SENTIMENT-BASED CLASSIFICATION OF TWEETERS AND UNIVERSITY PROGRAMS

A Thesis

by

BOLUN HUANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Narasimha Reddy
Committee Members,	Guofei Gu
	Srinivas Shakkottai
Head of Department,	Chanan Singh

August 2014

Major Subject: Computer Engineering

ABSTRACT

The rapidly growing World Wide Web (WWW) is no longer a passive information provider. Nowadays, Internet users themselves have become contributors to the WWW. A lot of user generated data, along with non-user-generated data, make our world an informative, however, perhaps over-informed society. The increasing amount of unorganized, disordered, unstructured, or even randomly generated data drove the momentum of big data analysis, aiming to discover and learn the hidden patterns behind the data. In this thesis, in particular, we look at two problems of mining knowledge from data.

In the first project, we are trying to classify "democrats" and "republicans" in Twitter. We first propose a sentiment-based classification model to classify "democrats" and "republicans", with the aim to address the problem that conventional quantitative features, such as tweet_count, follower-to-following ratio, election_tweet_count, cannot reflect the opinion alignment of tweeters. Therefore we utilize sentiment scores over multiple topics as our feature vector in the classification model. We innovatively proposed an automatic topic selection model to learn those distinguishing topics, making the sentiment feature selection domain independent. However, the sentiment-based classification model is not doing much better than non-sentiment model. Given the fact that sentiment-based classification model is not doing well enough, we propose using social relationship graph information to adjust our sentiment vectors. The graph-adjusted sentiment model achieves an accuracy higher than 80 percent in classification. What's more, we deploy a completely graph-based model, Belief Propagation (BP) model on the social graph, which achieves a prediction accuracy higher than 85 percent. We conclude that the effect of social relationship graph

is more important than sentiment of tweets for classifying users into "democrats" and "republicans".

In the second project, we propose an alternative and new way to rank graduate schools using algorithms, instead of using qualitative surveys as U.S. News does. Based on the assumption that "schools tend to hire PhD graduates from better or peer schools" to become their faculty members, we propose deploying link-based ranking algorithms on the "*hiring graph*" among universities. We refine PageRank (PR) algorithm and Hyperlink-induced Topic Search (HITS) Algorithm by taking the edge weight into consideration, as our own way to rank graduate programs. In order to validate our approach, we collect two separate data sets to construct the "*hiring graph*", faculty data in top 50 Computer Science (CS) programs and faculty data in top 50 Mechanical Engineering (ME) programs across the United States. By comparing our new rankings with U.S. News ranking, we discover that some programs are either under-ranked or over-ranked by U.S. News. We also conduct extensive data analysis on our data, revealing a lot of interesting patterns and cases behind the U.S. News ranking. Finally, we conduct sensitivity analysis on each proposed algorithms to see how sensitive they are in response to changes in the "*hiring graph*".

DEDICATION

To my mother,

and my father.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Reddy, and my committee members, Dr. Gu and Dr. Shakkottai, for their continuous guidance and brilliant help during the course of my research.

I would also like to thank my father and my mother for their love and support during my graduate study. Without their love and support, I would never come this far.

I would like to thank Vandana Bachani for her brilliant help at the beginning of my first project.

Thanks also to my friends and colleagues during my stay in Texas A&M University. I would like to thank Srihasha Madala, Xin Wang, Abhishek Jain and Pradipta Bose for their support and help. I would also like to thank my colleagues in Dr. Reddy's research group, Zhiyuan, Allen, Sangwhan, Sheng and Shyam for their great helps in the course of my research.

Finally, I would like to thank my mentor in Flurry Inc, Soup Ranjan, for his great guidance in my internship project.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

1. INTRODUCTION

1.1 Motivation

Data seems to be dominating our world now. Most of the data is randomly generated by users. Much of the data is disordered, unstructured and unorganized. However, patterns of knowledge widely exist behind data. In the recent years, scientists and engineers have been paying lots of effort on effectively managing data and revealing the hidden patterns behind the data. In order to approach this goal, we extensively utilized machine learning, data mining, sentiment analysis, pattern recognition and knowledge discovery techniques and algorithms in order to reveal the hidden value behind the data. Particularly, we look at two problems on mining knowledge from data.

1.2 Sentiment-based User Classification in Twitter

The first problem we are interested in is to classify Twitter users according to their political point of view. More specifically, by working on a labelled data set obtained from Twitter Streaming data [1] crawled during the period of 2012 US Presidential Election, we are trying to separate politically active tweeters as *republican* and *democrat* according to their *political leanings*. The motivation is that we believe sentiment related feature set could perform better than non-sentiment features set in this particular problem. In order to reveal the subjectivity within the Election alignment, we propose using sentiment of topics, based on the hypothesis that different classes have different opinions on many topics. As we know, the performance of many classification models depends on the choice of features vectors. In this project, innovatively, we want to identify distinguishing features automatically based on statistical distribution of sentiments on a particular topic. We propose a scheme to examine and rank the degree of polarity of topics, aiming to find those most distinguishing topics. In addition, in order to improve the sentiment-based classification model, we took advantage of social relationship graph to adjust the sentiment feature vectors. Finally, we deploy a completely graph-based model—Belief Propagation (BP) model, given the observation that "tweeters with same political association tend to follow or to be followed by each other" from the social graph.

1.3 Algorithmic University Program Ranking

In the second project we are trying to develop our own way to rank graduate programs in a particular field. U.S. News ranks each graduate program across the USA based on both input from program deans and some other statistical indicators. We proposed an alternative and simple way to rank graduate programs using algorithms on what we call "*hiring graph*". Our motivation is based on a simple assumption that "universities tend to hire PhD graduate students from better or peer universities". In order to validate our approach, we collected two sample datasets: faculty data from top 50 Computer Science (CS) programs in the USA listed in the U.S. News, and faculty data from top 50 Mechanical Engineering (ME) programs in the USA listed in the U.S. News. We only collect two pieces of information for each faculty: 1) Which university did they graduate from; 2) In which year did they graduate from their graduate school. Once we have the data, we run experiments to test HITS-based algorithms, PR-based algorithms on the "*hiring graph*" and let the algorithms learn the rankings of the program automatically. Besides, by comparing our rankings and the U.S. News ranking, we observe a lot of interesting patterns and facts behind the U.S. News ranking, concluding that U.S. News might either over-rank or under-rank some of the programs. Finally, our sensitivity analysis shows how sensitive our algorithms are in terms of changes in the "*hiring graph*".

2. SENTIMENT-BASED USER CLASSIFICATION IN TWITTER

2.1 Introduction

Twitter, a popular social networking platform, has been attracting lots of research attention in the recent years. Twitter Streaming API [1] provides one percent of its entire data ("firehose") to the public for free, making data mining research on online social media possible. Twitter REST API [2] provides interfaces for collecting more personalized information of tweeters, making in-depth analysis possible. A typical example of Twitter analysis is that much research has been done on fighting against Twitter spammers. Amleshwaram et al in [3] proposed using a number of preselected quantitative features to separate twitter spammers from benign users, achieving a classification accuracy as high as 90 percent. In [4] Yang et al proposed a relationship graph based inference model on Twitter, also trying to separate spammers from ordinary Twitter accounts. Another research topic on Twitter is analyzing and revealing political election trends and results, often with the technique called "sentiment analysis" [5]. For example, in [6], Metaxas and Mustafaraj discussed how Twitter affects the Election progress and in what degree the Twitter trend reflects the offline Election result. Wang et al, in [7], developed a real-time system to analyze the sentiment characteristic of tweets during the 2012 US Presidential Election period, using Naive Bayes model on unigram features to predict the tweets' favorability. Choy et al, in [8], conducted statistical analysis on tweets during Singapore Presidential Election and compared their prediction based on their model to the actual result. Tumasjan et al, in [9], proposed a sentiment analysis model to predict German Presidential Election.

The previous election tweets analyses mainly focused on using statistical and demographical analysis to "predict" or "conjecture" the result of election. In this project, we would like to answer the following question: who is supporting which party or candidate. Unlike [6] and [7], we pick a different angle of view: can we classify followers of different political parties based on their tweets? We propose to employ sentiment analysis to answer this question.

2.2 Data Set

2.2.1 Data Description

2.2.1.1 Labelled Users

In our data set we have 393 labelled tweeters in total. All these tweeters are relatively active campaigners, either supporting *Democrat* or *Republican* candidates during the 2012 US Presidential Election. All these users are still active after the 2012 US Presidential Election. We collected data using Twitter Streamming API [1] during the 2012 US Presidential Election period, from October 01, 2012 to November 06, 2012.

Figure 2.1 shows the procedures employed in converting Twitter Streaming Data to our set of labelled users. We will explain the entire procedures step by step.

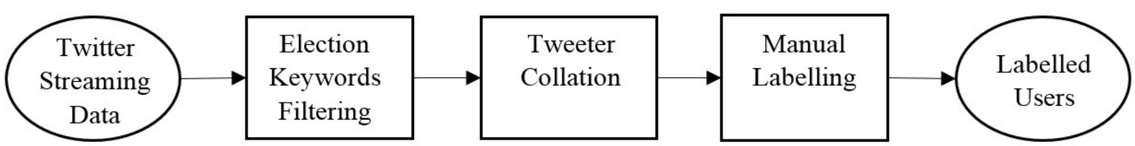


Figure 2.1: Data Pre-processing Procedures

The first step is *Election Keyword Filtering*. In this step, we filtered out the tweets related to election according to several keywords that we have defined. These keywords are listed in Table 2.1. As a result, 1,745,985 *election related tweets* are collected. After we have these tweets, in the second step, we collated these tweets according to its poster, so as to obtain a list of tweeters, who are active during the campaign. Hence we have a list of tweeters ordered in terms of tweet count that have at least one *election related tweet*. In addition, we trimmed the tail of the list to maintain a list of *high volume tweeters* who have at least 3 *election related tweets*. Step two gave us 78,334 *high volume tweeters* and the most active tweeter posted 173 *election related tweets*. Table 2.2 shows a sample of the *high volume tweeters*.

The last step is the *labelling* task. We manually pick some of the tweeters to label them as either "*democrat*" or "*republican*". The methodology to determine the label of each collected tweeter is to manually look at their *election related tweets*. If the tweeter either supports Romney and Republicans exclusively, disputes Obama and Democrats exclusively, or combine these two sentiments together, we label it as "republican"; Similarly, if the tweeter either support Obama and Democrats exclusively, disputes Romney and Republicans exclusively, or combine these two sentiments together, we label it as "*democrat*". Table 2.3 gives us four sample tweets from *IBumbybee*, which is labelled as "*republican*" in our data set. We can see that this person is clearly a promoter for Mitt Romney and a disputant for Barack Obama. Table 2.4 summarizes five sample tweets from *MsNatTurner*, which is labelled as

Table 2.1: Filtering Keywords

Keywords	
election	elections
obama	romney
republican	democrat
paul	ryan
biden	mitt
vote	president

Table 2.2: High Volume Tweeters

No.	Screen_Name	Election Related Tweets#
1	mohamedaldy	173
2	_peace_full_	169
3	CelebVoler	152
4	TeamMikeMorris	152
5	BidacudaVote	140
6	skew11	129
...		
78332	riaaxo	3
78333	iJUSTify_tweets	3
78334	HighSierraMan	3

"democrat" in our data set. We can see that *MsNatTurner* is in favor of Obama because *MsNatTurner* is appealing to re-elect Obama in her tweets.

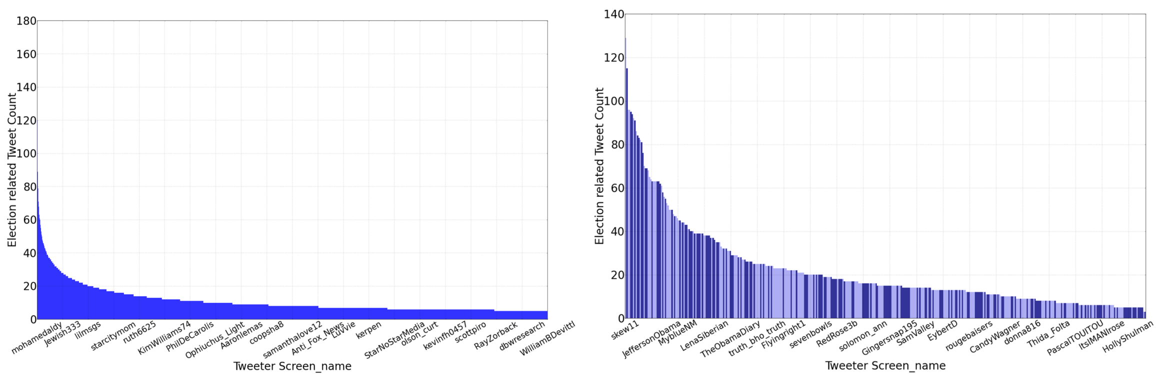
Table 2.3: Sample Tweets of IBumbybee ("Republican")

After three years of Obama, we are hopeless + changeless + we need Mitt Romney to bring America back!C̃hris Christie ...
RT @ CraigBowden2020: We must get Obama out! Support Mitt Romney! # tcot # EvictTheIncumbent # nobama # lnyhbt # MV4F # election2012...
RT @ OrwellForce: Reminder: Obama's plan to close the deficit by raising tax rates CANNOT work `http://t.co/W2RSewaL`
RT @ MsMelanie: Mitt Romney does not have to prove anything. He has already been successful in all areas of his personal and business...

In order not to introduce bias by only considering those extremely active campaigning tweeters, we also labelled those tweeters with relatively fewer tweets. We maintained a user distribution similar to the overall distribution so as to have a representative data set, as Figure 2.2 shows. In Figure 2.2b, the darker bars represent "*democrats*" and lighter bars represent "*republicans*". As we can see, we also have made sure that the "*republicans*" and "*democrats*" are evenly and fairly dis-

Table 2.4: Sample Tweets of MsNatTurner ("Democrat")

RT @ Mr_Maz: I have decreed that anyone who votes for @ mittromney is an textitidiot and douche!
RT @ JoeBiden: 3 reasons why President Obama should be re-elected: ***http://t.co/KtsjI5A3***
San Jose Mercury News: Re-elect President Obama ***http://t.co/q2JqJmLE*** *# Obama2012 # 4jobs # 2futures # p2 # tcot*
RT @ azmoderate: @ MittRomney talking about "rubbish" on a football field *and cleaning up a "lane" and compares it to hurricane clean up? ...*
@ sunshineejc: Five Practical Reasons Not To Vote Republican | Common Dreams ***http://t.co/Ibl5Aotr***



(a) Overall Tweeters

(b) 393 Labelled Tweeters

Figure 2.2: Distribution of Tweet Volume

tributed. Finally, among our 393 labelled tweeters, there are 212 "*republicans*" and 181 "*democrats*".

2.2.1.2 Recent Tweets

After having the labelled users as our ground truth, however, we did not use the Streaming Twitter data we have during the Presidential Election Period to do our experiments. There are several reasons for this. First of all, Twitter Streamming API only provides one-percent random sample of the entire data. Secondly, in the Streaming Twitter data we have collected, as we can see in Table 2.2, the volumes of some tweeters is relatively small that they have little sentiment information in their *election related tweets*. Thirdly, since we only care about the sentiment in tweets, we don't care about whether the tweets are from Presidential Election Period or not as long as there are sentiment within the tweets. Based on the above reasons, we collected a new data set to deploy our sentiment analysis model in September 2013. We crawled the recent 200 tweets for the 393 labelled tweeters using Twitter REST API [2] so that we made sure that every user in our data set has a reasonable volume of tweets—200 recent tweets. Hence, in total we have 78,600 tweets and all the following sentiment analysis model is based on these 78,600 tweets.

2.2.2 Social Relationship Graph

Another data we have collected is the friends and followers data of our labelled users, which is collected along side with the recent tweet data in September 2013. We used *GET friends/ids* [10] and *GET followers/ids* [11] in Twitter API to crawl the friends and followers of the 393 labelled users. Although the friends and followers data must include some other tweeters outside our labelled data set, we do not consider these social links and we project the social graph exclusively on the 393 users. Therefore we have created a social relationship graph with 393 nodes and 7,433 edges. Considering the labelled users are randomly chosen, the density of our subtracted graph is pretty good and sufficient enough to investigate into the social graph among these politically active tweeters.

2.3 Our Approach

In this section, we introduce our approach to analyzing the collected data. The first major focus is to utilize sentiment analysis model to classify the users. We propose an automatic way to discover what topics are the most distinguishing, in other words, the topics "separating" tweeters in terms of their political views. Another major focus is to take advantage of social relationship graph information to adjust or help improve the classification model to separate politically active tweeters.

2.3.1 Sentiment Analysis Model

Sentiment Analysis is a technique that tries to extract the opinion, sentiments, attitudes and emotions from written language. It is sometimes referred as "Opinion Mining". *Sentiment Analysis* relies on "Natural Language Processing"(NLP), which is out of the scope of this project. The scope of our sentiment analysis is on computational treatment of sentiment scores given by the NLP in order to solve our classification problem.

A basic task of sentiment analysis is to discover the polarity of a given text in a corpus. For example, Park et al in [12] utilized sentiment analysis to classify tweets to be positive, negative and neutral. Researchers in [7], [8] and [9] all focusing on revealing the political subjectivity in the granularity of tweets posted during the election period.

In our project, we propose utilizing sentiment analysis to determine the political alignment of a particular tweeter with respect to some topics or trending keywords. In our data set, each labelled tweeter is an author of 200 tweets and our data set could be considered as a *Corpus* of 393 *Documents*.

