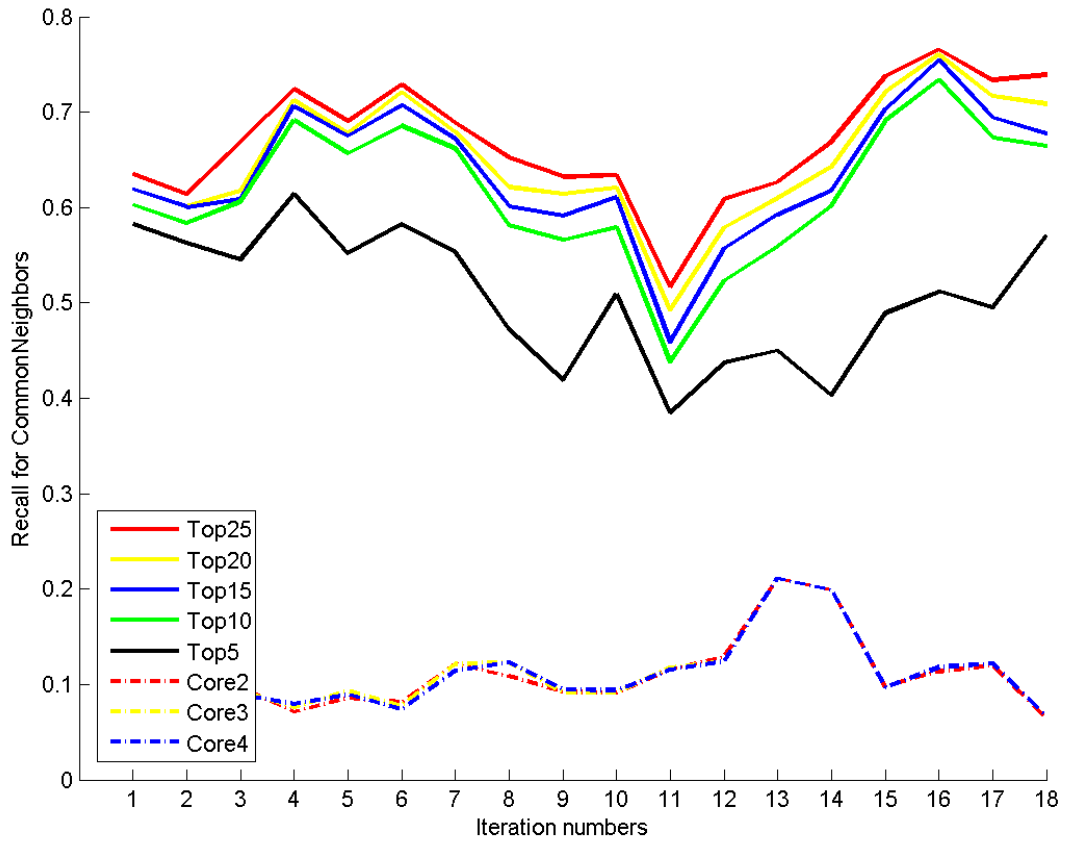
Similar to the second variant results given above, analysis of the results proved that our algorithm also fares better than the third variant of the Nowell's and suprisingly, specificity for the Top 5% fares better than that of Nowell's.

Graphs comparing the results for the feature 'Common Neighbors' is given below. All the remaining three features showed a similar behavior.

