Classification of tweets into policy agenda topics

by

Rihui Li

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Johnny S. Wong, Major Professor
Wallapak Tavanapong
David Peterson

Iowa State University

Ames, Iowa

2016

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Page

ACKNOWLEDGMENTS

First and foremost, praises and thanks to God, the Almighty, for being my strength and guidance throughout my research work. Only due to his blessings I could complete the research and finish my thesis successfully.

I would like to thank my major professor, Dr. Johnny Wong for his guidance and support throughout the course of this research. I value his inputs and patience he has shown with me during this study.

Thank you to my POS committee member Dr. Wallapak Tavanapong for her precious ideas on how to move this research forward. Thanks for the time you help me improve my presentation skills and revise my defense slides and thesis.

Thank you to my POS committee member Dr. David Peterson for his valuable input in the political science domain. Your great help matters on formulating the ground truth and guiding the manual topics classification.

I would like to give my special thanks to my godly wife Jingfang Tang, who is definitely a blessed gift from God. Thank you for your continuous prayer for me. Thank you for all your support, encouragement, and quiet patience in my life.

ABSTRACT

Twitter is a novel online microblogging service launched in July 2006. This service has been rapidly gaining worldwide popularity. It has more than 500 million users, out of which there are more than 332 million active users in May 2015. Twitter website is one of the ten most visited websites and has been described as “the SMS of the Internet.” It is not only widely used in a person’s daily life, but also in politics, such as running election campaigns, mining or influencing public opinions.

We study the problem of automated classification of tweets posted on official accounts by a state’s senate and a state’s House of Representatives as well as accounts by individual senators and house representatives, to one of the 21 policy agenda topics specified by Policy Agenda Project [2]. This problem is a multi-class classification problem for short text since each tweet has a limit of 140 characters. Compared with traditional text classification, short text classification has a special characteristic that the content is short and sparse. Therefore, it is very challenging to extract a useful feature for classification. To achieve a reasonable performance, we investigate three methods including Support Vector Machine (SVM) with Linear Kernel, SVM with Topics Grouping and Tweets Merging, and Convolutional Neural Networks (CNN). Based on the experimental results, the CNN method achieved the best performance of 77.34% accuracy on an independent testing set of 1,388 tweets in the Iowa dataset. Furthermore, the CNN method is robust and stable without the need to manually tune the hyper-parameters, which are the attributes of the neural network such as the number of hidden layers, the number of units per layer, and the connections per unit.

CHAPTER 1

INTRODUCTION

Twitter is a hot online microblogging service [1], which has been widely used by politicians [2-3]. They use Twitter to mine or guide the public opinion or broadcast their political agendas. They can also know users' political alignment by sentiment analysis of the users' tweets [4]. In 2016, Hillary Clinton, a candidate for the Democratic presidential nomination, launched a political fundraiser in Twitter for her election campaign [5]. Senators and House representatives use Twitter to convey to the public about policy agendas under discussion in a state's Senate and House through official and individual accounts. These political tweets can be very useful for political science scholars to conduct research on government related issues.

Policy Agendas Project [6] defines 20 major topics and 220 subtopics and a codebook [7] describing them in detail along with guidelines and exceptions to manually classify text into one of these 20 topics as listed in Table 1.1. Topic 11 is missing from the list. In Congressional Bill Project [8], the classification has been used to manually assign congressional bills to different topics and subtopics. Automated classification of congressional bill titles according to these categories have been investigated [9-10].

Table 1.1. List of 20 Policy Agenda Topics

Topic Number	Topic Name
1	Macroeconomics
2	Civil Rights, Minority Issues, and Civil Liberties
3	Health
4	Agriculture
5	Labor and Employment
6	Education
7	Environment
8	Energy
9	Immigration
10	Transportation
12	Law, Crime, and Family Issues

Table 1.1. (Continued)

13	**Social Welfare**
14	**Community Development and Housing Issues**
15	**Banking, Finance, and Domestic Commerce**
16	**Defense**
17	**Space, Science, Technology and Communications**
18	**Foreign Trade**
19	**International Affairs and Foreign Aid**
20	**Government Operations**
21	**Public Lands and Water Management**

By classifying tweets into policy agenda topics, political science scholars can potentially discover what is under a discussion by senators and House representatives, how policy agendas are formed and conveyed to the public via Twitter, how the conveyed agendas compared to the agendas of the bills discussed in the Senate and the House**.** However, manually labeling each tweet for the corresponding topic as shown in Table 1.2 is labor intensive and time consuming for human coders as the number of tweets is very large and some tweets could be seen as belonging to more than one topic. In our study, we collected 308,601 tweets from 472 accounts of only 11 states. It took two political science students over 8 months to manually label part of this tweet collection for this study.

Table 1.2. Tweets and the manually assigned policy agenda topics

Tweet Body	Topic Name
Radio IA: Gov says FY2017 budget must be 'very frugal'Not surprising after his no-vote massive corp. tax cut! https://t.co/Cr5lGj4TqZ	**Macroeconomics**
Former Sen. Warnstad (SCJ): "Flag of rebellion deserves no public place of honor" http://t.co/PMV9mn9U3L #ialegis http://t.co/l4bihznbdc	**Civil Rights, Minority Issues, and Civil Liberties**
RT @mccoyforsenate: Rx drug & heroin overdose deaths could be prevented in IA by allowing Iowans to get meds that counteract overdoses.htt…	**Health**
McCoy: Senate will pass reforms this year to address the problems Branstad/Reynolds continues to ignore.	**Mixed Topics**

1.1 Problem Statement

Automated classification of a tweet into one of the above policy agenda topics has not been investigated in the literature. However, it has the potential to assist political scientists to learn about the formation of

state policy agendas by state legislatures and the strategies in which these agendas are communicated to the public. This classification problem is challenging due to short and informal text content of tweets. Furthermore, a tweet sometimes contains words appearing in more than one policy agenda topic. We call such a tweet an ambiguous tweet. In our study, we assign ambiguous tweets their own topic named "Mixed Topics." By adding this new topic, we have a total of 21 policy agenda topics in our study. The research problem is to design a 21 class classifier that automatically assigns a tweet into one of these classes effectively. This problem has many challenges. Firstly, tweets are short, making it difficult to extract useful features for classification. Secondly, tweets are very informal with hashtags and URLs. Finally, the problem is a 21-class classification problem, which is more challenging to achieve very high accuracy for each class compared to a binary class classification. In fact, how high the accuracy needs to be in order for the classifier to be practically useful.

1.2 Contributions

There are four major contributions in our study as follows.

(1) We extended an existing Java application to collect tweets from official State Senate and House Representatives and individual senator and representative accounts. The application uses Twitter REST API to retrieve tweets and store them into a relational database.

(2) This thesis is first to investigate the problem of classifying a tweet into one of the 21 policy agenda topics. The solution has its application for prediction of the top k policy agenda topics under discussion by a state's legislature in a given time period.

(3) We investigated three classification methods: Support Vector Machine with linear kernel (SVM) using Term Frequency weighted by Inverse Document Frequencies, Support Vector Machine with Topics Grouping and Tweets Merging, and Convolutional Neural Network (CNN) to find the method that gives the best accuracy. We found that the CNN method performs best among these approaches with 77% accuracy in an independent testing set containing 1388 tweets by Iowa Senate and House.

(4) We developed a new REST API for retrieving top *k* policy agendas per month per state and for finding states with similar policy agenda topics. With the new REST API, it is very convenient for user to retrieve the valuable information from the classification result, such as the hot topics discussed by a state's legislature in a given month in a given state and the similarity in topics discussed in a set of states in the past one year.

1.3 Organization

The rest of the thesis is organized as follows. Chapter 2 introduces background and related work in text classification and short text classification, tweet classification in Poli-informatics, Convolutional Neural Network for text classification, and policy agenda topics related study. Chapter 3 presents our proposed work for policy agenda topic classification. Chapter 4 shows the experimental design and results. Chapter 5 is our conclusion and the description of the future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

Our study is a multi-class short text classification for policy agenda topics in poli-informatics. In this chapter, we summarize related work in text/short text classification, classification in the domain of poli-informatics and classification for policy agendas.