In our problem, we have two target classes, which are "*democrat*" and "*republican*". Let's say the two target classes are C_1 and C_2 respectively. The underlying assumption of these two classes is that they have different opinions on many topics. Let our *corpus* (labelled data set) S consists of N documents, $D_0, D_1, D_2, \ldots, D_{N-1}$. Assuming we have M topics, $T_0, T_1, \ldots, T_{M-1}$, discovered from our corpus, each topic T_j must have a probabilistic distribution $H(T_j)$ against the two classes:

$$H\left(T_j\right) = \begin{cases} H_{C_1}\left(T_j\right), & \text{if } T_j \text{ in } C_1 \\ H_{C_2}\left(T_j\right), & \text{if } T_j \text{ in } C_2. \end{cases} \tag{2.1}$$

where $H_{C_i}\left(T_j\right)$ can be expressed as:

$$H_{C_i}\left(T_j\right) = \begin{cases} P_{Pos}, & \text{when } T_j \text{ is Positive in } C_i \\ P_{Neu}, & \text{when } T_j \text{ is Neutral in } C_i \\ P_{Neg}, & \text{when } T_j \text{ is Negative in } C_i \end{cases} \tag{2.2}$$

where P_{Pos} is the probability that T_j has a positive sentiment, similarly for P_{Neu} and P_{Neg}. We note that the sum of P_{Pos}, P_{Neu} and P_{Neg} is equal to 1.

Usually the distribution $D_{C_1}\left(T_j\right)$ and $D_{C_2}\left(T_j\right)$ are different, based on the assumption that "Different classes have different opinions on many topics". The larger the difference is, the better the T_j can distinguish C_1 and C_2. Based on the above model, we propose a method to discover the most distinguishing topics out of the corpus in an "automatic" way.

Moreover, in order to correlate the sentiment of topics to the sentiment of tweeters, we construct the following model. Saying each tweeter as a document D, for any document $D_i \in S$, regardless of what class D_i belongs to, we have a projection of the sentiment of all topics T_j on D_i. Letting $P_{D_i}\left(T_j\right)$ represent the sentiment of T_j projected on document D_i, for every D_i, we have a unique vector for each tweeter

$$V\left(D_i\right) = \left(P_{D_i}\left(T_0\right), P_{D_i}\left(T_1\right), \ldots, P_{D_i}\left(T_{M-2}\right), P_{D_i}\left(T_{M-1}\right)\right) \tag{2.3}$$

where M is the total number of topics. After having the sentiment vectors of tweeters, we could feed the vectors into a learning classifier to train the model or into a trained classifier to predict the class of D_i.

In order to construct our sentiment analysis model, we utilized three public sentiment analysis tools. The first one is AlchemyAPI [13]. AlchemyAPI is a powerful NLP tool. Given a text in natural language, Alchemy returns a sentiment score in the range of $[-1, 1]$, indicating whether the text exposes negative/neutral/positive sentiment. The smaller the score is, the more negative the text is; the larger the score is, the more positive the text is; score of 0 indicates a neutral sentiment of the text.

In addition to Alchemy API, we also take advantage of two word sets, Sentiword Net [14] and Sentiment Lexicon dataset [15]. These two word sets are pools of words with sentiments. In Section 2.3.2 we will use these tools to help us with topic selection.

For convenience, we denote the sentiment score returned by Alchemy API, SentiwordNet and Sentiment Lexicon as $S_{Alchemy}$, $S_{Sentiword}$ and $S_{Lexicon}$ respectively. By using these three sentiment analysis tools, we generate the sentiment scores for every single tweet. Thus, each tweet i has three sentiment scores: $S_{Alchemy}(i)$, $S_{Sentiword}(i)$ and $S_{Lexicon}(i)$.

2.3.2 Automatic Topic Discovery

In tweets, every single word could be considered as a "*keyword*" or "*topic*". However, a lot of noise exists in tweets. For example, "haha" could be a word while it could not be considered a topic usually. For another example, "the" is a word while it is not a topic. In order to not introduce avoidable noise into our topic candidates, we utilize what is called "hashtag" in tweets. "Hashtag" is a word in tweets starting with a number sign #, usually with the goal to emphasize a particular event, people, hot topic, group and so on. By only considering *hashtags* instead of every single word, we narrow down and simplify our topic domain greatly. Within our corpus, we compute two metrics for each hashtag: *TagCount* and *UserCount*. *TagCount* is simply the frequency of appearance of the hashtag, and *UserCount* is the number of distinct users who have tweets containing this word. In other words, *TagCount* reflects the popularity of the hashtag and *UserCount* reflects the range of coverage of the hashtag.

As a result, 8,446 hashtags are discovered, which is a large number. Among all the hashtags, # *tcot* has the largest *TagCount* as 3,669, saying that it appears 3,669 times in our corpus. The first filtering step would be to select those hashtags with greater popularity as well as sufficiently large user coverage. We filter out those with *TagCount* less than 37, which is less than 1 percent of the largest *TagCount*. Based on the previous filtering, then we filter out those with *UserCount* less than 38 (less than 10 percent of the labelled users). We obtain 78 hashtags, or say, "topic candidates", with reasonable popularity and sufficient user coverage. Table 2.5 shows all the 78 hashtags with number sign # removed.

Since each tweet has an Alchemy sentiment score $S_{Alchemy}(i)$, we define: As long as hashtag h appears in tweet i, we denotes that $S_h(i) = S_{Alchemy}(i)$. Since a hashtag might appear in multiple tweets, say L_h tweets, thus the sentiment of a each hashtag would be a set of L_h sentiment scores of h as:

$$S_h = \{S_{Alchemy}(i))\} \tag{2.4}$$

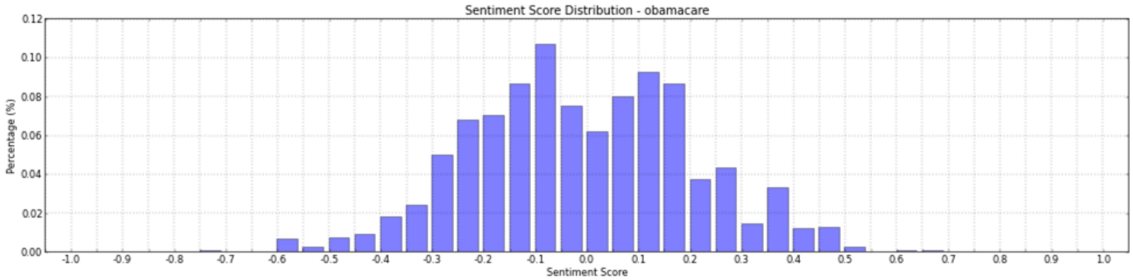
where $\{i\}$ denotes the set of tweets containing topic h.

Thus, each hashtag has a distribution of sentiment scores. Figure 2.3 shows 4 examples of distribution of sentiment scores returned by Alchemy. As we can see, some topics like "obamacare" and "gop" have a bipolar shape of distribution and wide range; some topics like "liberal" do not have an obvious bipolar shape and the range of sentiment scores is narrower. Based on these observations, we propose using the *Max-Min_Diff* characteristics to rank how distinguishing the topic is. For hashtag h, $\textit{Max-Min\_Diff}\,(h) = Max\,(S_h) - Min\,(S_h)$, which is the absolute difference between the maximum sentiment score of h and minimum sentiment score of h. We believe that larger the *Max-Min_Diff* is, the more distinguishing h is. Among $S_{Sentiword}$, $S_{Alchemy}$ and $S_{Lexicon}$, $S_{Sentiword}$ gives wider range of sentiment scores for hashtags. Hence, we use $S_{Sentiword}$ to rank the hashtags. After ranking the hashtags using the *Max-Min_Diff* of $S_{Sentiword}\,(i)$, we have an ordered list of those hashtags. Table 2.6 shows part of the resulting ranking. The Top Set, *Sentiment_Top5*, from No. 1 to No. 5, contains *tcot*, *obama*, *syria*, *gop* and *rednationrising*. The Middle Set, *Sentiment_Middle5*, from No. 40 to No. 44, contains *nsa*, *aca*, *guncontrol*, *iran* and *profile*. The Last Set, *Sentiment_Last5*, from No. 74 to No. 78, contains *ohio*, *liberal*, *nyc*, *forward* and *climate*. In Table 2.6, *Diff* means *Max-Min_Diff*, and *Neg/Neu/Pos* indicates the number of times of that hashtag to be determined as Negative, Neutral and Positive respectively.

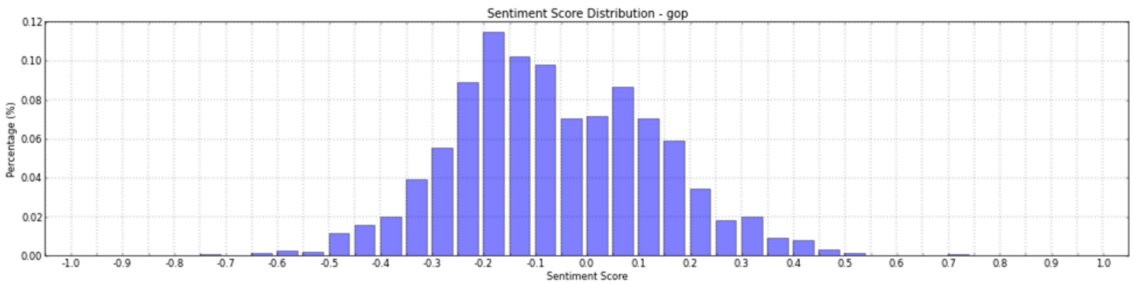
Our hypothesis is that topics that have more divergent scores of sentiment will lead to better classification of tweeters into two classes of "democrat" and "republican". If this hypothesis is valid, The top ranking topics *Sentiment_Top5* should perform better in classifying our labelled tweeters than *Sentiment_Middle5* and *Sentiment_Last5*. We will show the comparative results among these feature sets in Section 2.5.1.

Table 2.5: Top 78 HashTags

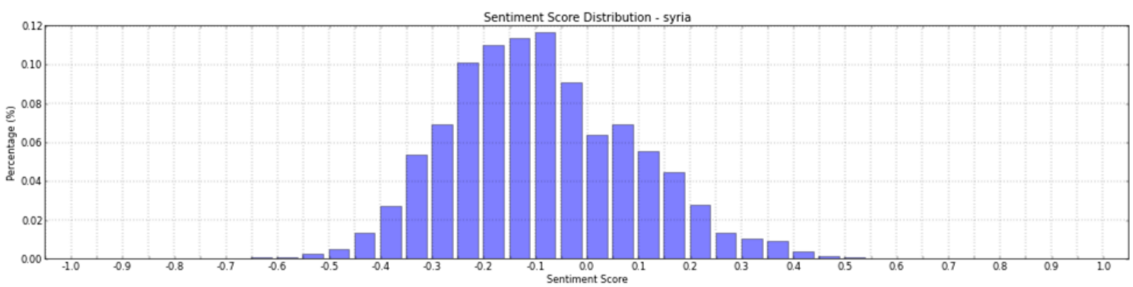
Hashtag	UserCount	TagCount	Hashtag	UserCount	TagCount
tcot	266	3669	fb	42	71
syria	281	1620	newtown	40	71
p2	188	1578	zimmerman	103	69
benghazi	237	976	god	252	65
obama	366	915	tyranny	57	65
teaparty	138	775	christian	124	64
uniteblue	103	717	guncontrol	44	64
gop	288	633	nyc	91	63
obamacare	304	600	catholic	44	63
tlot	96	583	politics	136	62
pjnet	86	486	iraq	160	61
ff	58	285	aca	47	61
lnyhbt	76	273	america	335	59
tgdn	84	266	economy	163	59
israel	124	262	families	106	59
irs	164	244	bible	57	59
ccot	85	213	democrats	218	56
defundobamacare	66	200	liberal	169	56
dontfundit	47	170	jobs	210	55
johnmccainismoreuselessthan	38	150	trayvon	108	54
nsa	144	140	religion	67	50
nra	108	136	women	247	49
news	322	133	maddow	49	49
rednationrising	38	121	war	310	48
whatobamacaremeanstome	46	109	fail	107	47
forward	121	99	dems	171	45
faith	75	98	potus	157	45
egypt	143	97	russia	139	44
immigration	133	96	healthcare	123	44
msnbc	114	96	romney	105	44
video	291	88	chicago	107	43
msm	92	88	military	257	42
cnn	148	87	libertarian	41	42
iran	102	85	kerry	206	41
republicans	248	80	assad	177	40
lgbt	54	78	ohio	104	39
prolife	66	77	abortion	144	38
vote	308	75	climate	86	37
congress	305	74	sandy	60	37



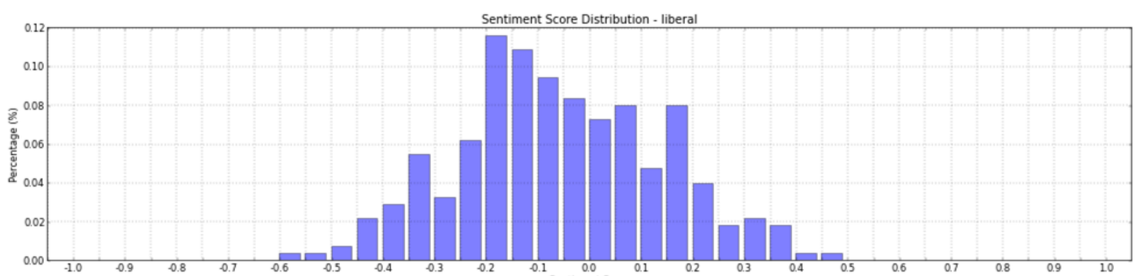
(a) "obamacare"



(b) "gop"



(c) "syria"



(d) "liberal"

Figure 2.3: Sentiment Score Distribution of HashTags

Table 2.6: HashTag Statistics

			SentiwordNet3.0				Alchemy API				Sentiment Lexicon			
		Hashtag	Neg/Neu/Pos	Min	Max	Diff	Neg/Neu/Pos	Min	Max	Diff	Neg/Neu/Pos	Min	Max	Diff
Top Set	1	tcot	1059/1522/1088	-0.75	0.75	1.5	1622/1018/1029	-0.702	0.645	1.347	1808/581/1280	-0.75	0.75	1.5
	2	obama	270/334/311	-0.75	0.75	1.5	393/250/272	-0.682	0.553	1.235	441/114/360	-0.75	0.75	1.5
	3	syria	577/602/441	-0.75	0.75	1.5	842/355/423	-0.673	0.495	1.168	917/177/526	-0.75	0.75	1.5
	4	gop	182/259/192	-0.75	0.75	1.5	300/170/163	-0.603	0.492	1.095	304/133/196	-0.75	0.75	1.5
	5	rednationrising	37/48/36	-0.75	0.75	1.5	48/37/36	-0.441	0.507	0.948	66/19/36	-0.75	0.75	1.5
							...							
Middle Set	40	nsa	62/40/38	-0.5	0.438	0.938	76/35/29	-0.545	0.495	1.04	87/17/36	-0.545	0.495	1.04
	41	aca	13/31/17	-0.438	0.5	0.938	24/14/23	-0.424	0.488	0.912	24/11/26	-0.438	0.5	0.938
	42	guncontrol	13/29/22	-0.286	0.625	0.911	23/23/18	-0.478	0.495	0.973	26/16/22	-0.478	0.625	1.103
	43	iran	38/28/19	-0.438	0.438	0.876	55/18/12	-0.569	0.42	0.989	56/8/21	-0.569	0.438	1.007
	44	prolife	13/41/23	-0.5	0.375	0.875	32/23/22	-0.422	0.435	0.857	34/16/27	-0.5	0.435	0.935
							...							
Last Set	74	ohio	9/18/12	-0.25	0.312	0.562	12/14/13	-0.316	0.47	0.786	13/8/18	-0.316	0.47	0.786
	75	liberal	43/3/10	-0.344	0.208	0.552	31/7/18	-0.417	0.354	0.771	41/0/15	-0.417	0.354	0.771
	76	nyc	29/15/19	-0.281	0.25	0.531	20/25/18	-0.673	0.302	0.975	29/7/27	-0.673	0.302	0.975
	77	forward	30/41/28	-0.188	0.312	0.5	43/23/33	-0.404	0.477	0.881	50/8/41	-0.404	0.477	0.881
	78	climate	5/15/17	-0.208	0.25	0.458	12/12/13	-0.267	0.307	0.574	12/3/22	-0.267	0.307	0.574

2.3.3 Decision Tree Classification

We use Decision Tree learning algorithm as our classifier. The idea behind decision tree learning is to pick attributes that better separate positive and negative cases. Decision Tree Learning algorithm constructs a Decision Tree on the training data. Usually it is implemented in an iterative manner. In each iteration, the best match attribute, in our case, the "topic", is chosen to deploy a "split" on the data so as we could obtain the maximum *Information Gain*. The same procedure is applied on the sub-branches of the tree until every example is classified into a branch of the entire Decision Tree. The key principle of Decision Tree learning is to use Shannon's information theory to choose the attribute that gives the maximum *Information Gain*, which is defined as:

$$Gain\left(E, A\right) = Entropy\left(E\right) - \sum_{v \in Values(A)} \frac{|E_v|}{|E|} Entropy\left(E_v\right) \tag{2.5}$$

where E is set of examples, A is a single attribute and E_v is the set of examples where attribute $A = v$.

Based on Decision Tree Classifier, we compare two different set of attributes, or "features" in our experiment. They are non-sentiment features described in Section 2.3.3.1, and sentiment features described in Section 2.3.3.2.

2.3.3.1 Non-sentiment Features

In our "*non-sentiment classification model*", non-sentiment features is chosen directly from users' profile or computed from factual data of the user. To this end we extracted three categories of non-sentiment features, totally twelve. The first category is *factual profile features* captured directly from users' twitter profile, including *friends count*, *tweets count*, *favorites count*, *followers count* and *listed count*.

They are combined to reflect the uniqueness of each individual tweeter. The second category is *behavioural features*, consisting of *interval variance*, *retweet ratio*, *url ratio*, *unique mention* and *unique url*. Compared with the first category, *behavioural features* dig deeper into the behavioural characteristics of the tweeter. The third category is *contextual features*, including *polarity* and *tweet similarity*. We will briefly explain these features one by one.

Friends count ($FriCnt$): *FriCnt* is the total number of *friends* of the user.

Tweets count ($TwtCnt$): *TwtCnt* is the total number of *tweets* posted by the user.

Favorites count ($FavCnt$): *favorite tweets* are those tweets which the user "likes", "endorses" or gives a "thumb-up" to. *FavCnt* is the number of *favourite tweets* obtained from the user's profile.

Followers count ($FolCnt$): *FolCnt* is the number of *followers* that the user has.

Lists count ($LstCnt$): *lists* in Twitter are the "groups" or "memberships" that the user is in. *LstCnt* is the number of *lists* in the user's profile.