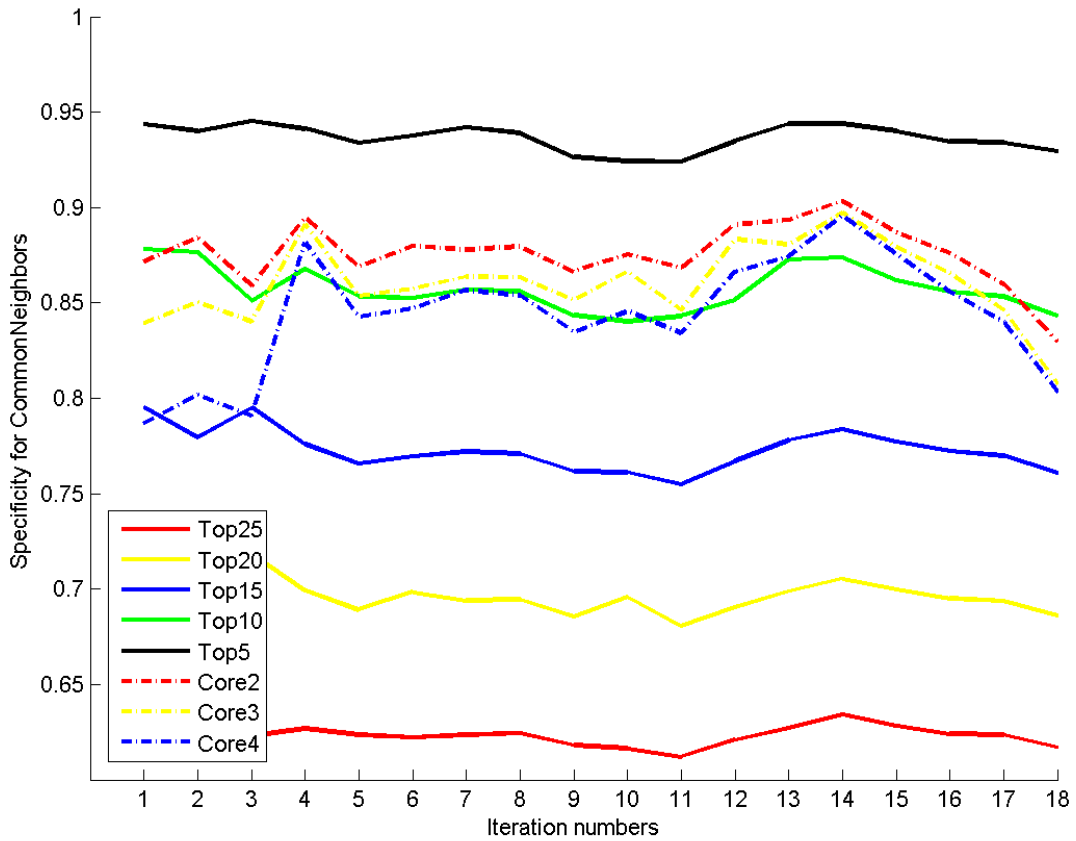
**Precision using Common Neighbors:**



**Recall using Common Neighbors:**



**Specificity using Common Neighbors:**



## Chapter 5

### Conclusion and Future Work

There are many link predictions algorithms which exist today and each one of them serves a specific purpose. For example, one algorithm focuses on predicting what link the user likely to click on the current web page in case of web pages whereas the other one focuses on who is likely to become friends with whom in the future in a social network. While the main aim of all these different approaches remains same, i.e., to predict the future edges with good accuracy, each one caters a specific purpose.

In our work we approached the problem of link prediction differently by making use of the past graph structure along with the current graph structure of an evolving graph. This methodology mainly caters to the specific area of how communication between people change over time.

Our algorithm is tested against different types of datasets and the relevancy measures such as precision, recall, and specificity values are calculated to evaluate the performance of our link prediction algorithm. A detailed analysis of our algorithm and results confirms that:

1. Taking into account the past structure and message flows in the graph, along with the current structure and flows yields performance that is better than that obtained by structural features alone.
2. Bayesian updating of probability for an edge being active works with very good results.
3. Different way of computing Prior probability also makes good Link Prediction with good Precision, Recall, and Specificity

4. How many edges to make predictions for: This is still an issue.

There are some areas of improvement in this work that can be explored in future research.

Given below is a list of future works that can be done based on our work:

1. Future research can focus on improving the Specificity/ability to classify the negatives more accurately
2. Different ways in which the prior probability is computed can be further explored
3. Factors specific to a particular dataset that influences the future edges can be included in the link prediction algorithm
4. We currently calculate the posterior probability of a link to appear in the future based on the current data and prior probability of that edge. For this purpose, we predefined the data period as six months past data, two months evidence data, to predict the next two months test data. We also experimented slightly by changing the evidence and test period to one month. This can be explored further by iteratively learning a Bayesian classifier from the data and then use that in calculating the posterior probabilities.

## References

- [1] David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management (CIKM '03)*. ACM, New York, NY, USA, 556-559.
- [2] Anna Goldenberg and Andrew W. Moore. 2005. Bayes net graphs to understand co-authorship networks?. In *Proceedings of the 3rd international workshop on Link discovery (LinkKDD '05)*. ACM, New York, NY, USA, 1-8.
- [3] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi. 2011. Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11)*. ACM, New York, NY, USA, 1100-1108.
- [4] Zan Huang, Xin Li, and Hsinchun Chen. 2005. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries (JCDL '05)*. ACM, New York, NY, USA, 141-142.
- [5] Jianhan Zhu, Jun Hong, and John G. Hughes. 2002. Using Markov Chains for Link Prediction in Adaptive Web Sites. In *Proceedings of the First International Conference on Computing in an Imperfect World (Soft-Ware 2002)*, David W. Bustard, Weiru Liu, and Roy Sterritt (Eds.). Springer-Verlag, London, UK, UK, 60-73.
- [6] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211-230, July 2003.
- [7] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review Letters* E, 64(025102), 2001.
- [8] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

- [9] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509-512, 15 October 1999.
- [10] Michael Mitzenmacher. A brief history of lognormal and power law distributions. *Internet Mathematics*, 1(2):226-251, 2004.
- [11] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaboration. *Physica A*, 311(3-4):590-614, 2002.
- [12] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. 2009. On the evolution of user interaction in Facebook. In *Proceedings of the 2nd ACM workshop on Online social networks (WOSN '09)*. ACM, New York, NY, USA, 37-42.