2.1 Text classification vs Short text classification

Traditionally, text classification, also known as document classification, refers to the classification for an entire document or large passage. One advantage of such a classification is that the content is rich enough for feature extraction even after applying data pre-processing techniques, such as removing stop words and word stemming. Many machine learning algorithms such as Support Vector Machine (SVM) with linear kernel, Naive Bayes, Decision Tree, and K-Nearest Neighbor (KNN) have been effective for the text classification problems [11-19]. Most of text classification researches focus on feature extraction and the performance improvement of KNN and SVM with linear kernel since they perform better than other algorithms. The assumption for the KNN method is that nearby instances in the vector space should be in the same class. The authors in [12] propose a revised KNN approach to improve the performance to deal with the problem that class distribution in the training set is uneven. The revised algorithm uses different numbers of nearest neighbors for different classes, rather than a fixed number across all categories. The experiments on Chinese text categorization show that the revised algorithm is less sensitive to the value of parameter *k* than the traditional one, and it can properly classify documents belonging to smaller classes with a large value of *k*. The authors in [17] have a deep discussion on the popular text classification algorithms, the result shows SVM with linear kernel achieves substantial improvements over the currently best performing methods and behaves robustly over a variety of different learning tasks. A good result could be achieved fully automatically without any manual parameters tuning by SVM with linear kernel. Based on the investigation in this study, SVM with linear kernel can perform

well due to four properties of text: (1) high dimensional input space, (2) few irrelevant features, (3) sparse document vectors, and (4) mostly linearly separable classes.

Compared with traditional text classification, short text classification refers to the classification of a short sentence. The challenge is that the content is short and sparse, which makes it difficult for feature extraction. Most existing solutions enrich the representation of the original short text. The authors of [20] proposed two approaches. One method is to search the context of this short text on Internet by using a crawling program, and add the context to the short text. The other method is to derive latent topics from an existing large corpus. By using these two methods, the content has been enriched and feature extraction is on the context or corpus instead of the original short text. The authors of [21] proposed a method to map the short text to the concept in Wikipedia. Mapping enriches the representation of short text and content in Wikipedia could be used for feature extraction.

2.2 Convolutional neural network (CNN) for short text classification

A convolutional neural network [22-24] is a type of feed-forward artificial neural network that contains multiple layers, such as the convolutional layer, pooling layer, and Softmax layer. CNN has been applied on image recognition successfully in the past decade [25-27]. In recent years, it have been used for natural language processing and proved to be useful for short text classification [28]. Traditionally, to build a text classifier, we formulate some human-designed features based on dictionaries, domain knowledge, or some special tree kernels [29]. However, with CNNs, we do not need design these features manually since the best features will be determined automatically during the training process. The authors in [28] propose a sentence classification method with a convolutional neural network. It uses an open source tool named word2vec to convert a sentence to a matrix, of which its row is the vector representation of a word in the sentence. In convolutional layers, the convolutional filters will be selected automatically and applied on the matrix to generate feature maps, which are a set of features. The best features in each feature map will be selected to penultimate layer (or max-pooling layer) by the max-pooling technique. Then these best features will be passed to the Softmax layer and processed to represent

the probability of each class. Despite little tuning of hyper-parameters of CNN during the training phase, a simple CNN with one layer of convolution performs remarkably well (accuracy was reached 89.6% on Opinion polarity detection subtask of the MPQA dataset [30]). The authors in [31] introduce a recurrent convolutional neural network (RCNN) for text classification. In this model, when learning word representations, a recurrent structure is applied to capture contextual information, which will considerably reduce noise. The max-pooling layer judges which words play key roles in text classification to capture the key components in the text automatically. Unlike the traditional CNN method, it uses a bi-directional recurrent structure instead of a fixed window size for filtering. By using this structure, it is easy to capture the contextual information to the greatest extent possible when learning word representations. On the same datasets, RCNN performs better with 96.49% accuracy compared with traditional CNN with 94.79% accuracy.

2.3 Tweet classification in Poli-Informatics

Most of the studies involving tweets in the field of poli-informatics focus on the political sentiment analysis in Twitter [33-35]. One of the key steps is sentiment classification: the sentiment information is extracted from a tweet by some special tool and classified into three or more classes, such as very positive, positive, negative, very negative and neutral. The authors of [36] predicted election result based on sentiment analysis. They investigated four different machine learning methods for sentiment classification: Multinomial Naïve Bayes (MNB), Support Vector Machine (SVM), Adaboost MNB (ADA-MNB) [37], and Adaboost SVM (ADA-SVM). The ADA-MNB classifier performs the best and achieves 65.09% classification accuracy in 10-fold cross-validation for three classes. The authors in [38] predicted the 2011 Dutch Senate election results by Twitter analysis, using two steps to enhance the prediction result. These steps are to (1) improve the quality of the document collection by removing certain tweets that contain apparent Dutch words but are actually written in another language; (2) perform sentiment analysis on the refined document collection using a binary classifier for negative or non-negative sentiment towards the party mentioned in the tweet. The authors in [39] proposed a 3-class

sentiment classification after sarcastic tweets were removed from the dataset before they were individually labeled with one of the following categories: **pos** (postive), **neg** (negative), **mix** (including both positive and negative), **neu** (no sentiment towards the topic), **nen** (tweets not in English), and **non** (tweets without any mention or relation to the topic). Only the pos, neg, and mix categories are used for classification, which is a limitation of the research since the other three categories are discarded manually. Several approaches have been proposed for this problem. Among them, the distance based scoring method was shown to achieve the best performance with 58.96% accuracy. The authors in [40] predicted the political alignment of Twitter users based on a training dataset of 1,000 manually classified individuals. The political alignments are classified into three types: left, right, and ambiguous. Using SVM, the accuracy was reached 91%.

2.4 Related work in policy agenda topics

Kaul developed a binary classifier to determine whether a tweet conveys a political policy agenda or not [41]. The approach includes several key steps. The pre-processing step removes stop words and applies the k-stemming technique for each tweet. The feature extraction step extracts term frequency and four additional hand-crafted features: whether or not the tweet contains a hashtag (Hashashtag), whether the tweet contains a URL (HasURL), the number of characters in the tweet (TweetStrength), and whether it has a keyword in any of the policy agenda topics (HasTopicKeyword). The experimental results show that these features are useful to improve the classifier's performance. Next, feature selection, a process of selecting the optimal set of features from the given list such that the classifier yields best results, is applied. Lastly, a machine learning approach, SVM with linear kernel, Naïve Bayes, and Decision Tree, is applied. Among all of these machine learning algorithms, SVM with linear kernel performs best. The best recall for the "Has Agenda" class was 0.891 and the worst recall for the same class was 0.769.

Janardhan investigated a multi-class classifier to classify a given bill title of a congressional bill to one of the 19 policy agenda topics [9]. Similar to [41], this approach starts with data pre-processing by

deleting stop words and applying k-stemming for each congressional bill title. The second step is feature vector extraction. Several methods had been investigated: term occurrence, term frequency, normalized term frequency, and normalized topic keyword frequency. The third step is feature selection and the last step is machine learning. Decision Tree, Naïve Bayes, KNN, SVM with linear kernel, and boosting were investigated. The SVM based classifier with the normalized term frequency feature representation performs best on bill titles with 75% accuracy.

Unlike the study in [41] that classifies a tweet into two classes: “agenda” or “no agenda,” we classify a tweet into one of the 21 major topics. Furthermore, the dataset in our study has tweets not only from the official State Senate and House accounts, but also from individual house representatives. Compared to the study in [9] that uses congressional bills, our dataset includes tweets. Tweets are shorter and less formal than bill titles in most cases. Besides SVM with linear kernel, we use a convolutional neural network to build a multi-class classifier.

CHAPTER 3

PROPOSED TWEET CLASSIFICATION METHODS

We investigated three approaches to build a multi-class classifier that classifies tweets into one of the 21 classes. They are SVM with linear kernel, SVM with Topic Grouping and Tweet Merging, and CNN. We chose SVM with linear kernel since it was found to perform best for text classification in the study [17]. We introduce Topic Grouping that groups some topics together to balance the training set and merge tweets in the same group in order to balance the number of merged tweets among the groups. We chose the CNN approach because the experimental result in the study [28] shows that it is promising for short text classification.

3.1 Approach 1: SVM with linear kernel

Figure 3.1 shows the whole architecture of this approach.