Interval variance ($IntVar$): *IntVar* measures the standard variation of the intervals between two consecutive tweets posted by the user. It reflect the frequency and pattern of how the user post the tweets. Formally it is defined as:

$$IntVar = \text{Standard-Deviation}\left(Intervals\right), \tag{2.6}$$

where $Intervals$ is a set of $Interval$ between all adjacent tweets posted by the user.

Retweet ratio ($RtRat$): *RtRat* is the ratio of the number of retweets (tweets starting with "*RT*") over the number of total tweets of that user. Formally it is defined as:

$$RtRat = \frac{\text{Num of retweets}}{\text{Total num of tweets}}. \tag{2.7}$$

Url ratio ($UrlRat$): *UrlRat* is the ratio of the number of tweets containing URL over the total number of tweets of that user. Formally it is defined as:

$$UrlRat = \frac{\text{Num of tweets with URL}}{\text{Total num of tweets}}. \tag{2.8}$$

Unique mention ($UniMen$): *UniMen* refers to the ratio of the number of unique mentions (tweeter's *screen_name* starting with @) over the total number of mentions of that user. Formally it is defined as:

$$UrlMen = \frac{\text{Num of unique mentions}}{\text{Total num of mentions}}. \tag{2.9}$$

Unique url ($UniUrl$): *UniUrl* refers to the ratio of the number of unique URLs over the total number of URLs. Formally it is defined as:

$$UniUrl = \frac{\text{Num of unique URLs}}{\text{Total num of URLs}}. \tag{2.10}$$

Polarity (Pol): *Pol* refers to the ratio of mentioning of "Obama" over mentioning of "Romney". It is basically an alignment metric to see whom the user is talking about more, "Obama" or "Romeny". Formally, the equation of computing *Pol* is defined as:

$$Pol = \frac{\text{Num of tweets mentioning } Obama - \text{Num of tweets mentioning } Romney}{\text{Total num of tweets of that user}}. \tag{2.11}$$

Tweet similarity ($TwtSim$): *TwtSim* is a metric measuring the average similarity of the user's tweets. It is calculated using cosine similarity over TF-IDF vector of tweets. TF-IDF, *term frequency-inverse document frequency* is a metric to measure the importance of a term in terms of a document. Assuming the user has N tweets, called a *Corpus* C of tweets. Each tweet can be viewed as a document. The IF-IDF value of term t in document i $tfidf\,(t,i)$ is calculated as:

$$tfidf\,(t,i) = tf\,(t,i) \times idf\,(t,C)\,, \tag{2.12}$$

where $tf\,(t,i)$ is the frequency of term t in tweet i, and $idf\,(t,C)$ is the inverse document frequency of term t in corpus C, which is formally defined as:

$$idf\,(t,C) = \log \frac{N}{1 + \|\{i \in C : t \in d\}\|}, \tag{2.13}$$

where N is the total number of documents (tweets in our case) in the corpus and $\|\{i \in C : t \in d\}\|$ is the number of documents containing term t.

Each tweet i has a TF-IDF vector

$$V_{TF-IDF}\,(i) = \left(tfidf\,(t_0,i)\,, tfidf\,(t_1,i)\,, \cdots, tfidf\,(t_{M-1},i)\right), \tag{2.14}$$

where t_0, t_1, $\cdots$, t_{M-1} comprise all the terms in corpus C.

Formally, the *TwtSim* is defined as:

$$TwtSim = \frac{\sum\limits_{i \in C,\ j \in C \text{ and } i \neq j} \text{Cosine-Similarity}\left(V_{TF-IDF}\,(i)\,, V_{TF-IDF}\,(j)\right)}{\frac{1}{2} \cdot N\,(N-1)}, \tag{2.15}$$

where C is the corpus of tweets, N is the total number of tweets in C and Cosine-

Similarity(V_1, V_2) is a function that calculate the cosine similarity between vector V_1 and vector V_2.

We experimented three combinations of non-sentiment features: the first set contains only the five *behavioural features*, which are *IntVar*, *RtRat*, *UrlRat*, *UniMen* and *UniUrl*. We denote this set of non-sentiment features as *Non-sentiment_5* for later analysis. The second combination contains five *behavioural features* and two *contextual features*, including *Pol* and *TwtSim*. We denote it as *Non-sentiment_7* in the future discussion. The third combination contains all three categories of features, which is *Non-sentiment_7* "plus" *factual profile features*. We denote it as *Non-sentiment_12*. We will describe the differences among these three sets later.

2.3.3.2 Sentiment Features

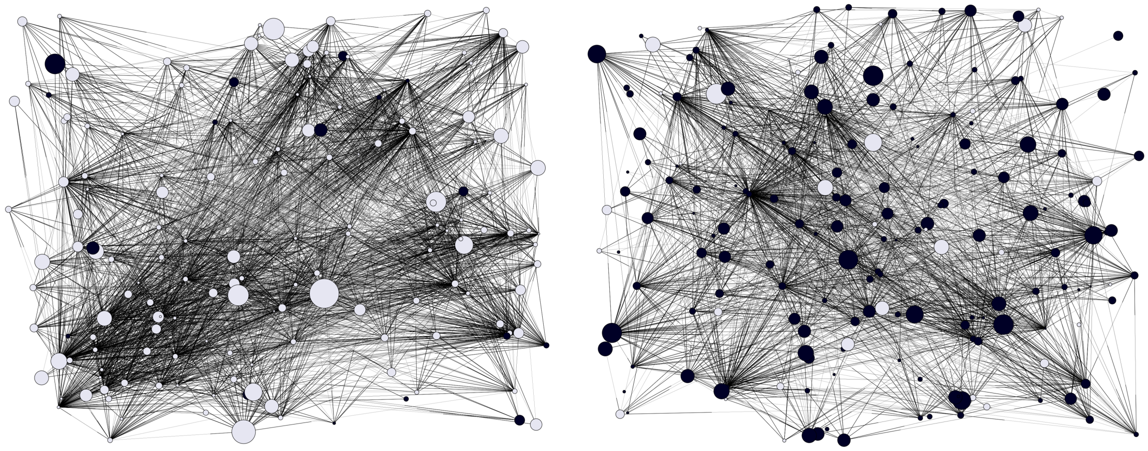
In our "*sentiment classfication model*", instead of using factual features, we use a vector of sentiment scores, as defined in Equation 2.3, as our feature vector. In our experiments, we compared the results from five different set of topics. The first set is *Sentiment_Top5*, the top 5 ranked topics in Table 2.6. The second set is *Sentiment_Middle5*, the middle 5 ranked topics in Table 2.6. The third set is *Sentiment_Last5*, the last 5 ranked topics in Table 2.6. The fourth set comprises 39 topics from the top half of Table 2.6, from hashtag No. 1 to hashtag No. 39, denoted as *Sentiment_TopHalf* for latter discussion. The fifth set comprises 39 topics from the bottom half of Table 2.6, from hashtag No. 40 to hashtag No. 78, denoted as *Sentiment_LatterHalf* for latter discussion. We will describe the observed differences among their performances. What's more, we conducted a comparison between sentiment features and non-sentiment features.

2.3.3.3 Graph-adjusted Sentiment Features

From the social relationship graph we have constructed before, we observed an interesting fact that tweeters with same political alignment tend to follow or to be followed by each other. This revealing fact might greatly help us achieve a more accurate classification of tweeters in terms of their political alignment. Figure 2.4a is the 2-level social relationship graph for tweeter *IBumbybee*, which is labelled as "republican" in our dataset. Figure 2.4b is the 2-level social relationship graph for tweeter *MsNatTurner*, which is labelled as "democrat" in our dataset. In Figure 2.4, the light nodes represent the users who are labelled as "republican", while the dark nodes represent the users who are labelled as "democrat". As we can see clearly, "democrats" tends to follow or to be followed by "democrats", while "republicans" tends to follow or to be followed by "republicans". Such "*clustering*" effect of tweeters bring us to propose a graph-based refinement on our sentiment classification model. The updating rule of sentiment score $SS_{U_i}(T_j)$ on topic T_j of user U_i is defined as:

$$SS_{U_i}(T_j) \leftarrow \alpha \cdot SS_{U_i}(T_j) + (1-\alpha) \cdot \frac{\sum_{U_k \in M} SS_{U_k}(T_j)}{N}, \tag{2.16}$$

where α is the adjustment factor, M is the set of outgoing neighbours of U_i and N is the size of M. This simply means that we "borrow" part of the sentiments of T_j from our friends in our social relationship graph. The value of α is ranging from 0 to 1. $\alpha = 0$ means that the sentiment score on a particular topic of a particular user totally depends on his friends. $\alpha = 1$ means that we do not apply the graph-based adjustment at all. Using the updated sentiment features, we construct the "*graph-adjusted sentiment model*" in our experiments. We propose this refinement for the following reason. Since the users might not talk about some topics, "borrowing" the



(a) *IBumbybee*("Republican") (b) *MsNatTurner*("Democrat")

Figure 2.4: 2-Level Social Relationship Graph

sentiments from their friends could enrich our data points.

2.3.4 Belief Propagation

Based on the observation discovered in Section 2.3.3.3, we propose another approach—Belief Propagation (BP) algorithm to solve our classification problem. In this case, knowing the ground truth (their labels) of a small set of tweeters, also called *seed*, we apply BP algorithm to predict the labels of the rest of tweeters.

BP is an approximation algorithm originally invented by Pearl [16] with the goal of solving the marginal probabilistic inference in the context of general graphs. Given a directed graph $G = (V, E)$ with a set V of nodes and a set E of edges, BP is used to estimate the marginal probability of undiscovered nodes based on the known probabilistic model of the other nodes. The "*belief*" of a node is denoted as the probability of the node being in a particular state. The *belief* of a node is influenced by the "*messages*" passed by the neighbours of that node, making the nature of BP as a "message-passing" algorithm. BP is usually implemented in an iterative manner; the algorithm stops when the marginal probabilistic distributions of all the nodes converge.

Denoting the *belief* of node i in a state x_k as $b_i(x_k)$, the computation of $b_i(x_k)$ depends on two factors: *(1)* the initial probability estimate of node i and *(2)* the mutual influence between the states of two neighbours.

Denoting the initial probability of node i being in state x_k as $\phi_i(i)$, and the probability of node j being in a state x_h given the probability of node i being in the state x_j as $\psi_{ij}(x_k, x_h)$, the *message* from i to j which estimates node i's perception of node j being in a particular state (x_h), is defined as:

$$m_{ij}(x_h) = \sum_{x_k \in S_i} \phi_i(x_k)\, \psi_{ij}(x_k, x_h) \prod_{l \in N(i) \setminus j} m_{li}(x_k)\,, \tag{2.17}$$

where $N(i)$ is the set of neighbors of node i and S_i represents all the states that node i could be in (In our case, $S = \{democrat, republican\}$).

Having a message from one node to another, the *belief* of node i being in state x_k is defined as:

$$b_i(x_k) = C\phi_i(x_k) \prod_{k \in N(i)} m_{ki}(x_k)\,, \tag{2.18}$$

where C denotes the normalization factor ensuring $\sum_{x_k \in S_i} b_i(x_k) = 1$.

At the beginning of BP, the unknown nodes are initialized with a *normal distribution*, giving equal probability for each possible state. In addition, all the messages are initialized as 1 before running BP. The messages get updated in each iteration and the algorithm stops when all the messages converge. And the messages are normalized at each iteration such that $\sum_{x_i \in S_i} m_{ki}(x_i) = 1$.

2.4 Evaluation Methodology

In order to evaluate the performance of the classification and BP inference accuracy described in 2.3, we employ two techniques, which are widely used in Machine Learning research.

2.4.1 Cross-Validation

Cross-Validation is a common evaluation technique in classification problems. In K-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples; Of the k subsamples, a single subsample serves as test data and the other $k-1$ subsamples are used as training data. The cross-validation result is the average of k repetitions, with each of the k subsamples used exactly once as test data.

In BP model described in Section 2.3.4 particularly, for K-fold cross-validation, one subsample is served as the known ground truth—the *seed* while the other $k-1$ subsamples are what we are going to predict. We repeat k times with each of the subsamples used exactly once as the *seed*.

All the K-fold cross-validation results discussed later are the average of 10 random K-fold cross-validations.

2.4.2 Accuracy

The prediction accuracy is a common way to measure the performance of classification model. Denoting the number of "democrats" predicted to be "democrat" as TD, the number of "democrats" predicted to be "republican" as FD, the number of "republicans" predicted to be "republican" as TR and the number of "republicans" predicted to be "democrat" as FR, the *Accuracy* of the model is defined as:

$$Accuracy = \frac{TD + TR}{TD + FD + TR + FR} \tag{2.19}$$

According to the above formula, *Accuracy* is basically the ratio of the number of correct estimates over the total number of predictions. The performances shown in the Result Section are all measured by the *Accuracy* defined above.

2.5 Results

In this Section, we are going to present the results of our models, which are *Non-Semtiment Classification Model* described in 2.3.3.1, *Sentiment Classification Model* described in 2.3.3.2, *Graph-adjusted Sentiment Classification Model* described in 2.3.3.3 and BP Model described in 2.3.4.

2.5.1 Does Automatic Topic Selection Work?

In order to evaluate our proposed automatic topic selection methodology, we applied different sentiment feature sets ranked by our automatic topic selection methodology. Table 2.7 compares the results among *Sentiment_Top5*, *Sentiment_Middle5* and *Sentiment_Last5*. As we know, these three feature sets are selected out of the automatic topic selection scheme described in Section 2.3.2. Before doing the experiments we expected that the top ranked topics—*Sentiment_Top5* could outperform the middle ranked topics—*Sentiment_Middle5* and latter ranked topics—*Sentiment_Last5*. In Table 2.7, column 2, 3 and 4 show the classification accuracies of *4-fold*, *5-fold* and *10-fold cross-validation* respectively.

As we can see from Table 2.7, the performance of *Sentiment_Top5* is better than that of *Sentiment_Middle5* and much better than *Sentiment_Last5*. The best accuracy *Sentiment_Top5* gets is 64.3 percent with 4-fold, while *Sentiment_Middle5* is getting an accuracy slightly smaller than 50 percent, which means not even better than a random guess. The *Sentiment_Last5* gets an accuracy as low as 39.8 percent, which is possibly because that the topics in *Sentiment_Last5* are somehow misleading.

Furthermore, Table 2.8 shows the comparative result between *Sentiment_TopHalf* and *Sentiment_LatterHalf*, which is also differentiated by our automatic topic selection scheme. In Table 2.8, column 2, 3 and 4 show the classification accuracies of *4-fold*, *5-fold* and *10-fold cross-validation* respectively.

From Table 2.8, *Sentiment_TopHalf* feature set surpasses *Sentiment_LatterHalf* feature sets with a classification accuracy of 0.627. The best result of *Sentiment_LatterHalf* feature set is 0.575 in *4-fold* cross-validation. Interestingly, the gap between these two sentiment feature set is not as big as the one observed from Table 2.7, probably because the dimensions of both *Sentiment_TopHalf* and *Sentiment_LatterHalf* are 38, which is large enough containing sufficiently rich information in both feature sets.

In summary, we could safely conclude that automatic topic selection does help to discover more distinguishing topics. However, we do not have a "good threshold" for *Max-Min_Diff* to select how many distinguishing topics as our features, which is the limitation of our automatic topic selection scheme.

Table 2.7: Comparative Result A for Automatic Topic Selection

Feature Set	4-fold	5-fold	10-fold
Sentiment_Top5	0.643	0.635	0.627
Sentiment_Middle5	0.495	0.485	0.484
Sentiment_Last5	0.412	0.398	0.399

Table 2.8: Comparative Result *B* for Automatic Topic Selection

Feature Set	4-fold	5-fold	10-fold
Sentiment_TopHalf	0.622	0.614	0.627
Sentiment_LatterHalf	0.575	0.574	0.568

2.5.2 Does Sentiment Matter?

In this section, we compare the performances of two sentiment feature sets, *Sentiment_Top5* and *Sentiment_TopHalf*, with the performances of three sets of non-sentiment features, *Non-sentiment_5*, *Non-sentiment_7* and *Non-sentiment_12*.

Table 2.9 shows the comparisons between sentiment feature sets and non-sentiment feature sets, from which we can observe several interesting results. *Sentiment_Top5* achieves the best performance with the highest prediction accuracy as 0.643. By comparing row *Sentiment_Top5* with row *Non-sentiment_12* and *Non-sentiment_7*, we can see that though not obvious, sentiment features yield slightly better classification result than non-sentiment features. Furthermore, the performance between *Non-sentiment_12* and *Non-sentiment_7* is pretty much even, while *Non-sentiment_5* yields a significant drop in performance, roughly as good as "random guess". As we discussed before, the only difference between *Non-sentiment_5* and *Non-sentiment_7* is that *Non-sentiment_7* contains two more features: *Polarity* and *Tweet_Similarity*. The performance gap between *Non-sentiment_7* and *Non-sentiment_5* shows that the contribution of *Polarity* and *Tweet_Similarity* is significant in Non-sentiment Classification Model.

As we can see, the sentiment model seems to make little difference compared to the non-sentiment model. In fact, several reasons could be behind the fact that sentiment classification model is not performing as well as what we expected. The first limitation is the data. There is unavoidable bias during the labelling of tweeters since it is conducted manually; the truncating of *high volume tweeters* also bring bias into our data. Thus, the data we have collected is probably not sufficient and representative enough. The second limitation comes from the sentiment analysis techniques we have used. Even though the NLP techniques is pretty mature and can easily handle formal written language processing, it is just not powerful enough to figure out complex irony tones or sarcasm in tweets. What's more, powerful sentiment analysis tool like Alchemy would also look naive in front of the informal "slang" used widely in Twitter. Sentiment in Twitter is far more complex than we thought.

2.5.3 Does Social Relationship Graph Matter?

As discussed in Section 2.3.3.3, we consider the value of the social relationship graph, where "users with similar political view tend to be grouped together". In order to take advantage of this valuable fact while still using sentiment features in classification, we propose "borrowing" sentiments from friends in the social relationship graph as we formally defined in Formula 2.16. We tested the effect of the value of α in Formula 2.16. Considering the *4-fold* cross-validation gives the best performance so far among all the experiments, we deploy the graph-based adjustment on this model only.

Table 2.10 shows the result of graph-adjusted sentiment model in terms of the

Table 2.9: Comparison between Sentiment Features and Non-sentiment Features

Feature Set	4-fold	5-fold	10-fold
Sentiment_Top5	0.643	0.635	0.627
Sentiment_TopHalf	0.622	0.614	0.627
Non-sentiment_12	0.625	0.630	0.635
Non-sentiment_7	0.630	0.621	0.631
Non-sentiment_5	0.512	0.533	0.525

Table 2.10: Result of Graph-adjusted Sentiment Classification Model

α	*Sentiment_Top5*	*Sentiment_Middle5*	*Sentiment_Last5*
0.1	0.853	0.779	0.807
0.2	0.825	0.747	0.811
0.3	0.801	0.759	0.796
0.4	0.777	0.750	0.774
0.5	0.764	0.741	0.770
0.6	0.774	0.730	0.749
0.7	0.787	0.728	0.744
0.8	0.755	0.724	0.745
0.9	0.781	0.741	0.747
1	0.643	0.495	0.412

adjustment factor α. As we can see, when considering the influence from friends, the classification accuracy jumps up 14 percent in feature set *Sentiment_Top5*, from 64.3% to 78.1%. And there is a trend that the smaller the α value is, the higher the accuracy of the model. Although the accuracy is not monotonously increasing as α value decreases, the accuracy reaches the highest, 85.3%, when $\alpha = 0.1$ on feature set *Sentiment_Top5*, which is more than 20 percent improvement from non-graphical model. It is a good indication that social information makes a significant difference in our case. It is noted that the graph structure improves results across all choices of sentiment topics. It is also noted that once graph structure is considered ($\alpha \neq 1$), the choice of sentiments do not seem to make significant difference.

2.5.4 Does Belief Propagation Work?

Given that graph information is dominating the sentiment of tweets, we also propose a completely graph-based model, the Belief Propagation (BP) Model to predict the association of political tweeters. In some cases, the BP algorithm is not able to give a prediction on some nodes because of the limitation of a certain graph structure. In this case, we assume two actions when this happens. For the tweeter that BP is not able to predict, we either do "*random guess*" or "*do not*

Table 2.11: Result of Belief Propagation Models

Model	2-fold	5-fold	10-fold	20-fold	50-fold
BP_randomguess	0.893	0.897	0.895	0.888	0.824
authBP_randomguess	0.873	**0.898**	0.893	0.890	0.886
BP_dontjudge	0.856	0.856	0.853	0.850	0.791
authBP_dontjudge	0.812	0.860	0.856	0.861	**0.863**

judge" which class it belongs to at all. Obviously "random guess" scheme would have a better accuracy since some of the unknown nodes have the chance to be guessed right. While not "judging", none the unknown nodes will be taken as correct prediction. We denoted these two schemes as *BP_randomguess* and *BP_dontjudge* respectively. Another case we have tested is to see the result by only considering the most authoritative tweeters as our "*seed*". The two same schemes in terms of unpredictable users are applied in this model, and denoted as *authBP_randomguess* and *authBP_dontjudge* respectively.

Table 2.11 shows the results of the above four BP models in *2-fold*, *5-fold*, *10-fold*, *20-fold* and *50-fold*. The best accuracy BP achieves is 0.898 by *authBP_randomguess* at 5-fold cross-validation. The best performance of BP is 86.3 percent accuracy, with "*dontjudge*". What's more, Apart from the *2-fold* cross-validation, the advantage of *authBP* over regular BP is obvious, especially in *50-fold*, when only 2 percent of the most authoritative nodes are served as "seed". In other words, in the absence of enough ground truth information, using most authoritative nodes in the graph can greatly improve the inference accuracy of BP.

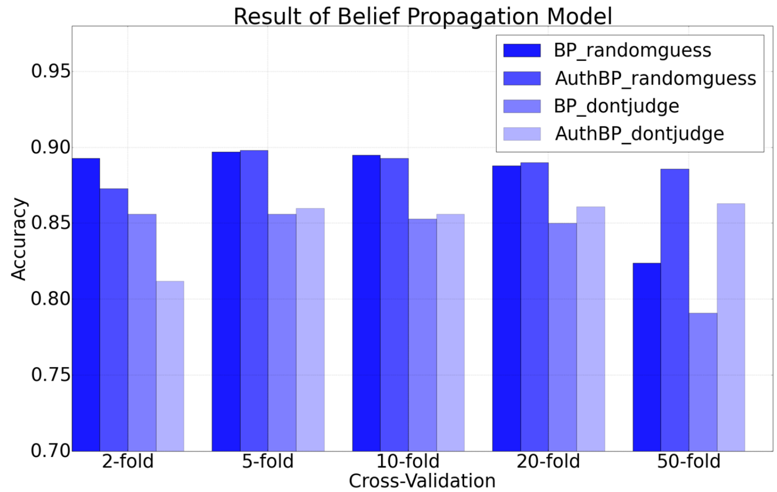


Figure 2.5: Result of Belief Propagation Model

2.6 Conclusion

In this project, first of all, we proposed a model to classify political tweeters by using sentiments in tweets. We came up with an automatic way to select the most distinguishing topics from tweets. We have shown that sentiment classification model reaches a classification accuracy as 64.3 percent, unfortunately, not much different from non-sentiment classification model. Besides, we also found that social relationship graph reveals lots of information in separating politically active tweeters. By taking the advantage of social relationship graph, we refine the sentiment classification model, which achieves a classification accuracy as high as 85 percent. What's more, the accuracy of graph-adjusted sentiment model increase when the graph information takes more effects. Finally, we deployed an alternative approach—Belief Propagation inference model on predicting the political alignment of tweeters based only on social graph, which also achieves high prediction accuracy.

We have concluded that the limitation of our sentiment model comes from the limitation of data collection as well as the sentiment analysis techniques. With *graph-*

adjusted sentiment model, we achieve a decent classification accuracy. On the other hand, it also shows us that graph factor takes a more significant role in separating tweeters than sentiment factor. Our alternative approach, the Belief Propagation inference model also does well in predicting the political association of tweeters. We have also seen that, the selection of the "*seeds*" obviously has an effect on the performance of Belief Propagation model.

3. ALGORITHMIC UNIVERSITY PROGRAM RANKING

3.1 Introduction

Ranking is a general and popular problem in our community as well as on the Web. Simply speaking, ranking is a knowledge discovery method, revealing the ordered, organized truth hidden behind disordered and unstructured data. For example, U.S. News provides rankings on academic organizations, providing reference to people choosing educational schools. Google deploys a PageRank [17] algorithm to determine the relevance and importance of a web page. In this project, our motivation is simply and clearly stated: we are trying to develop a simple and effective method to rank universities, graduate schools/programs in particularly. This work could be valuable by providing people reference when choosing schools to pursue higher education.

According to the statement from U.S. News website [18], they rank the graduate programs from both statistical data and expert assessment data. The statistical data includes both input and output measures. The input measures reflect the quality of students, faculties and any other resources into the programs. The output measures reflect the educational outcomes of the graduates from the academy. The expert assessment data is collected from the input of program deans. Each dean is asked to rank a program from 1 to 5 and the average rates are used to rank programs. Finally these two types of data is normalized, weighted and totaled into a ranking score. Besides, social scientists have also done research on university ranking methodologies. For example, Lukman et al in [19] proposed a model to compare universities regarding educational and environmental performances. A comprehensive study on the indicators, dimensions, methodologies in university ranking from sociology perspective of view is provided in [20]. Leydesdorff and Shin provided a new idea to rank universities in terms of their relative citation counts in [21]. Other metrics such as publication counts, industry hiring preferences have been used as well.

All these works on ranking universities are valuable. Differently, our work is trying to invent a new algorithmic methodology to solve this problem, which can achieve reasonable and reliable ranking of university programs. Some well-established graph-based ranking algorithms exist. Kleinberg proposed an effective reinforced algorithm to calculate the hubs and authorities in the hyperlinked environment in [22]. One year later, Page and Brin in [17] proposed the well-known PageRank, which is the fundamental algorithm of Google Search Engine. In order to solve our own algorithmic university ranking problem, we apply these techniques on our university hiring graph.

3.2 Data Set

3.2.1 Data Description

The intuition behind our thought is that "schools usually hire PhD graduates from better or peer schools". It means that, for example, if Harvard University (Harvard) hires a PhD graduate student from Cornell University (Cornell) to become a faculty member, it indicates that Cornell is at least as good as or better than Harvard. Since the hiring process is operated by local experts from each department or school, we believe that it reveals more sophisticated qualities of a program than U.S. News measurement does. Based on this hypothesis, we develop an algorithmic method, which use proper ranking algorithms to come up with our own ranking of programs in a particular field. Our algorithms will be applied on the so-called "*hiring graph*" of universities. For a simple example, if Harvard hires a PhD from Cornell, in the "*hiring graph*", there would be one directed edge from Harvard to Cornell with weight

1, and Harvard and Cornell are two nodes in the graph. Therefore, the university ranking problem simply becomes a graph-based ranking problem.

In order to construct such graph, we collected two faculty profile data sets in March 2014, from top 50 Computer Science (CS) Departments [23] and top 50 Mechanical Engineering (ME) Departments [24] across the USA respectively. We did not combined these two data sets together even though we have found a few ME professors were graduated with CS PhD. We collected these two separate data sets in order to show that our method will work in more than one field. For each faculty in our data set, we collected two pieces of information, where and when did the faculty get his/her PhD. Table 3.1 shows the sample data that we collect. In Table 3.1, column 2 to 5 represent all 4 entries for each faculty: 1) *Dept.*: Department that the faculty member works in; 2) *Univ.*: The university that the faculty member works in; 3) *PhD From*: Which university the faculty member graduated from as PhD; 4) *Year Grad.*: the year the faculty got his/her PhD.

Several things that we have to point out for our data set. First of all, we do not collect the name for each faculty for privacy concerns. Some of the professors are not posting their educational information on the web at all. Luckily, most of the faculties disclose their resume or educational information on their department page or personal page, making it possible for us to collect a large enough sample of the hiring graph. What's more, all the faculty data we collected is the current status of each program. This is to say that, the graph only reflects current employment and does not reflect historical employment. The graph also does not reflect the the hiring decisions that may have been terminated without tenure.

Unfortunately, since we cannot find any organization that can provide such data, all the data, we have, is collected on the website of each graduate program. Considering the data we want is posted in different format on each web page, we collected them manually instead of writing a crawler.

For the top 50 CS programs data set, we collected data from 2,018 faculty members currently in those programs. 1,793 (88.9%) faculty members out of the total, have PhD graduation year information on their web pages.

For the top 50 ME programs data set, we collected data from 1,941 faculty member currently in those programs. 1,709 (88.0%) faculty members out of the total, have educational year information on their web pages.

We note that this is a small sample of graduates from these programs. Second, a faculty member's career lasts 30 to 35 years or more and hence the data reflects the hiring decisions made over several years. Our data reflects that the faculty PhD graduation that range from 1949 to 2014 in CS data set and from 1946 to 2013 in ME data set. This enables us to bin the data based on year of graduation to obtain a historical progression of school new hirings.

While our methodology can be applied to the entire hiring graph of all CS or ME programs, we restrict ourselves to top 50 programs due to difficulties in collecting the data manually.

Table 3.1: Data Format Sample

Faculty	Dept.	Univ.	PhD From	Year Grad.
F1	CS	CMU	MIT	2005
F2	CS	Princeton	UTAustin	2009
F3	CS	TAMU	UIUC	1997
F4	ME	Cornell	Caltech	1987
F5	ME	UCLA	UCBerkeley	1991
F6	ME	Purdue	Stanford	2012

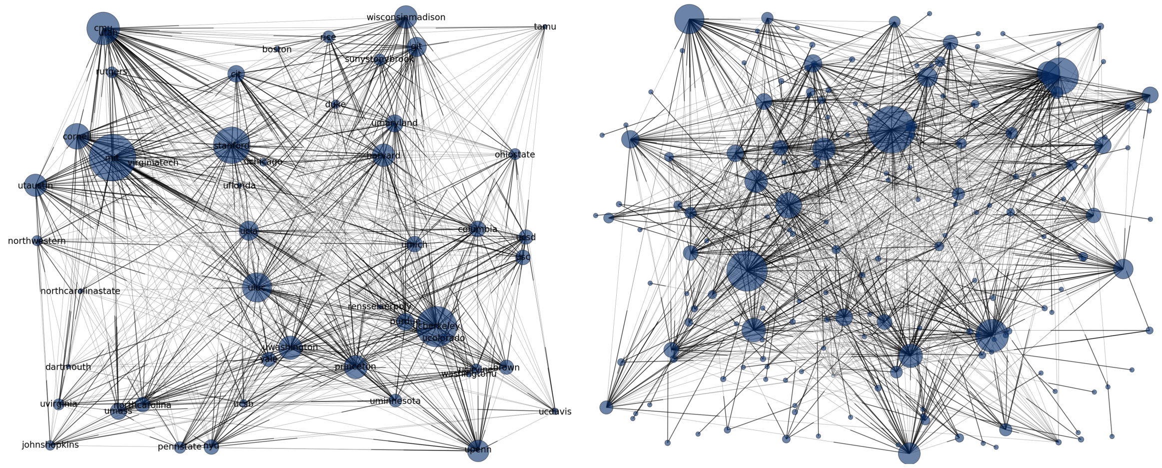
3.2.2 "Hiring Graph"

Our algorithms will be applied on the so-called mutually "*hiring graph*" of universities. For a simple example, if Harvard hires a PhD from Cornell, in the "*hiring graph*", there would be one directed edge from Harvard to Cornell with weight 1, and Harvard and Cornell are two nodes in the graph. Therefore, the university ranking problem simply becomes a graph-based learning problem.

Mathematically, the hiring graph could be denoted as a directed graph $G = (V, E)$, comprising a set V of nodes (universities) and a set E of edges. Edge $E(x, y)$ means there is at least one PhD from university y hired in university x as faculty. In the hiring graph, one university might hire several PhD graduates from another university as faculty members. In this case, we set the weight of each edge to be the number of PhD graduates hired from that university. For example, assuming university A hires 9 PhDs graduates from university B, regardless of the year in which they graduated, the weight of edge $E(A, B)$ would be 9.

In our CS data set, for example, many faculties would come from universities outside the top 50, such as Hebrew University (Hebrew) and University of Toronto (UToronto). In our CS data set, we have 182 universities in our graph in total, among which are the top 50 from U.S. News. In our ranking experiments, we might or might not consider those universities outside the top 50 while running our algorithms, and we stick to the top 50 for ranking. Similarly, there are 211 universities other than the top 50 in our ME data set. Figure 3.1a shows an example of how the hiring graph looks like exclusively for top 50 CS schools. Comparatively, Figure 3.1b shows an example of how the hiring graph looks like when considering all the recorded CS schools in our data set. We will discuss the difference of both cases in our results.

One thing we want to point out is that, some universities hire their own PhDs.



(a) Hiring Graph Subtracted (b) Hiring Graph Extended

Figure 3.1: Hiring Graph in CS Data Set

Hence there are some self-edges pointing to and pointed by the same node in the hiring graph. We expect that things might change in terms of whether we consider these self-edges or not. In our experiments, we will discuss the difference between both cases.

3.3 Our Approach

The hiring graph and ranking of graduate programs has similarities to web page and rankings of web pages. In page ranking methodologies, links pointing to a web page are seen to increase the authority or importance of those pages [17, 22]. Similarly, we postulate that hiring links in our graph provide similar information in our hiring graph about the quality of the graduate programs. Hence, we propose to employ page ranking algorithms on the hiring graph to ascertain the quality of the programs.

The simplest way is to rank graduate programs according to their *in-degree*, which basically represents "the number of PhD got hired by other schools" in the hiring graph. We also applied various link-based algorithms based on PageRank (PR) [17] and Hyperlink-induced Topic Search (HITS) [22] to do the ranking. In this section we are going to describe the algorithms that we used.

3.3.1 PR-based Algorithms

3.3.1.1 PR Algorithm

PR algorithm is originally invented to rank web pages according to their relative importance, which later became the foundation of Google Search Engine. It uses link structure of web pages exclusively without any text information on the web pages. It is based on a model called *random surf model*, in which a random surfer is assumed to periodically jump to any random web page in the Web [17].

According to our assumptions described before, an incoming edge of university p would also means the importance of the university, which is consistent with idea of PageRank. Thus we think that PageRank (PR) like algorithms could be applied in our problem.

Here we describe an iterative manner of computing the PR value of every node in a graph. Let $G = (V, E)$ be the directed graph with a set V of vertices or nodes and a set E of edges. At the beginning, the PR scores of all nodes are initialized as $\frac{1}{N}$ where N is the total number of nodes in the graph. In each iteration, the PR score $r(p_i)$ of node p_i is defined as:

$$r(p_i) = \frac{(1-\alpha)}{N} + \alpha \cdot \sum_{p_j \in M(p_i)} \frac{r(p_j)}{L(p_j)} \tag{3.1}$$

where N is the total number of nodes, $p_0, p_1, \ldots, p_{N-1} \in V$, $M(p_i)$ is the set of pages that link to p_i, $L(p_j)$ is the number of outgoing links from p_j, and α is the damping factor. Letting the damping factor $\alpha = 0.85$ is a democratic choice according to Page in [17], which has been proved to be effective through a large number of experiments. Hence, in our PR-based approach we also use the same value, 0.85, as our damping factor. The algorithm stops when the PR scores converge, or in other words, remain unchanged or change little between two consecutive iterations.

As we can see in equation 3.1, the PR scores of $p_j \in M(p_i)$ is bringing a normalized effect to node p_i since the PR score of p_j is divided by the number of outgoing links of p_j. In addition, since the edges in the Web graph are not weighted or are all weighted as 1, we dont take the edge weight into effect.