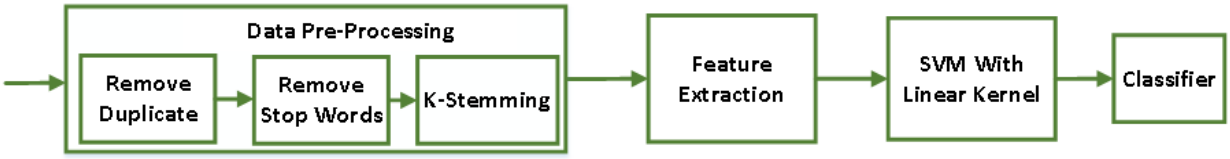

Figure 3.1. Architecture of Approach 1: SVM with Linear Kernel

3.1.1 Data Pre-processing

Data pre-processing is a common step in natural language processing (NLP), which removes information that is unlikely to contribute to the improved classification accuracy. We consider the following three aspects in our study.

(1) Remove duplicates. Some tweets are retweeted. The retweeted tweets are different from the original ones is that it starts with “RT @OriginalAccountName” in the tweet body. Occasionally, a human coder makes a mistake of assigning the duplicate tweets into different topics although they were retweeted from the same original tweet. That would make the duplicate tweets ambiguous, which impacts the classification accuracy. Therefore, we remove the retweets, keeping only the original tweets.

(2) Remove stop words & URLs. Some extremely common words, such as “the”, “is”, “at”, “which”, and “on” are of little value for classification. We define a set of stop words and remove all of them if they exist in each tweet. We also remove URLs in the tweet body since they are often short URLs that do not carry meaningful information.

(3) K-Stemming reduces a word to its root. For example, words "fishing" and "fished" could be reduced to the root "fish" although they are in a different form. There are several popular stemming methods in NLP, such as Porter Stemmer, K-Stemmer (K-Stem), and Hunspell Stemmer. We choose K-Stem because it can avoid over-stemming. Table 3.1 shows an example of over-stemming and under-stemming:

Table 3.1. Over-Stemming vs Under-Stemming

Original word	Over-Stemming (Porter)	Under-Stemming (K-Stem)
Politic	Polit	politic
Polite	Polit	polite

Although “Politic” and “Polite” have different meaning, an over-stemming algorithm can reduce them into the same root “polit.” However, K-Stem can avoid such a mistake.

3.1.2 Feature Extraction

Good features help improving the accuracy significantly. In our study, we use StringToWordVector (STWV) in Weka [42] to extract unique terms from tweets. Table 3.2 is a list of important parameters for STWV.

Table 3.2. Parameters of STWV for feature formulation

Parameters	Meaning
IDF Transform	Set if the word frequencies in a document should be transformed into: fij*log(num of Docs/num of Docs with word i) where fij is the frequency of word i in document (instance) j.

Table 3.2. (Continued)

TF Transform	Set if the word frequencies in a document should be transformed into: log(1+fij) where fij is the frequency of word i in document (instance) j.
TF-IDF Transform	Set if the word frequencies in a document should be transformed into: log(1 + fij) * log (num of Docs/num of Docs with word i), where fij is the frequency of word i in document (instance) j.
normalizeDocLength	Set if the word frequencies for a document (instance) should be normalized or not.
outputWordCounts	Output word counts rather than boolean value of 0 or 1(indicating presence or absence of a word)

In our study, we set the outputWordCounts to true and normalize all data both in the training set and testing set. We also use Term Frequency–Inverse Document Frequency (TF-IDF) as features. TF-IDF is a numerical statistic taken into account the rarity of a word in a collection or corpus [43]. Table 3.2 shows the formula for computing TF-IDF values.

3.2 Approach 2: SVM with Topics Grouping and Tweets Merging

Besides using SVM with linear kernel, we group some topics and merge tweets of the same topic to build our classification model in order to balance the number of tweets across topics. Figure 3.2 shows the distribution of policy agenda topics from tweets of Iowa senators and house representatives.

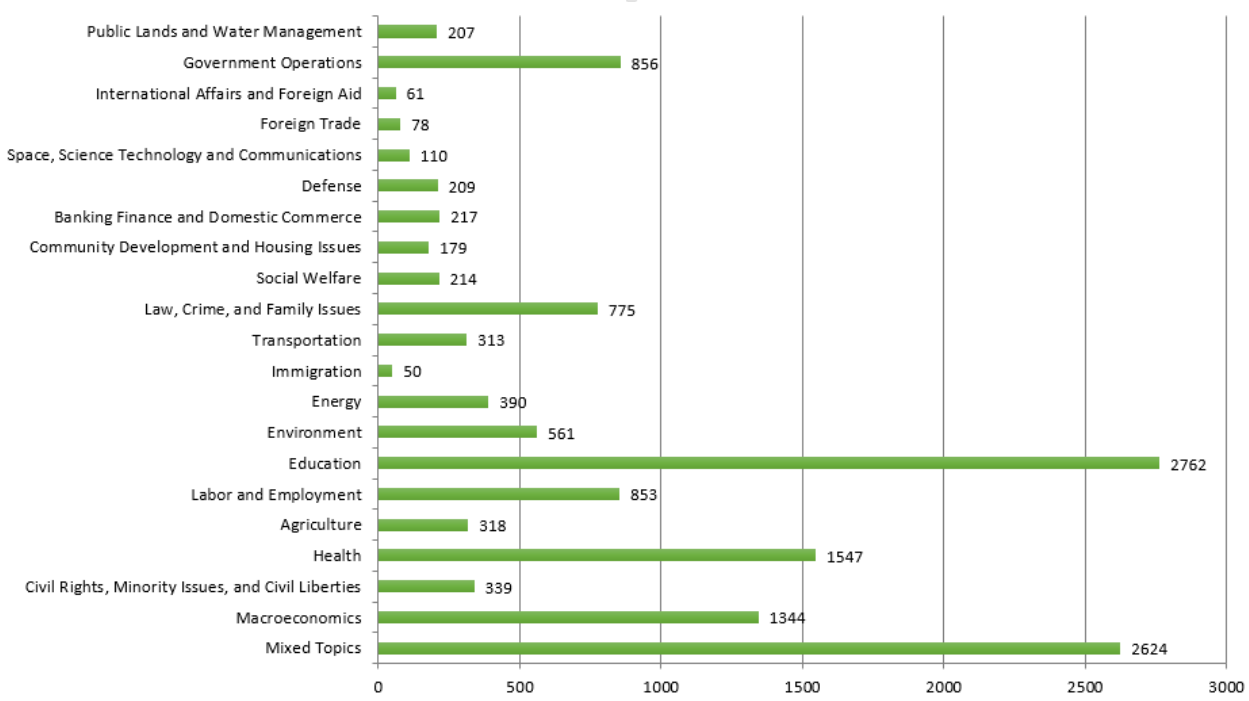

Figure 3.2. Policy Agenda Topics Distribution in Iowa dataset from June 2008-November 2015

Figure 3.2 reveals that the numbers of tweets are unevenly distributed across topics, which greatly affects the performance of the classifier. For example, the numbers of tweets in "Education" and "Mixed Topics" are 2,762 and 2,624, respectively. However, for some topics, such as "International Affairs and Foreign Aid", "Foreign Trade", and "Immigration", the number of tweets in each of these topics is lower than 100.

Our idea is to make the number of tweets evenly distributed across topics as much as possible using topics grouping and tweets merging. Figure 3.3 shows the overview of this approach, which is the same as the first approach except the "Topics Grouping" and "Merging Tweets" steps.

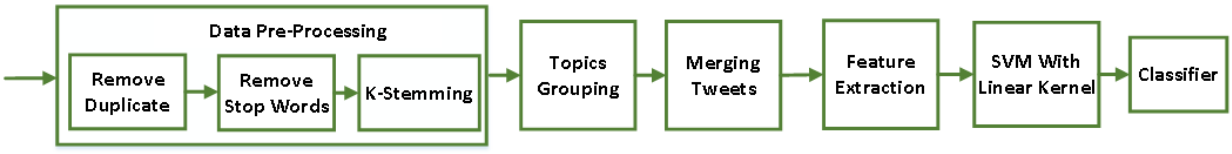

Figure 3.3. Architecture of SVM with Topics Grouping and Tweets Merging

3.2.1 Topics Grouping and Tweets Merging in Training Set

To even out the numbers of tweets across the topics, we group the topics with few tweets into a "Combined Topic." Figure 3.4 shows the example of the Iowa dataset where we grouped the fourteen topics with fewer tweets into the "Combined Topic".