3.3.1.2 Weighted PR Algorithm with Weights Normalized

In our problem, one significant difference of the hiring graph with the web graph is that every edge in the hiring graph has a weight. Hence we refined the original PR algorithm by taking the edge weight into consideration, which is called weighted PR algorithm. When still considering the normalization effect as it does in the original PR algorithm, the new formula of the PR score $r(p_i)$ of node p_i would become

$$r(p_i) = \frac{(1-\alpha)}{N} + \alpha \cdot \sum_{p_j \in M(p_i)} \frac{r(p_j) \cdot w(\varepsilon(p_j, p_i))}{W(p_j)} \tag{3.2}$$

where N is the total number of nodes, $p_0, p_1, \ldots, p_{N-1} \in V$, $\varepsilon(p_j, p_i) \in E$, $w(\varepsilon(p_j, p_i))$ denotes the weight of $\varepsilon(p_j, p_i)$, $M(p_i)$ is the set of pages that link to p_i, α is the damping factor, and $W(p_j)$ is the sum of the weights of outgoing links from p_j, whose formula is:

$$W(p_j) = \sum_{\varepsilon(p_j, p_k) \in E} w(\varepsilon(p_j, p_k)). \tag{3.3}$$

3.3.1.3 Weighted PR Algorithm with Weights Unnormalized

We also tested another refinement of PR algorithm, in which the incoming link effect is not normalized by a factor of incoming links' outgoing factor, but the total number of nodes in the graph. In this case, since the splitting factor is fixed and hence the normalization effect does not take into account for any node. This version of formula is defined as:

$$r\left(p_i\right) = \frac{(1-\alpha)}{N} + \alpha \cdot \sum_{p_j \in M(p_i)} \frac{r\left(p_j\right) \cdot w\left(\varepsilon\left(p_j, p_i\right)\right)}{N} \tag{3.4}$$

where N is the total number of nodes, $p_0, p_1, \ldots, p_{N-1} \in V$, $\varepsilon\left(p_j, p_i\right) \in E$, $w\left(\varepsilon\left(p_j, p_i\right)\right)$ denotes the weight of $\varepsilon\left(p_j, p_i\right)$, $M(p_i)$ is the set of pages that link to p_i and α is the damping factor.

As we can see in formula 3.4, without normalization, the actual number of edges and edge weights matter. We expected that those programs with large in-degree might probably take the advantage of unnormalization.

3.3.2 HITS-based Algorithms

3.3.2.1 HITS Algorithm

HITS algorithm is proposed by Kleinberg, with the initial intention to discover the "authoritative" sources of a particular topic in the WWW. Let $G = (V, E)$ be the Web graph comprising a set V of vertices (pages) and a set E of links. Innovatively, it defines two types of pages in the Web: *hubs* and *authorities*. A *hub* is a page that links to other pages; an *authority* is a page that is linked by other pages. The ranking philosophy behind HITS is *mutually reinforcing relationship*: "a good *hub* is a page that points to many good authorities'; a good *authority* is a page that is pointed to by many good hubs"[22]. HITS is usually implemented in iterative manner. In each iteration, the updating rules for the authority value $Auth\,(p)$ and hub value $Hub\,(p)$ of page p is formulated as

$$Auth\,(p) \leftarrow \sum_{\varepsilon(q,p)\in E} Hub\,(q) \tag{3.5}$$

and

$$Hub\,(p) \leftarrow \sum_{\varepsilon(p,q)\in E} Auth\,(q). \tag{3.6}$$

In each iteration the new values are updated from old values from last iteration. After each iteration, the hub scores and authority scores should be normalized before starting the next iteration. The algorithm will stop once the hub scores or authority scores converge (remain the same or change little).

Unlike PR algorithm, HITS algorithm considers the effect of hubs into account. In HITS algorithm, the effect of hubs and authorities will reinforce each other and those authorities pointed by strong hubs will stand out of those authorities pointed by weak hubs. In the hiring graph specially, to UCBerkeley for example, we expect that a link from MIT would be more important than say, a link from TAMU, because MIT has more credits to support UCBerkeley to be a better school. Under this assumption, in our experiments, we developed and tested several variations of HITS-based algorithm on our PhD hiring graph. Finally, we look at the authority scores of each program and rank them according to their authorities only.

3.3.2.2 Weighted HITS Algorithm

Similarly, we also take the weights of edges in the hiring graph into consideration. The updating rules are defined in equation 3.7 and equation 3.8 for the weighted HITS algorithm. The only difference in the following updating rules from the formula of HITS is that we multiply the weight of the incoming/outgoing edges when calculating the authority/hub of a given node.

$$Auth\,(p) \leftarrow \sum_{\varepsilon(q,p)\in E} Hub\,(q)\cdot w\,(\varepsilon\,(q,p)) \tag{3.7}$$

and

$$Hub\,(p) \leftarrow \sum_{\varepsilon(p,q)\in E} Auth\,(q)\cdot w\,(\varepsilon\,(p,q)), \tag{3.8}$$

where $w\,(\varepsilon\,(p,q))$ is the weight of edge from node p to node q.

3.3.2.3 HubAvg Algorithm

To overcome the shortcoming of the HITS algorithm of a hub getting a high weight when it points to a large number of low quality authorities, we also suggest the following refinement according to [25]. While the updating rule for authority remains the same, the updating rule for hub is averaged by the number of outgoing edges of the node:

$$Auth\,(p) \leftarrow \sum_{\varepsilon(q,p)\in E} Hub\,(q)\cdot w\,(\varepsilon\,(q,p)) \tag{3.9}$$

and

$$Hub\,(p) \leftarrow \frac{1}{M\,(p)} \sum_{\varepsilon(p,q)\in E} Auth\,(q)\cdot w\,(\varepsilon\,(p,q)), \tag{3.10}$$

where $w\,(\varepsilon\,(p,q))$ is the weight of edge from node p to node q and $M\,(p)$ is the sum of weights of outgoing edges of node p.

3.4 Evaluation Methodology

In order to evaluate the performance of the above link-based algorithms, we proposed using the U.S. News ranking as a baseline. However, this is not to say that U.S. News ranking is the "ground truth" since U.S. News ranking is also a subjective point of view. We only use it as a reference to analyze our own ranking method so that we could discuss and conclude from what we have observed. Since we also determined the top schools from the U.S. News website, U.S. News ranking would be a good reference for our comparative experiments.

3.4.1 RankDistance

In order to measure the distance between two rankings, we proposed an "edit-distance"-like measurement, called "RankDistance". The computation of "RankDistance" is described as follow. Supposing R_1 and R_2 are two rankings for a set of samples $S = (a_0, a1, \ldots, a_{N-1})$. Defining the rank of a_i in R_j as $P_{R_j}(a_i)$, The RankDistance $RankDist(R_1, R_2)$ between R_1 and R_2 is:

$$RankDist(R_1, R_2) = \frac{\sum\limits_{a_i \in S} |P_{R_1}(a_i) - P_{R_2}(a_i)|}{N}, \tag{3.11}$$

where N is the total number of samples.

From equation 3.11 we can see that the smaller the $RankDist(R_1, R_2)$ is, the closer R_1 and R_2 are. In our experiments, we compared our method with U.S. News ranking using $RankDist$. As we said before, we are not taking U.S. News as the ground truth with which our results have to perfectly match.

3.4.2 Sensitivity Analysis

Apart from $RankDist$, which measures how close the ranking to U.S. News, we also proposed another measurement called "sensitivity analysis", which measures how robust the algorithm is to small changes in data. The intuition of sensitivity analysis is that universities keep hiring and professors retire or leave universities every year for whatever reasons. Thus the hiring graph keeps changing slightly year after year. We do not expect significant change in our ranking results if a minor change happened in the hiring graph. The sensitivity analysis looks at this issue.

Our methodology to measure the sensitivity is simple. For each ranked program, we carry out two hypothetical changes separately in the graph regarding this program: *1)* add a non-existing edge from one top ranked program to this program; *2)* delete one existing edge from the best program that link to this program; if not available, delete one existing edge from the best program that linked by this program. The first change will boost the rank of the program and the second change will lower the rank of the program. Thus by running a specific algorithm, we will have both an upper bound and a lower bound for each program. We will present and discuss the sensitivity analysis results in detail in the following section.

3.5 Results

In this section we are going to present our experimental results on our two data sets: Top50 CS and Top50 ME. We tested five methods in all. They are: in-degree Ranking, weighted PageRank algorithm with weights normalized, weighted PageRank algorithm with weights unnormalized, weighted HITS algorithm and Hubavg algorithm. Table 3.2 provides a mapping between each algorithm and its abbreviation, which will be used in the following thesis for convenience.

For each data set, we are going to present the result of each method compared with the U.S. News ranking. In addition, we divided our data set into two roughly equal parts by a given year, to see whether there is a difference between the ranking in the most recent years and the ranking in the earlier years. Furthermore, we performed some case studies for an in-depth look into a few universities that are ranked differently between U.S. News and our approach. Moreover, we conducted sensitivity tests for all the algorithms.

3.5.1 Top50 CS

3.5.1.1 Graph subtracted or extended? Self-edges Retained or Removed?

Considering the entire Top50 CS data set, we have 182 schools and 1,106 edges. We generated a subtracted graph that only contains the top 50 schools. In the subtracted graph, we have 50 schools and 842 edges. In addition, as we know that there are self-edges in the graph, we also compared the differences between the one with self-edges and the one with self-edges removed.

Table 3.3 shows the results of our algorithms compared with U.S. News Ranking using *RankDist* measurement. According to the definition of *RankDist*, given a set of 50 samples, the maximum *RankDist* between two rankings we can get is 25, which occurs when one is exactly the reverse of the other one. Another common case is that, when we randomly shuffle the ranking, we get a *RankDist* about 16.63 obtained by 1000 trials of random shuffles. In Table 3.3, column 1 is a list of algorithms that we have applied; column 2 shows the *RankDist* to the U.S. News ranking when we employ the algorithm on *subtracted graph with self-edges retained*; column 3 shows the *RankDist* to the U.S. News ranking when we employ the algorithm on *subtracted graph with self-edges removed*; column 4 shows the *RankDist* to the U.S. News ranking when we employ the algorithm on *extended graph with self-edges retained*; column

Table 3.2: Algorithms and their Abbreviations

Algorithm	Abbreviation
In-degree Ranking Algorithm	IndeRank
Weighted PR Algorithm with weights normalized	WeightedPR_w_n
Weighted PR Algorithm with weights unnormalized	WeightedPR_wo_n
Weighted HITS Algorithm	HITS_Weighted
Hubavg Algorithm	HITS_Hubavg

Table 3.3: Results on Top50 CS Data Set

	RankDist to the U.S. News Ranking				
Algorithm	Subtracted graph with self-edges	Subtracted graph w/o self-edges	Extended graph with self-edges	Extended graph w/o self-edges	**Average**
Max.	25.0	25.0	25.0	25.0	25.0
RandomShuffle	16.63	16.63	16.63	16.63	16.63
IndeRank	5.04	3.92	5.0	4.08	**4.51**
WeightedPR_w_n	5.28	5.04	5.08	4.72	**5.05**
WeightedPR_wo_n	5.0	4.44	4.76	3.92	**4.53**
HITS_Weighted	4.72	4.2	4.72	4.16	**4.45**
HITS_Hubavg	4.44	3.92	4.4	3.88	**4.16**
Average	**4.896**	**4.304**	**4.792**	**4.152**	—

5 shows the *RankDist* to the U.S. News ranking when we employ the algorithm on *extended graph with self-edges removed*. The last column and the last row show the average of each row and each column respectively.

We can see clearly from Table 3.3, the *RankDist* values in column 4 are generally smaller than the values in column 2, which means that the results on *extended graph with self-edges retained* is closer to the ranking of U.S. News than the results on *subtracted graph with self-edges retained*. This is probably because we have more structural information in the extended hiring graph, even though we did not rank those programs outside the top 50 programs in our record. Particularly, HITS_Hubavg algorithm seems to do the best job in both cases, with *RankDist* 4.44 and 4.4 respectively. Following HITS_Hubavg, HITS_Weighted, WeightedPR_wo_n and IndeRank are also doing pretty good jobs in both cases with self-edges retained.

By comparing column 2 and column 3, we can see that, except WeightedPR_w_n, *RankDist* values in column 3 are all smaller than those in column 2, indicating that results obtained from graph without self-edges are generally closer to the U.S. News ranking compared with the graph with self-edges. In addition, by comparing column 4 and column 5, we can observe similar fact that all five algorithms are doing better in column 5 than those in column 4. The above observations indicate that removing self-edges in hiring graph helps improving the performance of algorithms. This is to say removing self-edges helps removing noises in hiring graph because self-edges do not reflect the mutual relationship between schools. In the case of *extended graph with self-edges removed*, HITS_Hubavg is performing the best with a *RankDist* of 3.88, then comes WeightedPR_wo_n (3.92), IndeRank (4.08) and HITS_Weighted (4.16). What's more, we can see that HITS-based algorithms are generally performing better than PR-based algorithms, probably because they consider the mutual reinforced effect from both hubs and authorities. Furthermore, WeightedPR without normalization is doing consistently better than WeightedPR with normalization.

By comparing the average *RankDist* obtained from each algorithm in all four cases, HIT_Hubavg yields the smallest *RankDist* of 4.16, followed by HITS_Weighted (4.45), IndeRank (4.51) and WeightedPR_wo_n (4.53).

By comparing the average *RankDist* obtained from all four case across all five algorithms, we can see that *extended graph without self-edges* achieves the smallest *RankDist* of 4.152, followed by *subtracted graph without self-edges* (4.304), *extended graph with self-edges* (4.792) and *subtracted graph with self-edges* (4.896).

3.5.1.2 Original Rankings

Table 3.4 and Table 3.5 show the original rankings of U.S. News accompanied with the results of all five algorithms in our experiments. As a side note, all the results in Table 3.4 and Table 3.5 are retrieved from the experiments on the *Extended Graph with self-edge* ***removed*** of the entire CS data set. A number of observations can be made from the results. MIT, CMU, Stanford and UCBerkeley always occupy the top 4 schools in the rankings and the top 20 schools are ranked roughly consistent across all the algorithms. What's more, CMU seems to be a little bit over-ranked by

U.S. News and MIT stands out all the time in all our five algorithms. At the first glance, our approach yields reasonable rankings that are consistent. It indicates that our approach is an effective way to rank graduate programs.

By comparing the results from WeightedPR_w_n and WeightedPR_wo_n, the ranking of some schools are dramatically different. For example, the in-degrees of UIUC and Harvard are 77 and 49 respectively, which means that UIUC is a larger program than Harvard. However, they are ranked differently by these two algorithms. UIUC is ranked No. 5 in WeightedPR_wo_n while No. 8 in WeightedPR_w_n; Harvard is ranked No. 5 in WeightedPR_w_n while No. 8 in WeightedPR_wo_n. For another example, the in-degree of UCLA and Caltech are 38 and 26 respectively. However, they are ranked differently by these two algorithms. UCLA is ranked No. 13 in WeightedPR_wo_n while No. 18 in WeightedPR_w_n; Caltech is ranked No 11 in WeightedPR_w_n while No. 15 in WeightedPR_wo_n. This is to say that, large programs, like UIUC and UCLA, are ranked higher in WeightedPR_wo_n than WeightedPR_w_n, while small programs, like Harvard and Caltech, are ranked higher in WeightedPR_w_n than WeightedPR_wo_n. As we know, it is the actual number of incoming edges that matters in WeightedPR_wo_n, in which case those large programs which places lots of PhDs would take the advantage. On the other hand, WeightedPR_w_n, which normalizes this effect, would not favor those large programs any more. In this case, the quality of PhD placements would take the advantage over the quantity of PhD placements. This also explains that Harvard with smaller in-degree is ranked even higher than UIUC with larger in-degree in WeightedPR_w_n. In sum, for PR-based algorithms, normalization is favoring smaller programs while unnormalization is favoring bigger programs, as expected. Considering WeightedPR_wo_n yields closer result to U.S. News ranking, we can conclude that the U.S. News might probably favor bigger programs as well.

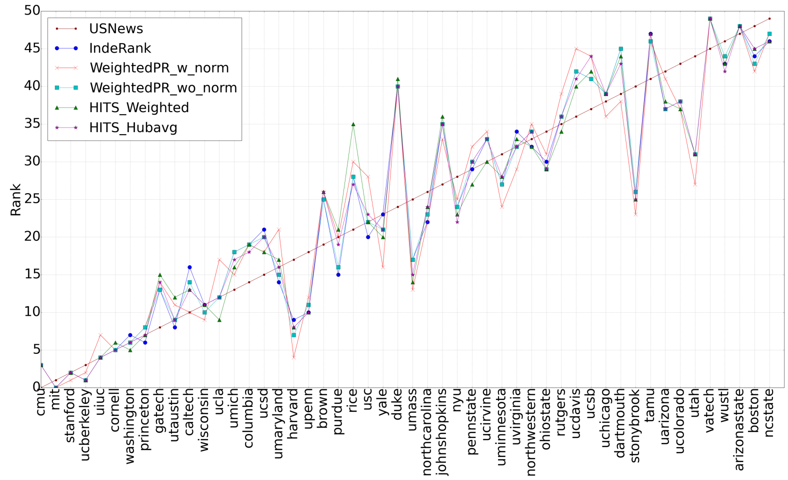


Figure 3.2: Ranking Divergence of CS Programs Compared to U.S. News

What's more, the two HITS-based algorithms, HITS_Weighted and HITS_Hubavg seem to give very similar rankings according to Table 3.4 and 3.5. This is because HITS-based algorithm is more stable than PR-based algorithms since HITS-based algorithms take the effects from both hubs and authorities into consideration.

Besides, we can also observe some significant differences in rankings for some schools. For example, Harvard is ranked much higher by our method than U.S. News. This is probably because Harvard is still getting the momentum from the earlier days when Harvard was strong in CS. Figure 3.2 shows the ranking divergence for each program between our algorithms against U.S. News. In Figure 3.2, the straight line is the baseline of U.S. News ranking. The curves of our algorithms are roughly consistent with the U.S. News baseline. We can also observe that there are some programs with huge divergence between our approach and the U.S. News ranking, such as Harvard, Duke, StonyBrook and Utah. We will discuss some interesting cases in detail in section 3.5.1.4 later.

Table 3.4: Results on Top50 CS Data Set (1∼25)

The rankings are retrieved from experiments on the entire ***Extended Graph with Self-edge Removed***						
Rank	USNews	IndeRank	WeightedPR_w_n	WeightedPR_wo_n	HITS_Weighted	HITS_Hubavg
1	cmu	mit	mit	mit	mit	mit
2	mit	ucberkeley	stanford	ucberkeley	ucberkeley	ucberkeley
3	stanford	stanford	ucberkeley	stanford	stanford	stanford
4	ucberkeley	cmu	cmu	cmu	cmu	cmu
5	uiuc	uiuc	harvard	uiuc	uiuc	uiuc
6	cornell	cornell	cornell	cornell	washington	cornell
7	washington	princeton	washington	washington	cornell	washington
8	princeton	washington	uiuc	harvard	princeton	princeton
9	gatech	utaustin	princeton	princeton	harvard	harvard
10	utaustin	harvard	wisconsin	utaustin	ucla	utaustin
11	caltech	upenn	caltech	wisconsin	upenn	upenn
12	wisconsin	wisconsin	utaustin	upenn	wisconsin	wisconsin
13	ucla	ucla	upenn	ucla	utaustin	ucla
14	umich	gatech	umass	gatech	caltech	caltech
15	columbia	umaryland	gatech	caltech	umass	gatech
16	ucsd	purdue	umich	umaryland	gatech	umass
17	umaryland	caltech	yale	purdue	umich	umaryland
18	harvard	umass	ucla	umass	umaryland	umich
19	upenn	umich	ucsd	umich	ucsd	columbia
20	brown	columbia	columbia	columbia	columbia	purdue
21	purdue	usc	purdue	ucsd	yale	ucsd
22	rice	ucsd	umaryland	yale	purdue	yale
23	usc	northcarolina	northcarolina	usc	usc	nyu
24	yale	yale	stonybrook	northcarolina	nyu	usc
25	duke	nyu	uminnesota	nyu	northcarolina	northcarolina

Table 3.5: Results on Top50 CS Data Set (26∼50)

The rankings are retrieved from experiments on the entire ***Extended Graph with Self-edge Removed***						
Rank	USNews	IndeRank	WeightedPR_w_n	WeightedPR_wo_n	HITS_Weighted	HITS_Hubavg
26	umass	brown	nyu	brown	stonybrook	stonybrook
27	northcarolina	stonybrook	brown	stonybrook	brown	brown
28	johnshopkins	uminnesota	utah	uminnesota	pennstate	rice
29	nyu	rice	usc	rice	uminnesota	uminnesota
30	pennstate	pennstate	uvirginia	ohiostate	ohiostate	ohiostate
31	ucirvine	ohiostate	rice	pennstate	ucirvine	pennstate
32	uminnesota	utah	ohiostate	utah	utah	utah
33	uvirginia	northwestern	pennstate	uvirginia	northwestern	uvirginia
34	northwestern	ucirvine	johnshopkins	ucirvine	uvirginia	ucirvine
35	ohiostate	uvirginia	ucirvine	northwestern	rutgers	northwestern
36	rutgers	johnshopkins	northwestern	johnshopkins	rice	johnshopkins
37	ucdavis	rutgers	uchicago	rutgers	johnshopkins	rutgers
38	ucsb	uarizona	ucolorado	uarizona	ucolorado	uarizona
39	uchicago	ucolorado	dartmouth	ucolorado	uarizona	ucolorado
40	dartmouth	uchicago	rutgers	uchicago	uchicago	uchicago
41	stonybrook	duke	duke	duke	ucdavis	duke
42	tamu	ucsb	uarizona	ucsb	duke	ucdavis
43	uarizona	ucdavis	boston	ucdavis	ucsb	wustl
44	ucolorado	wustl	wustl	boston	wustl	dartmouth
45	utah	boston	ucsb	wustl	dartmouth	ucsb
46	vatech	dartmouth	ucdavis	dartmouth	boston	boston
47	wustl	ncstate	tamu	tamu	ncstate	ncstate
48	arizonastate	tamu	ncstate	ncstate	tamu	tamu
49	boston	arizonastate	arizonastate	arizonastate	arizonastate	arizonastate
50	ncstate	vatech	vatech	vatech	vatech	vatech

3.5.1.3 Recent Years vs Earlier Years

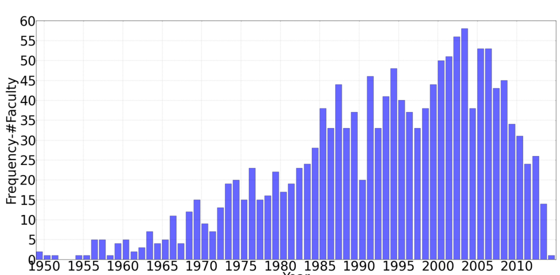
In this section we focus on comparing the recent data and the earlier data, expecting to discover some differences out of the comparison. As we described in 3.2.1, more than 80 percent of the entries have *Year Grad.* information. Thus, based on the entries with year information, we generated the distribution of year data in Top50 CS data set, which is shown in Figure 3.3. Figure 3.3a on the left is the frequency distribution of years, and Figure 3.3b on the right is the Cumulative Distribution Function (CDF) of year distribution.

Although the data set is quite "recent" since all the data we have collected appear on the web pages currently, we can see that there are a number of professors graduated decades ago. Some of them might still be active in academic fields and some of them might just be "emeritus faculties" that only hold the title and no longer active. In this case, we are interested to see what would happen if we only consider the data from recent years and what would be the differences compared with the entire data. As we can see in Figure 3.3b, the CDF curve crosses 50 percent between calendar year 1994 and 1995. In fact, before 1994 inclusively, there are 875 data points; after 1994 exclusively, there are 918 data points. The numbers are roughly equal and it would be fair to divide the data set by year 1994 into two equally large subsets to analyze the effect of year of graduation.

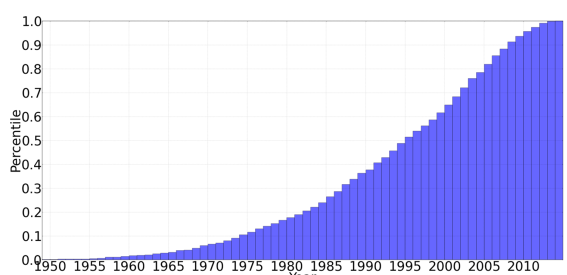
Table 3.6 shows the comparison between the results of recent years and earlier years. In Table 3.6, column 2 shows the resulting *RankDist* applied on the entire data set; column 3 shows the resulting *RankDist* applied on the data set from 1949 to 1994 inclusively; column 4 shows the resulting *RankDist* applied on the data set from 1995 to 2014 inclusively.

Interestingly, the *RankDist* values in column 4 are all smaller than the *RankDist* values in column 3. On average, the values in column 4 are 27% smaller than the values in column 3, indicating that the ranking results from recent 20 years are much closer to the U.S. News ranking than the results on years before 1995. Considering there is unavoidable noise in the data set, this improvement is quite significant. We expect that it is probably because some old CS programs such as Harvard and Yale would do better in the old days, while some new CS programs like Gatech and UCSD would boost up recently, letting the old CS programs going down in the ranking. We will investigate into such special cases later. In addition, compared with the results on the entire data set, the recent year seems to do a little bit worse but the *RankDistances* are pretty close. The reason that the result on recent 20 years could not do as good as the results on the entire data set is the amount of information. Data from the recent 20 years is only half of the entire data. It would be unfair to compare them since using the entire data has the advantage of having more data points.

Furthermore, in the experiments on recent 20 years (1995 $\sim$ 2014 inclusively), WeightedPR_wo_n is doing the best with *RankDist* 4.16, then comes IndeRank (4.28) and then HITS_Hubavg (4.64). In the earlier years data, WeightedPR_wo_n and HITS_Hubavg obtain the lowest *RankDists* of 6.04 and 6.12 respectively.



(a) Frequency Distribution



(b) CDF

Figure 3.3: Distributions of Years in Top50 CS Data Set

Table 3.6: Results between Recent Years and Earlier Years on CS Data Set

	RankDist with U.S. News on Extended graph w/o self-edges		
Algorithm	Entire Data	1949∼1994	1995∼2014
IndeRank	4.08	6.28	4.28
WeightedPR_w_n	4.72	6.44	4.68
WeightedPR_wo_n	3.92	6.04	4.2
HITS_Weighted	4.16	6.42	5.0
HITS_Hubavg	3.88	6.12	4.64

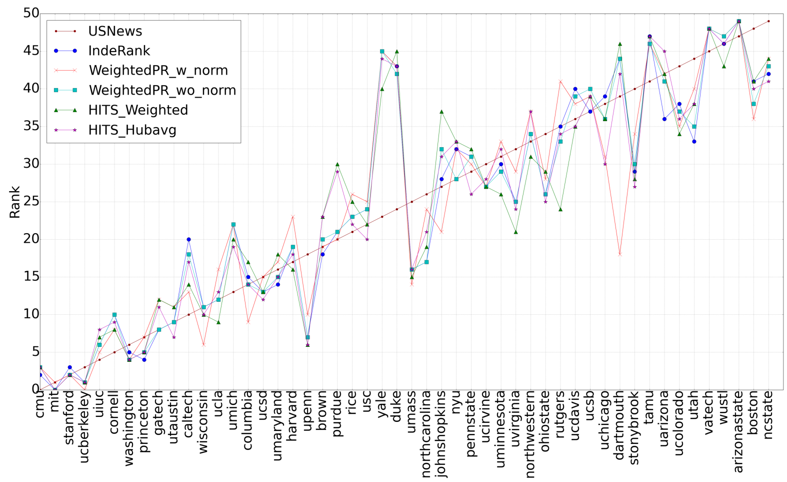


Figure 3.4: Ranking Divergence of CS Programs Compared to U.S. News (1995-2014)

Figure 3.4 shows the ranking divergence of all programs obtained from recent data. Yale and Duke is doing not as well as U.S. News estimates.

3.5.1.4 Observations

In order to know where the differences between U.S. News ranking and our rankings come from, we investigate into the actual rank changes for each program in our data set. Because the space is limited in this report and because WeightedPR_wo_n and HITS_Hubavg seems doing better than other algorithms according to our pre-

Table 3.7: Rank Difference Comparison on CS Data Set

	WeightedPR_wo_n				*HITS_Hubavg*			
Univ	Entire	'49∼'94	'95∼'14	AbsDif	Entire	'49∼'94	'95∼'14	AbsDif
Yale	+2	+12	-22	34	+2	+12	-21	33
NYU	+4	+12	0	12	+6	+16	-5	21
Purdue	+4	+8	-1	9	+1	+5	-9	14
Harvard	+10	+11	-2	13	+9	+12	-1	13
UCSD	-5	-19	+2	21	-5	-19	+3	22
Gatech	-5	-20	0	20	-6	-24	-3	21
Rice	-7	-16	-2	14	-6	-16	-1	15
Columbia	-5	-9	0	9	-4	-10	0	10
Utah	+13	+17	+9	8	+13	+18	+6	12
Duke	-16	-8	-18	10	-16	-9	-19	10
StonyBrook	+14	+20	+10	10	+15	+21	+13	8
Caltech	-4	-7	-8	1	-3	-6	-7	1
TAMU	-5	-9	0	9	-4	-6	-6	0
UIUC	0	0	-2	2	0	0	-4	4
Stanford	0	0	0	0	0	+1	0	1
UTAustin	0	+1	0	1	0	+1	+2	1
MIT	+1	+1	+1	0	+1	+1	+1	0

vious discussion, in this section, we only use these two algorithms and part of the entire programs to explain our observations.

Table 3.7 shows the exact difference for each program in WeightedPR_wo_n ranking and HITS_Hubavg ranking compared with U.S. News ranking. The positive value means the rank is higher than the rank in U.S. News; the negative value means the rank is lower than the rank in U.S. News. The $AbsDif$ value is simply the absolute difference between the values in '49 $\sim$ '94 and '95 $\sim$ '14. We can see that some of the programs get ranked dramatically different from their rank in U.S. News, such as StonyBrook, Harvard and Duke. It is these programs that enlarge the difference between the results from our algorithms and the U.S. News.

The first block, consisting of Yale, Purdue, Harvard and NYU, is comprising the programs that are doing much better before 1994 than they did after 1994. Part of the reason could be that they are old programs, who have establishing their academic strengths in the earlier days. Another reason could be that they fell behind in the recent years, for example, as we can see, Yale and Purdue are ranked much lower from recent data by our algorithms than U.S. News.

The second block, including Gatech, UCSD, Rice and Columbia, is comprising the programs that are doing much worse before 1994 than they did after 1994. This is probably because they are young programs and grew fast in the recent years.

The third block, includes those programs that are "under-estimated" or "over-estimated" by U.S. News. For example, StonyBrook and Utah are under ranked by U.S. News while Duke, Caltech and TAMU are over ranked by U.S. News.

The last block consists of those programs that are relatively stable in both our rankings and U.S. News ranking. The common characteristics of these programs, such as Stanford and UIUC are ranked roughly the same by both our algorithms and U.S. News. When seeing the whole scenario, such programs compose the majority of the programs, making the *RankDist* as low as about 3.88.

These observations are not coincident but all reflected from the hiring graph. Here is an example. Duke and UMass are equally ranked as No. 25 in U.S. News ranking. However, in the hiring graph, Duke is ranked lower than UMass. Figure 3.5a shows the neighbours of UMass in the hiring graph; Figure 3.5b shows the neighbors of Duke in the hiring graph. In Figure 3.5, the medium dark nodes are the target node we are looking at; the dark nodes are the nodes pointed by the target node; the light nodes are the nodes pointing to the target node. Hence, the target node is hiring PhDs from dark nodes; the light nodes are hiring PhDs from our target node.

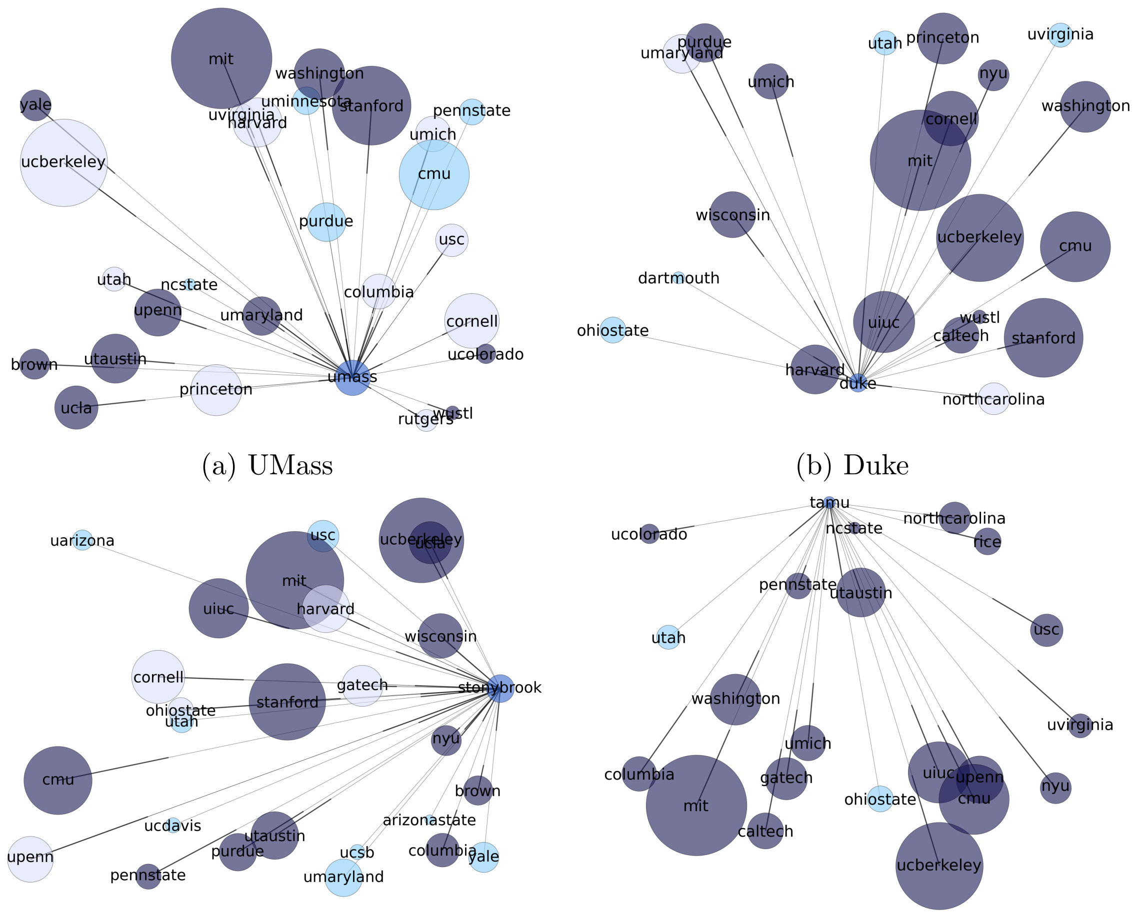
We can see in 3.5a, CMU, UCBerkeley, Princeton, Cornell, Harvard, Purdue and some other schools have hired PhD graduates from UMass. On the other hand, even though ranked similarly by the U.S. News, Duke is performing much worse compared with UMass in terms of hiring graph. As we can see in 3.5b, only Utah, UVirginia, UMaryland, Dartmouth, NorthCarolina and OhioState have hired PhDs from Duke. Since these programs are not as highly ranked as the programs that hired UMass PhDs, UMass gets ranked higher in our approach.

Here is another example. In the U.S. News ranking, StonyBrook and TAMU are equally ranked as No. 40. However, they get ranked differently in our approach. The PhDs from StonyBrook went to Cornell, Harvard, Gatech, UPenn, UMaryland, UCSB, Yale, UCDavis and so on, while the PhDs from TAMU went to Utah and OhioState. We can see a gap between the qualities of these two sets, and StonyBrook gains more credits from the higher quality programs hiring its graduates. This is why StonyBrook is ranked about 15 ranks higher than the U.S. News by our approach. We can also confirm this observation from the third block in Table 3.7.

Another interesting observation is that some schools are doing better in the earlier days while doing not that well in the recent 20 years. Harvard and Yale seem to be two typical examples of such programs. Table 3.8 shows the incoming edges of Harvard with year and Table 3.9 shows the incoming edges of Yale with year. We can see that Harvard's PhDs got hired widely among Universities before 1994 while only a few of them got hired in the recent 20 years. Yale has 15 incoming edges before 1994, while only 2 after 1994. Since these schools did not place as many of their graduates as faculty in recent years, their rankings fall by a substantial number when we look at recent data.