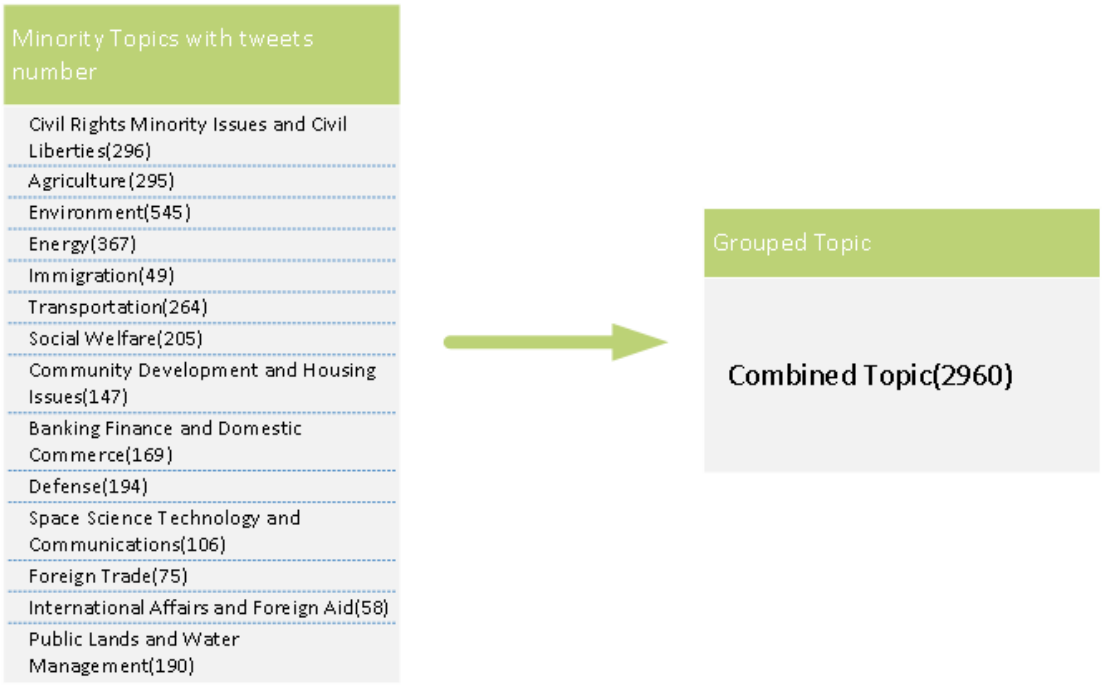

Figure 3.4. Group Minority Topics into a Combined Topic in Iowa

After topics grouping, the numbers of tweets across topics were still unevenly distributed as shown in the second column in Table 3.3. We merged the tweets of the same topic in a sliding window in chronological order in the training set. Table 3.3 shows the window sizes for the combined topic and the numbers of tweets after merging in columns 3 and 4. The numbers of tweets across the topics become more even and each tweet becomes longer.

Table 3.3. Sliding window sizes and the numbers of tweets before/after merging of the Iowa data set

Topic Name	Number of Tweets (After Grouping)	Windows Size	Number of Tweets (After merging)
Mixed Unknown Topics	2345	5	469
Macroeconomics	1200	2	600
Health	1362	3	454

Table 3.3. (Continued)

Education	2442	5	489
Labor and Employment	761	2	380
Law Crime and Family Issues	691	1	691
Government Operations	756	2	378
Combined Topic	2898	6	483

3.2.2 Tweets Merging in Testing Set

Since the merged tweets in the training set becomes longer, using the original tweets in the testing set to test the performance of the classification model built by SVM with topics grouping and tweets merging results in low accuracy even when normalizing all the data in the training and testing sets. We then merged tweets in the testing set. Unlike the merging in the training set, we do not know which topic a tweet belongs to in the testing set. We explore two ideas for merging tweets in the testing set based on the assumptions that two tweets belong to the same topic if their text are more similar. We calculate the similarity of the text of the tweets and use it to merge tweets in the testing set.

3.2.2.1 Text Similarity

There are many well-known methods to calculate text similarity. We investigated three different popular similarity functions: Euclidean Distance, Cosine Similarity, and Jaccard Similarity. Given two tweets: T_1 with n unique words and T_2 with m unique words. Let $S_1 = \{w_{11}, w_{12}, w_{13}, ..., w_{1n}\}$ and $S_2 = \{w_{21}, w_{22}, w_{23}, ..., w_{2m}\}$ be the sets of unique words in T_1 and T_2, respectively. Let vectors $V_1 = [v_{11}, v_{12}, v_{13}, ..., v_{1k}]$ and $V_2 = [v_{21}, v_{22}, v_{23}, ...v_{2k}]$ be the TF-IDF vector representations of T_1 and T_2, respectively, where $K = | S_1 \cup S_2 |$.

(1) Euclidean Distance

$$eDist(T_1, T_2) = \sqrt{V_1 \cdot V_2} = \sqrt{(v_{21} - v_{11})^2 + (v_{22} - v_{12})^2 + ... + (v_{2k} - v_{1k})^2}$$

The smaller of the value of $eDist(T_1, T_2)$, the closer the tweets T_1 and T_2.

(2) Jaccard Similarity

$$jcardSim(T_1, T_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

The $jcardSim(T_1, T_2)$ of zero indicates that the two tweets are totally different. When $jcardSim(T_1, T_2)$ equals one, the two tweets are considered identical.

(3) Cosine Similarity

$$\cos ineSim(T_1, T_2) = \frac{V_1 \cdot V_2}{\| V_1 \| * \| V_2 \|} = \frac{\sum_{i=1}^{k} v_{1i} * v_{2i}}{\sqrt{\sum_{i=1}^{k} v_{1i}^2} \sqrt{\sum_{i=1}^{k} v_{2i}^2}}$$

If $\cos ineSim(T_1, T_2)$ equals one if the two tweets are identical. If it equals zero, these two tweets are totally different.

3.2.2.2 Individual Similarity vs Center Similarity

We investigated two different methods to predict a policy agenda topic of a tweet in the testing set for the grouping purpose.

1. Grouping with the individual similarity: for each tweet in the testing set, we calculate the similarity with each tweet in the training set using their feature vectors. The tweet from the testing set is assigned the topic of the tweet in the training set with the largest similarity to it. This method incurs high processing overhead.
2. Grouping with the center similarity: We first calculate the center for each topic from the feature vectors of that topic in the training set. Then, for each tweet in the testing set, we calculate its similarity with the calculated center of each topic. The tweet is assigned the topic that it is most similar with to the center of the topic. This method reduces the computation overhead and the effect of the outliers in the training set.

3.3 Approach 3: Convolutional Neural Network

Due to CNN success in image classification and recently in text classification, we investigate how well this approach works on short text classification.

3.3.1 Vector representation of a word with word2vec

A word needs to be represented by a vector to be used with CNN. The traditional approach is to assign a unique number for each word in the dictionary, for example, 1 for “child” and 2 for “son”. Then, we can represent the word “child” as [1,0,0,0……] and “son” as [0,1,0,0……]. The length of the vector is the size of dictionary. Obviously, such a word representation is context free. The drawbacks of this representation are as follows. (1) We have a large and sparse vector representation for each word. (2) This vector representation does not capture semantic relationship between words whether they are exchangeable or appear in similar sentences.

In the CNN method [27], the word embedding method is used to derive a vector representation of a word based on the distributional hypothesis. That is words appearing in a similar context share similar semantic meaning. As a result, they should have a similar word representation. For example, in these two sentences: “China is a great country” and “USA is a great country.” The words “China” and “USA” share the same surrounding words (or context), the vector representations of “China” and “USA” should be similar, for instance, [-0.2, -0.2, 0.0, 0.1] for “China” and [-0.1, -0.2, 0.2, 0.1] for “USA”. The length of the vector using this approach is far shorter than the vector length of the traditional method.

In our study, we use the Google’s open source word2vec trained using a large corpus to convert each word into its representation taking into account the context. For each word, word2vec returns top *k* words and corresponding cosine similarity to the given word in descending order. The result forms a k-dimensional vector as a word embedding representation of the given word. Figure 3.5 shows an example of the word “Soccer” represented in the word embedding form as [0.762262, 0.722919, 0.709261, 0.663754, 0.658547, 0.646368, 0.641979, 0.619528, 0.601519, 0.593004]. This representation means that “Soccer” is seen in the same context as “soccer” the most, followed by “Football”, “Lacrosse”, and so on.

Word	Cosine distance
soccer	0.762262
Football	0.722919
Lacrosse	0.709261
Youth_Soccer	0.663754
Volleyball	0.658547
Basketball	0.646368
Softball	0.641979
Baseball	0.619528
SOCCER	0.601519
Soccer_Federation	0.593004

Figure 3.5. Top ten closet similar words returned by word2vec

3.3.2 Convolutional Neural Network

Figure 3.6 shows the overall architecture of the Convolutional Neural Network (CNN) approach applied to our problem. Firstly, each tweet without any retweets is converted into a 2-dimensional matrix where each row in the matrix is the word representation of each word in the tweet. The rows are ordered in the order that the words appear in the tweet. Secondly, during training, in the convolutional layer, different filters with various window sizes are automatically selected by CNN and applied on this matrix to generate feature maps. Thirdly, the max-over-time pooling layer selects one feature for each tweet from the convolutional layer. The output feature vector is passed to a fully connected Softmax layer whose output is the probability distribution over our 21 classes (i.e., 20 policy agenda major topics and one additional mixed topic). The training process determines the appropriate parameter values that result in the model that gives the highest classification accuracy using the verification data.

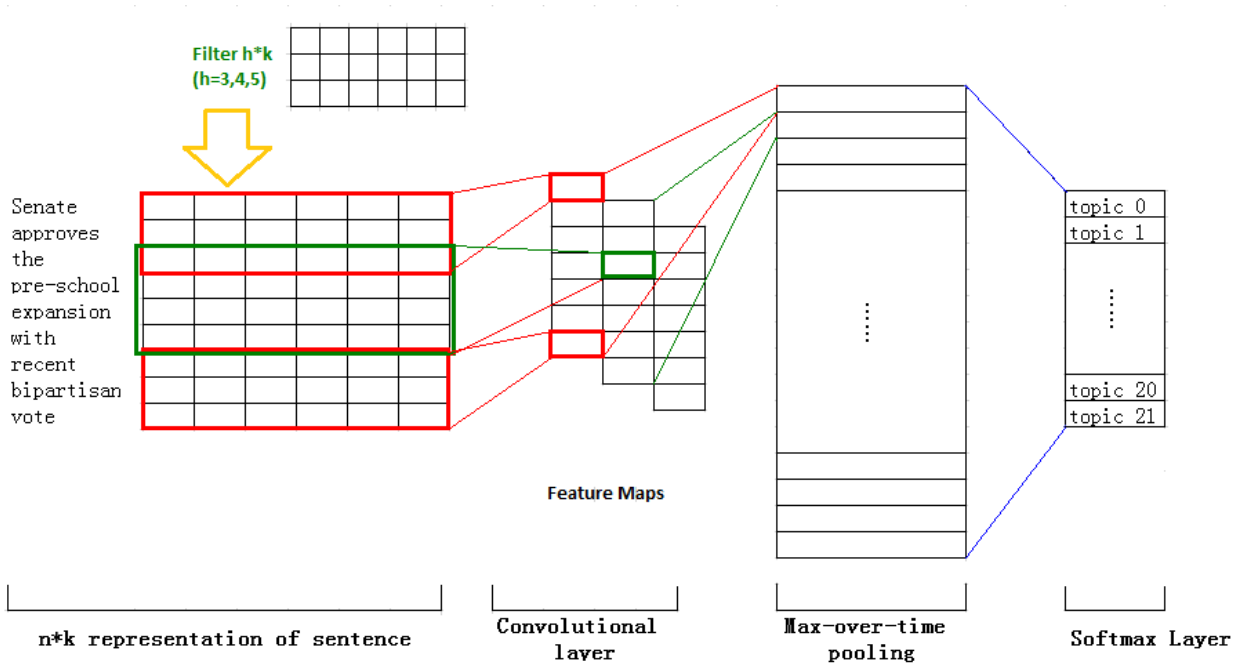

Figure 3.6. Architecture of Approach 3: CNN

We explain each step in detail in the CNN approach as below.

(1) Convert a tweet to a corresponding 2-dimensional matrix: The word2vec tool converts each tweet into a 2-dimensional matrix where each row is of a fixed size *k* and is a vector representation of a word in the tweet. For a tweet with *n* words, the result of this step is an *n*k* matrix for each tweet.

(2) Apply the filter with the sliding window of size *h* words (i.e., *h* rows)*:* Let $T = [w_1 w_2 w_3 ... w_n]$ represent a tweet, where w_i represents the i-th word in the tweet. With the window size of *h* words, we generate *n-h+1* substrings from the tweet. Each substring *i* is denoted as $T_i = w_i w_{i+1} w_{i+2} ... w_{i+h-1}$ where $i = 1,2,....,n-h+1$. By using word2vec, we get a *h*k* matrix for each substring. The CNN method randomly applies a filter $F = R^{h*k}$ with the size of *h*k* on this *k*h* matrix, which results in an *h*h* matrix M_i. For all these *n-h+1* substrings of the tweet, we have *n-h+1* M_i matrices with the dimension *h*h* where $i = 1,2,....,n-h+1$.

(3) Generate a feature: To generate a feature, the CNN method uses a non-linear function, such as a hyperbolic tangent function, to convert the *h*h* matrix to a real value. We use the following function.

$c_i = f(M_i + b)$, where *b* is a bias term which is also an *h*h* matrix. After applying the function for each matrix M_i where $i = 1,2,....,n-h+1$, we get a feature map $c = [c_1, c_2, ..., c_{n-h+1}]$ that has one element c_i for each of the *n-h+1* matrices.

(4) Max-over-time pooling operation: For each feature map, the max-over-time pooling operation selects the maximum value in this feature map. That is, $\hat{c} = \max\{c_i\}$ where $i = 1,2,....,n-h+1$. The step chooses the most important feature for each feature map for a specific filter.

For one tweet, the CNN approach generates the best feature given one filter. Since many different filters with different sliding window sizes are applied, the CNN approach generates a set of best features, which forms a penultimate layer $z = \{\hat{c}_1, \hat{c}_2, \hat{c}_3 ... \hat{c}_m\}$, where *m* is the number of filters.

(5) Softmax layer: The output of the penultimate layer is passed to a fully connected Softmax layer whose output is the probability distribution over 21 classes.

(6) Regularization: We use the L2 norm and a dropout technique for regularization, which prevents overfitting. For the filter matrix $F = R^{h*k}$, if the L2 norm of F is larger than *s* (s is a hyperparameter in CNN and was set by an empirical value in our study), i.e. $\| F \|_2 > s$, the CNN method rescales it to $\| F \|_2 = s$ after a gradient descent step. The dropout technique is applied on the penultimate layer, by setting a dropout parameter *p* where $0 \le p \le 1$, only (1-*p*) of the total features in the penultimate layer are passed to the Softmax layer.

(6) Static vs non-static channel: With the static channel method, CNN uses the pre-trained vectors from word2vec. All words with its cosine similarity vector remain the same and the other hyper-parameters of the model are learned during the whole process. Compared with the static channel method, the non-static channel adjusts the pre-trained vector in a different epoch of the training process.

3.4 REST API

We proposed two API endpoints as shown in Table 3.4. The similarStates endpoint is to list the top *k* states with the distribution of tweets categorized into the policy agenda topics similar to that of a given state. On the other hand, the similarTopic endpoint is to query for the top *k* states with the most percentage of tweets in a given topic for a given year and a given month. The two API endpoints should enable political scientists to be able to test different policy diffusion among states.