3.5.1.5 Sensitivity Analysis

Our expectation is that, one or two faculties coming or leaving the department should not affect the rank of the department dramatically. Any change in rankings from such small changes in hiring graph is considered to provide an idea of fidelity of rankings.



(a) UMass (b) Duke

(c) StonyBrook (d) TAMU

Figure 3.5: One-level Neighbouring Graphs in CS Data Set

Thus we proposed measuring the upper bound and lower bound of the rank for each program under the circumstance when there is a minor change in the hiring graph. To measure the upper bound of a program's rank, we accordingly add a "*Virtual*" edge from the # 1 program (e.g., *MIT* in CS data) to that program, which means that MIT just hired a PhD from that program; if there is already an edge from MIT to that program, we will simply increase the edge weight by 1. To measure the lower bound of a program's rank, we accordingly delete an existing edge from highest ranked program to that program. If the target edge has a weight

Table 3.8: Incoming Neighbours of Harvard in CS Data Set

Harvard's Incoming Nodes					
Univ.	Year	Univ.	Year	Univ.	Year
NYU	1950	UMaryland	1970	Caltech	1980
NorthCarolina	1956	Duke	1970	Cornell	1981
UCBerkeley	1959	NYU	1970	MIT	1984
UCLA	1963	MIT	1972	UMaryland	1985
Purdue	1963	Princeton	1973	Dartmouth	1986
Yale	1965	Harvard	1974	Gatech	1989
UMass	1966	UArizona	1974	UPenn	1989
NorthCarolina	1967	StonyBrook	1976	Boston	1992
UCDavis	1967	Duke	1977	CMU	1993
Yale	1968	Wustl	1978	Columbia	1993
USC	1969	Stanford	1980	UPenn	1993
UIUC	1995	StonyBrook	2003	Duke	2008
UCLA	1996	Boston	2003	Northwestern	2012
Cornell	1997	ArizonaState	2005	ArizonaState	2012
StonyBrook	1998	Harvard	2007		

more than 1, we will decrease the edge weight by 1; if the target edge has a weight exactly as 1, we will remove the edge from the graph. The reason we perform these two manipulations is that we have already seen that the quality and quantity of incoming edges play an essential role in the ranking of programs. We expect that these experiments would provide an idea of the sensitivity of our rankings.

As a result, Figure 3.6 shows the sensitivity bound of each program by all our five algorithms. From 3.6a to 3.6e, the x axis are the programs order by the rank from top to bottom; the y axis are the ranks. In these figures, each program has a bar that represents its sensitivity variation bound. The bottom of the bar means the upper bound that how high it could be ranked when adding a virtual significant edge; the top of the bar means the lower bound that how low it could be ranked when deleting a significant edge of the program. Thus, the narrower the variation

Table 3.9: Incoming Neighbours of Yale in CS Data Set

Yale's Incoming Nodes					
Univ.	Year	Univ.	Year	Univ.	Year
Dartmouth	1975	UCLA	1982	Northwestern	1986
UMass	1977	UMaryland	1982	USC	1987
CMU	1979	Princeton	1986	NYU	1988
Princeton	1980	Northwestern	1986	UPenn	1994
NYU	1980	Northwestern	1986	Rutgers	1994
Cornell	2005				

bound is, the less sensitive that program's ranking is to minor changes in the hiring graph.

In Figure 3.6a, IndeRank is doing well among the top 25 schools, while exhibiting significant variation below the top 25. This is because the IndeRank only cares about the number of incoming edges for a particular program, and many programs below top 25 have similar number of incoming edges. Hence, here comes the greatest disadvantage of IndeRank, which is that IndeRank is not able to rank those programs with the same number of incoming edges, even though the qualities of these edges vary. In other words, IndeRank only care about the quantity of edges but not the quality of edges. This seems to indicate that IndeRank leads to wide fluctuation in rankings for schools from 25 to 50 with minor changes.

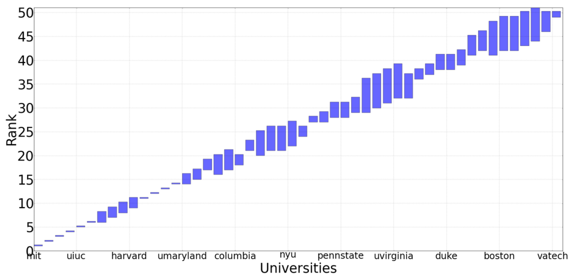
In Figure 3.6b, WeightedPR_w_n is not doing well either. Especially, the upper bounds for the lower programs are extremely wide, which means that adding an edge from MIT to a program significantly boosts the rank of that program. This happens because of two reasons. First of all, adding a high quality incoming edge suddenly become the major contribution of that program since the the program does not have many incoming edges. Secondly, the nature that PageRank only care about the *authority*, brings up the *authority* of that program instantly when adding an edge to that program pointed by another well established *authority*. Thus, WeightedPR_w_n seems to be very sensitive to potentially small changes in the hiring graph.

The performance of the other three algorithms are fairly similar in terms of the sensitivity graphs. One interesting thing is that WeightedPR_wo_n does not have the problem of WeightedPR_w_n, probably because it does not normalize the influence from incoming edges. In addition, HITS-based algorithms are very robust to minor changes. Another interesting thing is that, we can observe a "step-like" shape in 3.6c, 3.6d and 3.6e, indicating some programs share either upper bound or lower bound or both of them. It is a clear indicator that these programs might be about even and difficult to say which is a better one and hence should be ranked together.

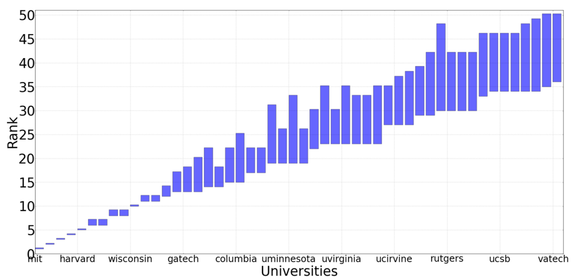
Table 3.10 summarizes the average sensitivity bounds for all the algorithms. The *UpperBound* indicates the average boost-up we can achieve for each algorithm; the *LowerBound* indicates the average degradation we get for each algorithm; the *Abs.Range* is simply the absolute difference between *UpperBound* and *LowerBound*. The *UpperBound* of WeightedPR_w_n (5.76) is extremely high, which is consistent with our analysis on the sensitivity graph. Based on these results in Table 3.10, Weighted_wo_norm, HITS_Weighted and HITS_Hubavg seem to offer a better distinction of programs.

Table 3.10: Average Sensitivity Bounds of all algorithms on CS Data Set

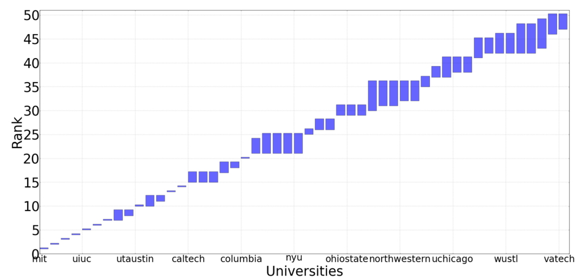
	IndeRank	*WeightedPR _w_norm*	*WeightedPR _wo_norm*	*HITS _Weighted*	*HITS _Hubavg*
UpperBound	+1.54	5.76	+1.54	1.6	1.4
LowerBound	-1.54	-1.98	-0.92	-1.24	-1.16
Abs.Range	3.08	7.74	2.46	2.84	2.56



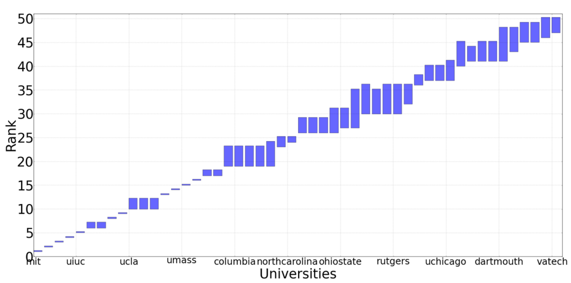
(a) IndeRank



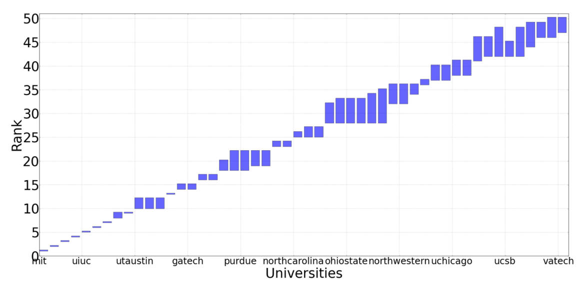
(b) WeightedPR_w_n



(c) WeightedPR_wo_n



(d) HITS_Weighted



(e) HITS_Hubavg

Figure 3.6: Sensitivity Graphs on CS Data Set

3.5.1.6 Discussion

Up until now, we have seen that our proposed method provides a new way of ranking programs in CS data set. HITS_Hubavg and WeightPR_wo_norm are doing the best in terms of both *RankDist* when comparing to U.S. News ranking and sensitivity. Even though IndeRank achieves fairly good *RankDist*, the nature of its disadvantage does not convince us that it is a good way to rank graduate programs. In addition, we also observed a large variation of the upper bound from WeightedPR_w_n, making it less robust compared with HITS_Hubavg, WeightPR_wo_norm and HITS.

Our analysis resulted in some programs being very differently ranked from U.S. News. Notably, Harvard and Yale do not do as well in our rankings and Stonybrook and Minnesota do significantly better in our rankings. Our analysis also showed that some programs have seen significant change in the last 20 years; Gatech and UCSD have considerably improved while Purdue and Yale have significantly decreased in placing their PhDs in academic.

3.5.2 Top50 ME

We have shown that our approach works pretty well in Top50 CS data set. More importantly, if robust enough, our approach should not only work for CS graduate programs but also universally for other graduate programs. In order to validate our methodology, we collected another separate data set, which is the faulty profile data from Top 50 ME programs in the USA, and then run our algorithms. If our algorithms are effective, we should be able to obtain similar results as we did in the CS data set. Although it is not sufficient to prove that our approach could be "universally" applied, we could still conclude that our approach is not only suitable for ranking CS program but also suitable for other programs.

3.5.2.1 Original Rankings

Similar to what we did in the CS data set, we first examine how our approach performed in *subtracted graph with self-edges*, *subtracted graph without self-edges*, *extended graph with self-edge* and *extended graph without self-edges*. Table 3.11 shows the comparisons among these cases.

As we can see in Table 3.11, results obtained from *extended graph without self-edges* are the best among the four. This is a similar observation as in CS data set. The best *RankDist* we achieved is from HITS_Weighted, which is 4.48. HITS_Hubavg (4.8), IndeRank (4.88) and WeightedPR_wo_n (5.0) also yield rankings pretty close to U.S. News ranking. Even though we observed that, in the case of HITS_Weighted and HITS_Hubavg, *RankDists* from graph with self-edges are smaller those from graph without self-edges, averagely the result from *extended graph without self-edges* is still the smallest. One thing we noticed is that the *RankDist* in ME data set is slightly larger than that in CS Data Set. This is probably because there are more noises in ME programs, in which case some ME programs hire PhD from other fields,

Table 3.11: Results on Top50 ME Data Set

	RankDist to the U.S. News Ranking				
Algorithm	Subtracted graph with self-edges	Subtracted graph w/o self-edges	Extended graph with self-edges	Extended graph w/o self-edges	**Average**
Max.	25.0	25.0	25.0	25.0	25.0
RandomShuffle	16.66	16.66	16.66	16.66	16.66
IndeRank	5.36	4.88	5.52	4.96	**5.18**
WeightedPR_w_n	6.84	6.8	6.6	6.04	**6.57**
WeightedPR_wo_n	6.08	5.52	5.08	5.0	**5.42**
HITS_Weighted	4.48	5.12	4.48	5.12	**4.8**
HITS_Hubavg	4.84	5.04	4.8	4.84	**4.88**
Average	**5.52**	**5.472**	**5.296**	**5.192**	—

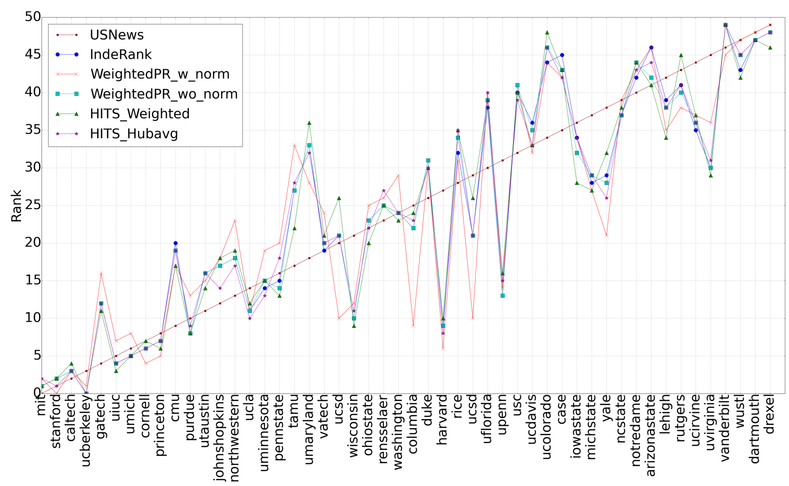


Figure 3.7: Ranking Divergence of ME Programs Compared to U.S. News

such as Aerospace, Material, Civil Engineering and CS and so on.

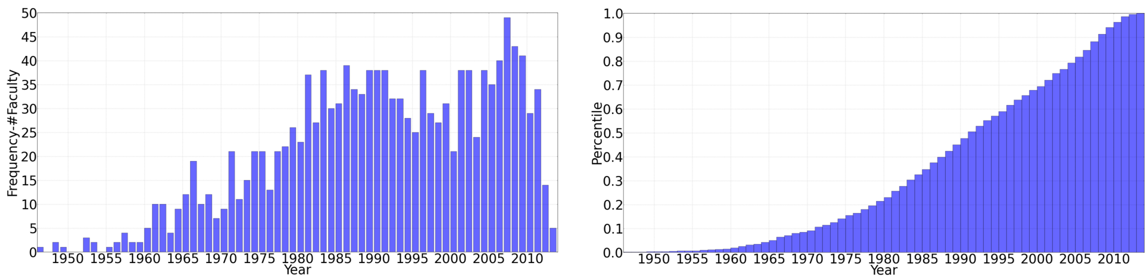
Table 3.12 and Table 3.13 combined show the original rankings of our algorithms obtained from the *extended graph without self-edges* of the entire data set, with the U.S. News ranking. Figure 3.7 shows the ranking divergence of each program between our algorithms and U.S. News. We can see that, some programs, such as UMaryland, Harvard, UPenn and UVirginia, are ranked dramatically different from the U.S. News. We will discuss some of these cases in Section 3.5.2.3.

Table 3.12: Results on Top50 ME Data Set (1∼25)

The rankings are retrieved from experiments on the entire ***Extended Graph with Self-edge Removed***						
Rank	USNews	IndeRank	WeightedPR_w_n	WeightedPR_wo_n	HITS_Weighted	HITS_Hubavg
1	mit	ucberkeley	stanford	ucberkeley	ucberkeley	ucberkeley
2	stanford	mit	ucberkeley	mit	mit	stanford
3	caltech	stanford	mit	stanford	stanford	mit
4	ucberkeley	caltech	caltech	caltech	uiuc	caltech
5	gatech	uiuc	cornell	uiuc	caltech	uiuc
6	uiuc	umich	princeton	umich	umich	umich
7	umich	cornell	harvard	cornell	princeton	cornell
8	cornell	princeton	uiuc	princeton	cornell	princeton
9	princeton	purdue	umich	purdue	purdue	harvard
10	cmu	harvard	columbia	harvard	wisconsin	purdue
11	purdue	wisconsin	ucsd	wisconsin	harvard	ucla
12	utaustin	ucla	ucla	ucla	gatech	wisconsin
13	johnshopkins	gatech	wisconsin	gatech	ucla	gatech
14	northwestern	upenn	purdue	upenn	pennstate	uminnesota
15	ucla	uminnesota	upenn	pennstate	utaustin	johnshopkins
16	uminnesota	pennstate	utaustin	uminnesota	uminnesota	upenn
17	pennstate	utaustin	gatech	utaustin	upenn	utaustin
18	tamu	johnshopkins	cmu	johnshopkins	cmu	northwestern
19	umaryland	northwestern	johnshopkins	northwestern	johnshopkins	pennstate
20	vatech	vatech	uminnesota	cmu	northwestern	cmu
21	ucsd	cmu	pennstate	vatech	ohiostate	vatech
22	wisconsin	ucsd	yale	ucsd	vatech	ucsd
23	ohiostate	columbia	ucsb	columbia	tamu	ohiostate
24	rensselaer	ohiostate	northwestern	ohiostate	washington	columbia
25	washington	washington	vatech	washington	columbia	washington

Table 3.13: Results on Top50 ME Data Set (26∼50)

The rankings are retrieved from experiments on the entire ***Extended Graph with Self-edge Removed***						
Rank	USNews	IndeRank	WeightedPR_w_n	WeightedPR_wo_n	HITS_Weighted	HITS_Hubavg
26	columbia	rensselaer	ohiostate	rensselaer	rensselaer	ucsb
27	duke	ucsb	rensselaer	ucsb	ucsd	yale
28	harvard	tamu	michstate	tamu	michstate	rensselaer
29	rice	michstate	umaryland	yale	iowastate	tamu
30	ucsd	yale	washington	michstate	uvirginia	michstate
31	uflorida	uvirginia	duke	uvirginia	duke	duke
32	upenn	duke	rice	duke	ucsb	uvirginia
33	usc	rice	ucdavis	iowastate	yale	umaryland
34	ucdavis	umaryland	tamu	umaryland	ucdavis	ucdavis
35	ucolorado	iowastate	iowastate	rice	lehigh	iowastate
36	case	ucirvine	lehigh	ucdavis	rice	rice
37	iowastate	ucdavis	uvirginia	ucirvine	umaryland	ucirvine
38	michstate	ncstate	ucirvine	ncstate	ucirvine	ncstate
39	yale	uflorida	rutgers	lehigh	ncstate	lehigh
40	ncstate	lehigh	ncstate	uflorida	uflorida	usc
41	notredame	usc	uflorida	rutgers	usc	uflorida
42	arizonastate	rutgers	usc	usc	arizonastate	rutgers
43	lehigh	notredame	case	arizonastate	wustl	case
44	rutgers	wustl	notredame	case	case	notredame
45	ucirvine	ucolorado	ucolorado	notredame	notredame	arizonastate
46	uvirginia	case	vanderbilt	wustl	rutgers	wustl
47	vanderbilt	arizonastate	arizonastate	ucolorado	drexel	ucolorado
48	wustl	dartmouth	wustl	dartmouth	dartmouth	dartmouth
49	dartmouth	drexel	dartmouth	drexel	ucolorado	drexel
50	drexel	vanderbilt	drexel	vanderbilt	vanderbilt	vanderbilt



(a) Frequency Distribution

(b) CDF

Figure 3.8: Distributions of Years in Top50 ME Data Set

3.5.2.2 Recent Years vs Earlier Years

For top50 ME data set, we also compared the case between earlier years and recent years. Figure 3.8 shows the distributions of year data in our top50 ME data set. Figure 3.8a on the left is the frequency distribution of years, and Figure 3.8b on the right is the Cumulative Distribution Function (CDF) of year distribution.