Table 3.4. REST API endpoints for predicting the top k popular topics

GET API endpoints	**Description**
/v1/policyagendas/tweets/similarStates?state=*{state}*&year=*{year}*&month=*{month}*&k=*5*&distinct=*{true|false}*	Returns a JSON string of top *k* states ranked by the similarity in the percentage of tweets in each topic to that of a given state in a given year and a given month. The search range is limited to one year dated back from the year and month parameters. **Required parameters:** state, year, month **Optional parameters:** • distinct with the default value of *false* • *k* with the default value of 5 **Format of the parameters:** • Year parameter: mm/dd/yyyy • Month parameter: [0,12] • State parameter: two characters such as IA, IL Example: /v1/policyagendas/tweets/similarStates?state=IA&year=2012&month=5&k=2 {state: IA, year: 2012, month: 05, k: 2, similarity: [

Table 3.4. (Continued)

	{ cosineSimilarity: 1, Month:5, Year:2012, statePair: “IA,IA”, vector: [// similarity in terms of percentage of tweets of each of the 21 topics, from topic 1, to topic 21 53.21, // 53 percent of tweets are in topic 1 21.00, // 21 percent of tweets are in topic 2 11.01, 6.02, …] cosineSimilarity: 0.9, Month:5, Year:2012, statePair: “IA,WA”, vector: [21.00 22.00 …] }, {cosineSimilarity: 0.8, Month:5 Year:2012 statePair: “IA,IL”, vector: [11.00, 12.00, ….] } } In this example, we show the percentage of tweets classified into each topic for IA and the next top 2 states, WA and IL, with similar classified policy agenda topics. We do not take into account the mixed agenda topic. This query should search for all the policy agendas similar to Iowa whether they occur in the same year or month or not.

Table 3.4. (Continued)

	Optional parameters: The distinct parameter is either *true* or *false* to limit the record number from a same state, if distinct is true, only one record from one state could be shown in the result.

/v1/policyagendas/tweets/similarityTopic?topic=*{topicnumber}*&k=*2*&year=*{year}*&month=*{month}*	Returns a JSON of vectors of top *k* states with the most percentage of tweets in a given topic for a given year and a given month. List the percentage of tweets in that topic for each state.
	
	Required parameters: topic, year, month
	Optional parameters:
	• k with the default value of 5
	
	Return result: JSON string
	{ topic: 1,
	year: 2012,
	month: 05,
	k: 1,
	topstates: [
	{IA: {
	year:2012:
	month: 05,
	percentTweets: 0.50}
	},
	{IL: {
	year:2012,
	month:04,
	percentTweets:05
	}
	}
	]
	}
	
	The query parameters and values are always listed in the returned result.

CHAPTER 4

EXPERIMENTAL DESIGN AND RESULTS

4.1 Datasets

We collected 308,601 tweets from 472 official accounts of Senate and House and of individual senators and house representatives from eleven states during the period of 06/29/2008 to 11/29/2015. Appendix B shows the Twitter handles of these accounts used for the data collection process. Figure 4.1 reveals the difference in the distribution of the number of tweets across states. These states, New York, Missouri, Minnesota, and California, have noticeably fewer tweets than the other states because the dataset of these states does not include tweets of individual senators or house representatives.

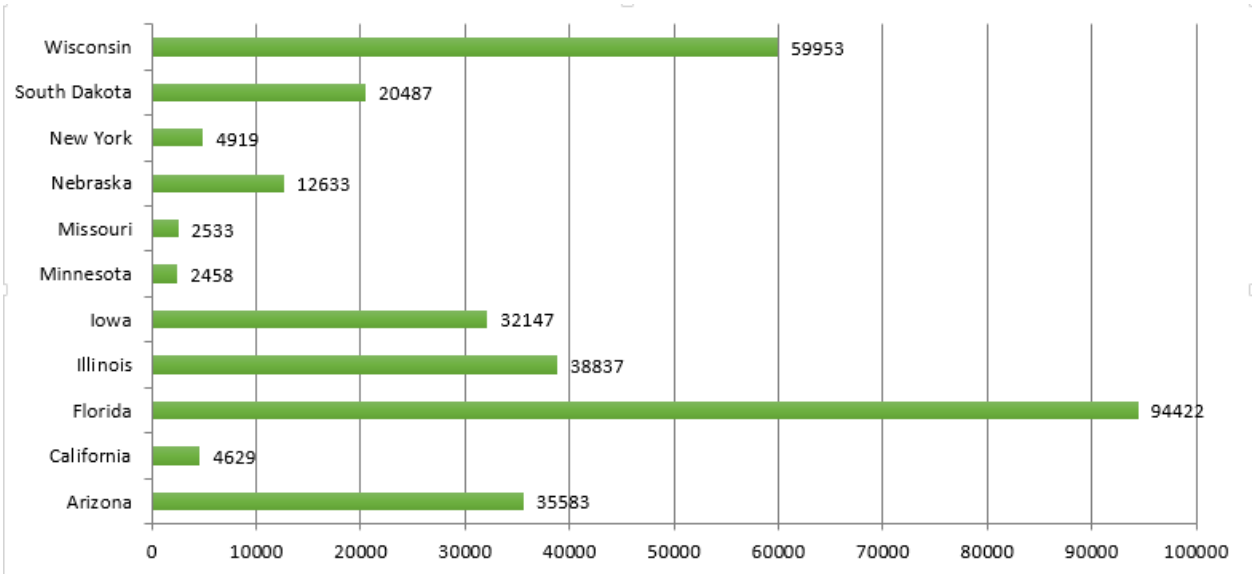

Figure 4.1. Distribution of tweets from official senate, senator, and house representative accounts from eleven states

We selected the tweets from Iowa and Nebraska for hand labeling of ground truth by two political science students under the guidelines and rules specified in the Policy Agendas codebook and under the guidance of a political scientist. The labeling process consists of two steps: (1) Label a tweet as “Has Agenda” or “No Agenda”. (2) Assign each tweet labeled as “Has Agenda” one of the 21 policy agenda

topics. Only the tweets labeled as “Has Agenda” were used as the ground truth in our study. Each tweet was labeled by one student. All the tweets labeled as “Has Agenda” from Iowa are referred to as the Iowa dataset and all the tweets labeled as “Has Agenda” from Nebraska are referred to as the Nebraska dataset. Each of the Iowa dataset and Nebraska dataset was divided into non-overlapping training and testing sets.

Figure 4.2 shows the distribution of tweets across topics in the Iowa and Nebraska datasets. Excluding the mixed unknown topic, “Education” and “Health” topics were two most popular topics for Iowa. Unlike Iowa, “Education” was not talked about in Nebraska as much as in Iowa. “Health” and “Macroeconomics” were two most popular topics for Nebraska.

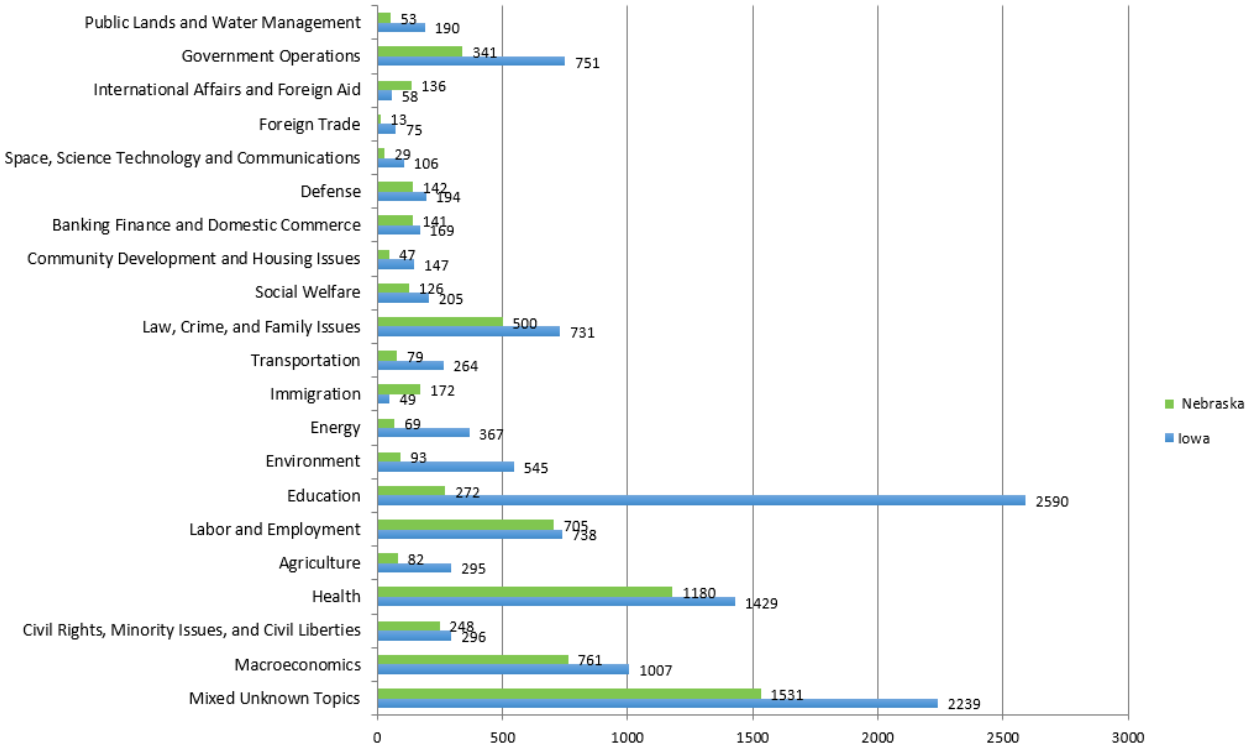

Figure 4.2. Distribution of the number of tweets across policy agenda topics in the Iowa and Nebraska datasets

4.2 Implementation

We use MySQL Community Server 5.7.13 for Windows for storing the tweets. The website for ground truth was implemented in PHP5. Our REST API was developed in Java using Jersey 2.22.2 REST API toolkit. The source code is shown in the Appendix.

4.3 Experimental Design

In our experiment, we used 10% of total tweets in the ground truth as an independent testing set. The rest was used as a training set for 10 fold cross validation to build the multi-class classifier. Figure 4.3 shows the number of tweets used in the training set for 10 fold cross validation and the independent testing set. The Nebraska dataset has about half the number of tweets of the Iowa dataset.

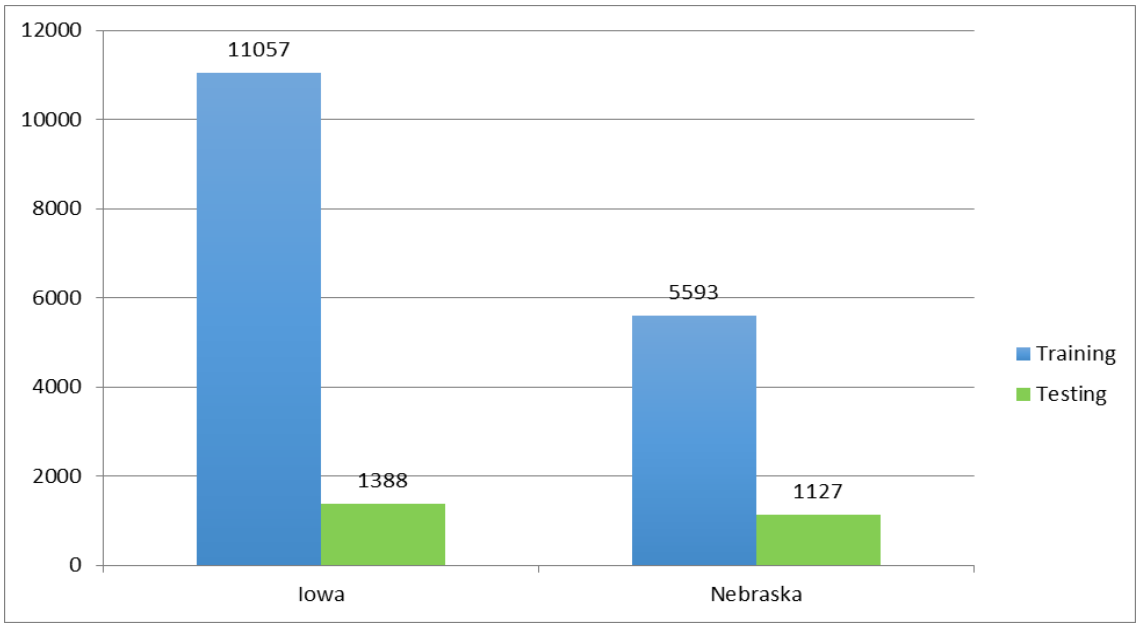

Figure 4.3. Number of tweets in the training and testing sets of Iowa and Nebraska datasets

Besides the 10 fold cross validation method, each approach has its own parameter setting. For Approach 1, we chose “Poly Kernel” for SVM parameter. Table 4.1 is a list of parameters of STWV used in the experiments.

Table 4.1. Parameter values for STWV

Parameters	Settings
IDFTransform	True
TFTransform	True

Table 4.1. (Continued)

normalizeDocLength	normalize all data
outputWordCounts	True

Most parameter values for Approach 2 were same as those of Approach 1 except the parameter values for topics grouping and tweets merging. For the training set of the Iowa dataset, we grouped 14 topics with fewest tweets into one "Combined Topic," but for the training set of the Nebraska dataset, we grouped 15 topics into one "Combined Topic" instead. The number of groups was different because of the different distribution of the number of tweets in the two datasets. Figures 3.3 and 4.4 show the grouping of topics for training sets of the Iowa and Nebraska datasets, respectively.

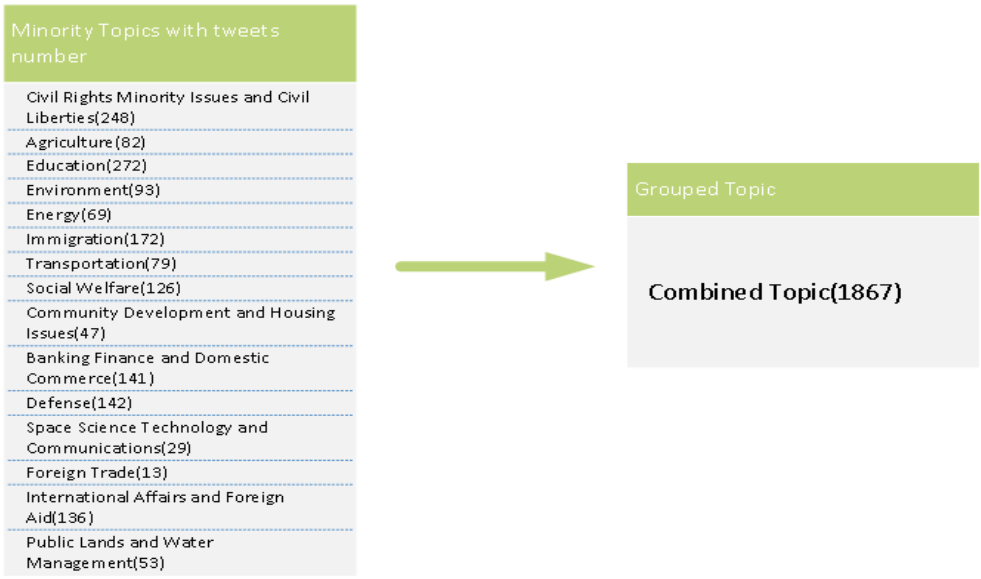

Figure 4.4. Group Minority Topics into a Combined Topic in Nebraska

For tweets merging in the training set, different window sizes were used for different topics since the distribution of the number of tweets from the Nebraska dataset is different from that of the Iowa dataset. Table 3.3 and Table 4.2 show the parameter values and result from tweets merging for Iowa and Nebraska, respectively.

Table 4.2. Sliding window size and number of tweets before/after merging in the Nebraska dataset

Topic Name	Number of Tweets (After grouping)	Window Size	Number of Tweets (After merging)
Mixed Topics	1812	4	453
Macroeconomics	890	2	445
Health	1269	3	423
Labor and Employment	754	2	377
Law Crime and Family Issues	531	1	531
Government Operations	400	1	400
Combined Topic	1924	5	384

For tweets merging in the testing set, we experimented with the "individual based" and the "center based" text similarity methods. Since tweets in the "Mixed Topic" are ambiguous, we also measured the classification accuracy when merging of tweets in the same topic was used when including or excluding the "Mixed Topic."

For Approach 3, most of the hyper-parameters are tuned automatically by CNN. There are only three parameters that require manual setting. The first one is the word's vector dimensional length in word2vec. We chose 300 as the vector length which was an empirical value. The second one is the set of windows sizes, which we chose the empirical value set {3,4,5}. The third one is the selection of the channel, either Static or Non-Static.