In ME data set, the earliest year is 1946 and the latest year is 2013. As we can see from 3.8b, the CDF curve crosses 50 percent between calendar year 1990 and 1991. In fact, up to 1990, there are 814 data points; after 1990, there are 896 data points. The numbers are roughly equal and it would be fair to divide the data set by year 1990 into two equally large subsets to analyze the effect of year.

Table 3.14 shows the comparsion between the results from the recent and earlier years data. In Table 3.14, column 2 shows the resulting $RankDist$ applied on the entire data set; column 3 shows the resulting $RankDist$ applied on the earlier data set (1946 $\sim$ 1990); column 4 shows the resulting $RankDist$ applied on the later data set from (1991 $\sim$ 2013).

We can see clearly from Table 3.14 that the result is consistent with the result obtained from CS data set. The ranks obtained from the year between 1991 and 2013

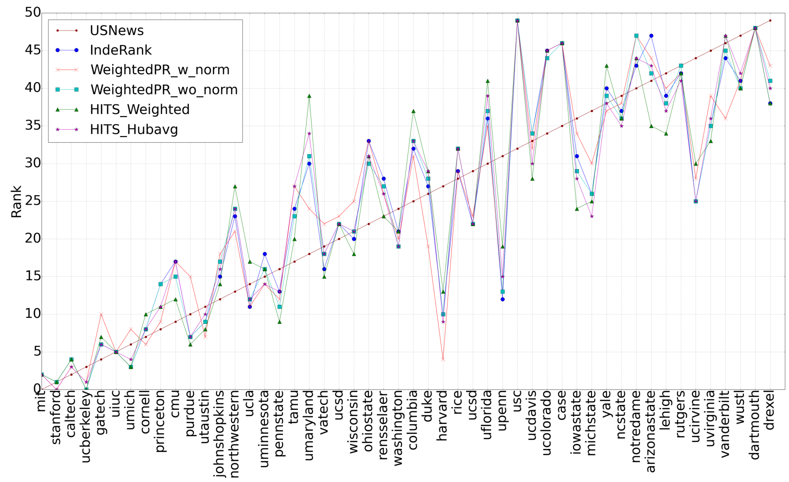


Figure 3.9: Ranking Divergence of ME Programs Compared to U.S. News (1991-2013)

are generally closer to the U.S. News than the ranks obtained from the years before 1991. The best case on recent data falls on WeightedPR_wo_n, with a *RankDist* of 5.52, followed by IndeRank (5.6), WeightedPR_w_n (5.6) and HITS_Hubavg (5.68).

3.5.2.3 Observations

In ME data set, we still observed some interesting differences in rankings when our approach is compared with U.S. News. Table 3.15 shows the exact difference for each program in WeightedPR_wo_n ranking and HITS_Hubavg ranking compared

Table 3.14: Results between Recent Years and Earlier Years on ME Data Set

	RankDist with U.S. News on Extended graph w/o self-edges		
Algorithm	Entire Data	1946∼1990	1991∼2013
IndeRank	4.96	7.12	5.6
WeightedPR_w_n	6.04	7.84	5.6
WeightedPR_wo_n	5.0	7.4	5.52
HITS_Weighted	5.12	7.76	6.0
HITS_Hubavg	4.84	7.36	5.68

Table 3.15: Rank Difference Comparison on ME Data Set

	WeightedPR_wo_n				*HITS_Hubavg*			
Univ	Entire	'49∼'94	'95∼'14	AbsDif	Entire	'49∼'94	'95∼'14	AbsDif
CMU	-10	-12	-6	6	-10	-10	-8	2
TAMU	-10	-21	-6	15	-11	-17	-10	7
UMaryland	-15	-24	-13	11	-14	-23	-16	7
Rice	-6	-5	-4	1	-7	-3	-4	1
Harvard	18	17	17	0	19	17	18	1
UCSB	3	-19	9	28	4	-19	9	28
Gatech	-8	-22	-2	20	-8	-26	-2	24
Columbia	3	11	-8	19	2	8	-8	16
PennState	2	-7	5	12	-2	-12	3	15
Wisconsin	11	13	0	13	10	13	0	13
UTAustin	-5	-14	2	16	-5	-11	1	12
Yale	10	23	-1	24	12	23	0	23

with U.S. News ranking. The positive value means the rank is higher than the rank in U.S. News; the negative value means the rank is lower than the rank in U.S. News. The $AbsDif$ value is simply the absolute between the values in '49 $\sim$ '94 and '95 $\sim$ '14.

The first block, including CMU TAMU, Rice, UMaryland and Harvard, is comprising those programs that are either under-estimated or over-estimated by the U.S. News. For example, Rice, UMaryland, TAMU and CMU are over ranked by U.S. News, while Harvard is under ranked by U.S. News. The second block comprises those programs that are performing dramatically different before 1990 and after 1990. For example, Gatech, UCSB, PennState and UTAustin are probably young programs and they experienced a boost of rank during the year from 1991 to 2013. The other three, Columbia, Wisconsin and Yale are obviously experiencing a downfall and not many other Universities recognizing their strength in ME as they did in the old days.

3.5.2.4 Sensitivity Analysis

In order to measure the responsiveness of these algorithms to minor change in the hiring graph, we conducted sensitivity analysis on ME data set as we did on the CS data set. In this case, the virtual edge is chosen to be a virtual edge from UCBerkeley to a given program, since UCBerkely is ranked as No. 1 in most of the cases. Thus, the narrower the variation bound is, the less sensitive the algorithm is. All these five algorithms are doing well in ranking the top 20 programs, with little jitter.

Table 3.16 summarizes the average variation bound for each algorithm. As we can see, HITS_Weighted has the smallest variation bound as $Abs.Range = 1.88$, Then follows HITS_Hubavg ($Abs.Range = 2.12$) and WeightedPR_wo_n ($Abs.Range = 3.28$).

Table 3.16: Average Sensitivity Bounds of all algorithms on ME Data Set

	IndeRank	*WeightedPR _w_norm*	*WeightedPR _wo_norm*	*HITS _Weighted*	*HITS _Hubavg*
UpperBound	+2.44	+7.5	+1.86	0.46	0.8
LowerBound	-2.42	-2.34	-1.42	-1.42	-1.32
Abs.Range	4.86	9.84	3.28	1.88	2.12

3.5.2.5 Discussion

After seeing the result from another complete different data set—faculty data from top 50 ME programs in the USA, we show that our algorithms work well in both CS data set and ME data set. Most of the results and analysis are consistent in both data sets. What's more, in ME data set, HITS_Hubavg is performing the best in terms of *RankDist* and HITS_Weighted is performing the best in terms of sensitivity. It would be unfair to say that the other algorithm are doing not well since we do not have any perfect ground truth in our experiments.

More importantly, as we did in the CS data set, we found some valuable observations from our analysis.

4. CONCLUSION AND FUTURE WORK

In this thesis we presented two separate projects, both related to data mining and knowledge discovery.

In the first project—Tweeter Classification using Sentiment Analysis, we collected the recent 200 tweets in September 2013 for those politically active tweeters, and then labelled them as either "*democrats*" or "*republicans*". We automatically discovery distinguishing topics and used these topics as a feature vector to classify the tweeters. The result shows that our new methodology performed not much better than non-sentiment approach, reaching a classification accuracy around 64 percent. Then, with the help of social relationship graph information, we are able to boost the accuracy of adjusted sentiment model up to 85 percent. We concluded that the limitations of our sentiment model come from both Twitter data and existing sentiment analysis tools, which are not robust enough on complex social media data in Twitter. We also deploy Belief Propagation model to infer the political association of tweeters in the social graph, which achieves highest prediction accuracy among all the models we have.

The future work of the first project would mainly focus on improving the quality of the data. If possible, the sentiment classification model should be tested in "firehore" data instead of streaming data, to see if that would make a difference. In addition, sentiment analysis model should be improved considering the complexity of Twitter data.

In the second project—Algorithmic University Program Ranking, we propose a new and alternative way to rank graduate programs using algorithms. We have shown that our approach reasonable and reliable rankings for graduate programs.

In addition, our approach works in both CS data set and ME data set, indicating that our approach is capable across fields. Among all our five algorithms, on average, WeightedPR_wo_norm, HITS_Hubavg and HITS_Weighted seems doing well in terms of both *RankDist* to U.S. News and sensitivity. A reasonable rank for graduate programs might probably be the average of the four algorithms—WeightedPR_w_n, WeightedPR_wo_n, HITS_Hubavg and HITS_Weighted, because each algorithm has its own advantage.

Moreover, we observe lots of interesting patterns and facts from our data. By extensive data analysis, we not only discover what is behind the "*hiring graph*" but also reveal valuable knowledge beyond U.S. News ranking.

The future work of this project will move a further step based on what we have currently, which is to construct a model for "cross-domain" university ranking. Given the fact that some programs hire Ph.D.s from other fields. For example, an Mechanical Engineering program might hire a Computer Science Ph.D. specializing in Robotics. We believe that the "cross-domain" effect across fields also matters in the *hiring graph*. It would be interesting to come up with an algorithm able to rank each universities in multiple programs at once.

REFERENCES

[1] Twitter Streaming API: `https://dev.twitter.com/docs/api/streaming`. Retrieved on May 05 2014.

[2] Twitter REST API: `https://dev.twitter.com/docs/api`. Retrieved on May 20 2014.

[3] A. A. Amleshwaram, N. Reddy, S. Yadav, G. Gu and C. Yang. *CATS: Characterizing Automation of Twitter Spammers*. Bangalore, India: Communication Systems and Networks (COMSNETS), 2013, pp. 1-10.

[4] C. Yang, R. Harkreader, J. Zhang, S. Shin and G. Gu. *Analyzing Spammers Social Networks for Fun and Profit: A Case Study of Cyber Criminal Ecosystem on Twitter*. New York, NY, USA: Proceedings of the 21st international conference on World Wide Web, 2012, pp. 71-80.

[5] B. Pang and L. Lee. *Opinion Mining and Sentiment Analysis*. Hanover, MA, USA: Journal, Foundations and Trends in Information Retrieval, 2008. Volume 2 Issue 1-2, pp. 1-135.

[6] P. T. Metaxas and E. Mustafaraj. *Social Media and the Elections*. Journal, SCIENCE, 26 October 2012. Volume 338, pp. 472-473.

[7] H. Wang, D. Can, A. Kazemzadeh, F. Bar and S. Narayanan. *A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle*. Stroudsburg, PA, USA: Proceedings of the ACL 2012 System Demonstrations, 2012, pp. 115-120.

[8] M Choy, M. L. F. Cheong, M. N. Laik and K. P. Shung. *A Sentiment Analysis of Singapore Presidential Election 2011 Using Twitter Data with Census Correction.*

Journal of Information Technology & Politics, 2012.

[9] A. Tumasjan, T. O. Sprenger, P. G. Sandner and I. M. Welpe. *Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment*. In Fourth International AAAI Conference on Weblogs and Social Media, 2010.

[10] *Get_ friend* API call in Twitter: `https://dev.twitter.com/docs/api/1/get/friends/ids`. Retrieved on May 21 2014.

[11] *Get_ follower* API call in Twitter: `https://dev.twitter.com/docs/api/1/get/followers/ids`. Retrieved on May 21 2014.

[12] A. Pak and P. Paroubek. *Twitter as A Corpus for Sentiment Analysis and Opinion Mining*. Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC10), European Language Resources Association (ELRA), Valletta, Malta (May 2010), pp. 1921.

[13] Alchemy API: `http://www.alchemyapi.com/`. Retrieved on May 21 2014.

[14] S. Baccianella, A. Esuli, and F. Sebastiani. *SentiWordNet3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining*. In Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC10), Valletta. pp. 22002204.

[15] B. Liu, M. Hu and J. Cheng. *Opinion Observer: Analyzing and Comparing Opinions on the Web*. Proceedings of the 14th International World Wide Web conference (WWW-2005), May 10-14, 2005, Chiba, Japan.

[16] J. Pearl. *Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach*. Proceedings of the Second National Conference on Artificial Intelligence. AAAI-82: Pittsburgh, PA. Menlo Park, California: AAAI Press. pp. 133136.

[17] L. Page, S. Brin, R. Motwani and T. Winograd. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report, Stanford University, 1999.

[18] How US News calculated the 2015 best graduate school rankings: `http://www.usnews.com/education/best-graduate-schools/articles/2014/03/10/how-us-news-calculated-the-2015-best-graduate-schools-rankings`. Retrieved on May 13 2014.

[19] R. Lukman, D. Krajnc and P. Glavic. *University Ranking Using Research, Educational and Environmental Indicators.* Journal of Cleaner Production, Volume 18, Issue 7, May 2010, pp. 619-628.

[20] J. C. Shin, R. K. Toutkoushian and U. Teichler. *University Rankings: Theoretical Basis, Methodology and Impacts on Global Higher Education.* 2011: Springer Science+Business Media.

[21] L. Leydesdorff and J. C. Shin. *How to Evaluate Universities in Terms of Their Relative Citation Impacts: Fractional Counting of Citations and the Normalization of Differences Among Disciplines.* Journal of the American Society for Information Science and Technology. Volume 62, Issue 6, 2011, pp. 1146-1155.

[22] J. M. Kleinberg. *Authoritative Sources in a Hyperlinked Environment.* New York, NY, USA. Journal of the ACM, Volume 46 Issue 5, Sept. 1999, pp. 604-632.

[23] Computer Science ranking in US News: `http://grad-schools.usnews.rankingsandreviews.com/best-graduate-schools/top-science-schools/computer-science-rankings`. Retrieved from March 23 2014.

[24] Mechanical Engineering ranking in US News: `http://grad-schools.usnews.rankingsandreviews.com/best-graduate-schools/`

top-engineering-schools/mechanical-engineering-rankings?int=997808. Retrieved in Apr 25 2014.

[25] A. Borodin, G. O. Roberts, J. S. Rosenthal and P. Tsaparas. *Link Analysis Ranking: Algorithms, Theory, and Experiments*. New York, NY, USA. Journal of ACM Transactions on Internet Technology (TOIT), Volume 5 Issue 1, February 2005, pp. 231-297.

APPENDIX A

UNIVERSITY ABBREVIATIONS

Table A.1: Mapping between Universities and their Abbreviations in this thesis—1

Abbreviation(s)	University
arizonastate/ArizonaState	Arizona State University
boston/Boston	Boston University
brown/Brown	Brown University
caltech/Caltech	California Institute of Technology
case/Case	Case Western Reserve University
cmu/CMU	Carnegie Mellon University
columbia/Columbia	Columbia University
cornell/Cornell	Cornell University
dartmouth/Dartmouth	Dartmouth College
drexel/Drexel	Drexel University
duke/Duke	Duke University
gatech/Gatech	Georgia Institute of Technology
harvard/Harvard	Harvard University
iowastate/IowaState	Iowa State University
johnshopkins/JohnsHopkins	Johns Hopkins University
lehigh/Lehigh	Lehigh University
michstate/MichState	Michigan State University
mit/MIT	Massachusetts Institute of Technology
ncstate/NCState	North Carolina State University
northcarolina/NorthCarolina	University of North Carolina at Chapel Hill
northwestern/Northwestern	Northwestern University
notredame/NotreDame	University of Notre Dame
nyu/NYU	New York University
ohiostate/OhioState	Ohio State University
pennstate/PennState	Pennsylvania State University
princeton/Princeton	Princeton University
purdue/Purdue	Purdue University
rensselaer/Rensselaer	Rensselaer Polytechnic Institute
rice/Rice	Rice University
rutgers/Rutgers	Rutgers University

Table A.2: Mapping between Universities and their Abbreviations in this thesis—2

Abbreviation(s)	University
stanford/Stanford	Stanford University
stonybrook/StonyBrook	State University of New York at Stony Brook
tamu/TAMU	Texas A& M University
uarizona/UArizona	University of Arizona
ucberkeley/UCBerkeley	University of California, Berkeley
ucdavis/UCDavis	University of California, Davis
uchicago/UChicago	University of Chicago
ucirvine/UCIrvine	University of California, Irvine
ucla/UCLA	University of California, Los Angeles
ucolorado/UColorado	University of Colorado
ucsb/UCSB	University of California, Santa Barbara
ucsd/UCSD	University of California, San Diego
uflorida/UFlorida	University of Florida
uiuc/UIUC	University of Illinois at Urbana-Champaign
umaryland/UMaryland	University of Maryland
umass/UMass	University of Massachusetts Amherst
umich/UMich	University of Michigan
uminnesota/UMinnesota	University of Minnesota
upenn/UPenn	University of Pennsylvania
usc/USC	University of Southern California
utah/Utah	University of Utah
utaustin/UTAustin	University of Texas at Austin
uvirginia/UVirginia	University of Virginia
vanderbilt/Vanderbilt	Vanderbilt University
vatech/Vatech	Virginia Polytechnic Institute
washington/Washington	University of Washington
wisconsin/Wisconsin	University of Wisconsin-Madison
wustl/Wustl	Washington University in St. Louis
yale/Yale	Yale University