4.4 Experimental Result on Classification Accuracy

Although SVM with linear kernel was found to achieve the best performance for text classification [17], it does not perform as well for the classification of tweets into the 21 policy agenda topics. Figure 4.5 shows that the highest accuracy using SVM with linear kernel is only 60.44% on the testing set of the Iowa dataset. Furthermore, there is a large reduction in accuracy in the testing set of about 10% lower compared to that of the training set. This approach is not as good as we have seen in longer text such as bill titles and documents. The TF-IDF features extracted from short text are not

discriminative enough for this classification problem. Finally, the number of tweets of each topic is different. In other words, the dataset is unbalanced.

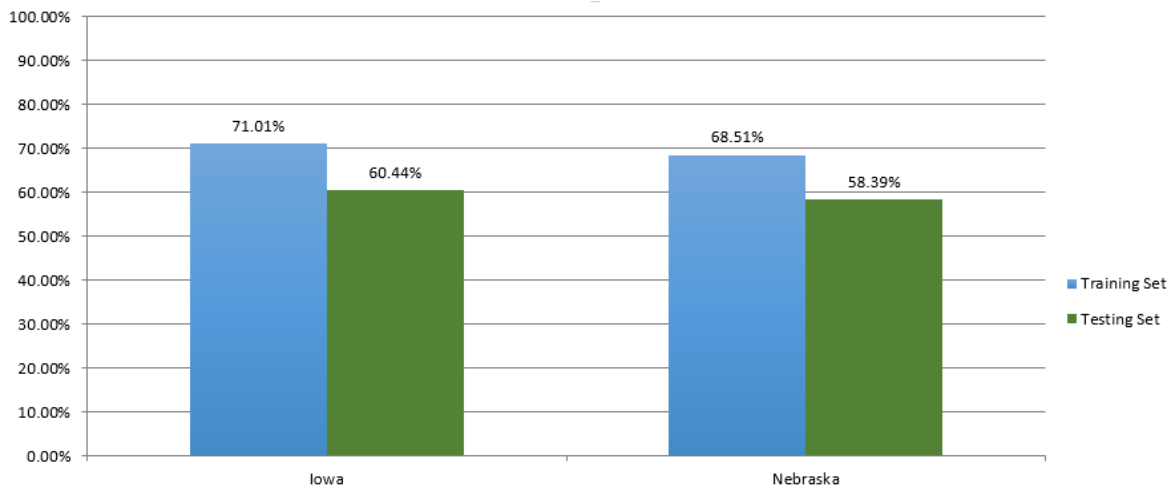

Figure 4.5. Accuracy of Approach 1: SVM with Linear Kernel on Iowa Testing Set

Figure 4.6 and Figure 4.7 reveals that "Topics Grouping and Tweets Merging" improves the accuracy around 20% on the training set compared to SVM without this step. This performance improvement could be the benefit of a more balanced training set and a longer text combining multiple tweets of the same topic. However, the best accuracy on the testing set is only 56.39%, which is even worse than that of Approach 1. The reason is the ineffectiveness of merging tweets on the testing set since the best accuracy for this task is only 62.02% as shown in Table 4.3. The best accuracy comes from the center-based method both in Iowa and Nebraska regardless whether the "Mixed Topics" is included or not.

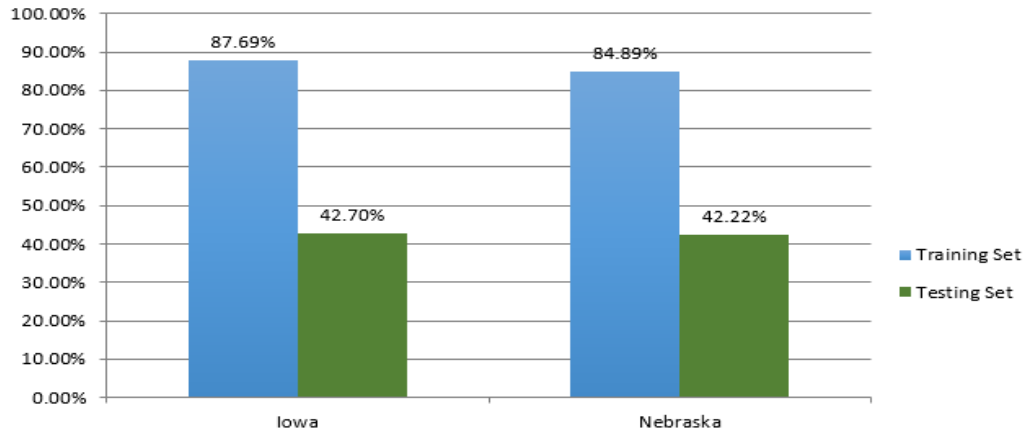

Figure 4.6. Accuracy of Approach 2 : Include Mixed Topics

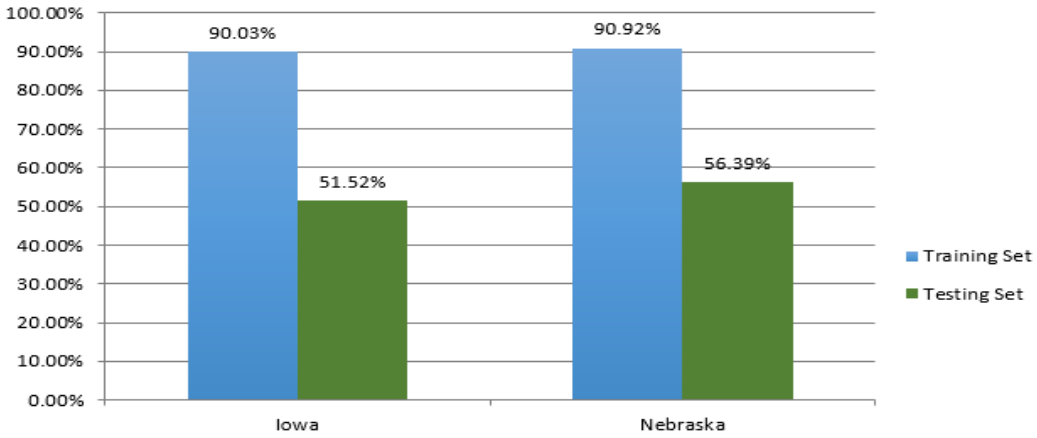

Figure 4.7. Accuracy of Approach 2 : Exclude Mixed Topics

Compared with the previous two approaches, Figure 4.8 and Figure 4.9 shows that the CNN approach achieves the best accuracy of 76.55% on the testing set of the Iowa dataset. This approach is stable since there is less than 1% accuracy difference on the training set and the testing set.

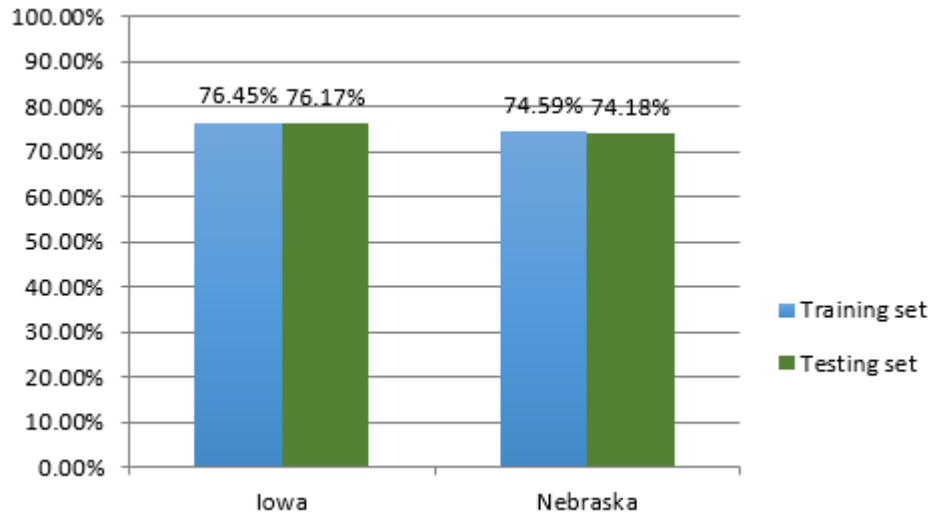

Figure 4.8. Accuracy of Approach 3 : CNN Static

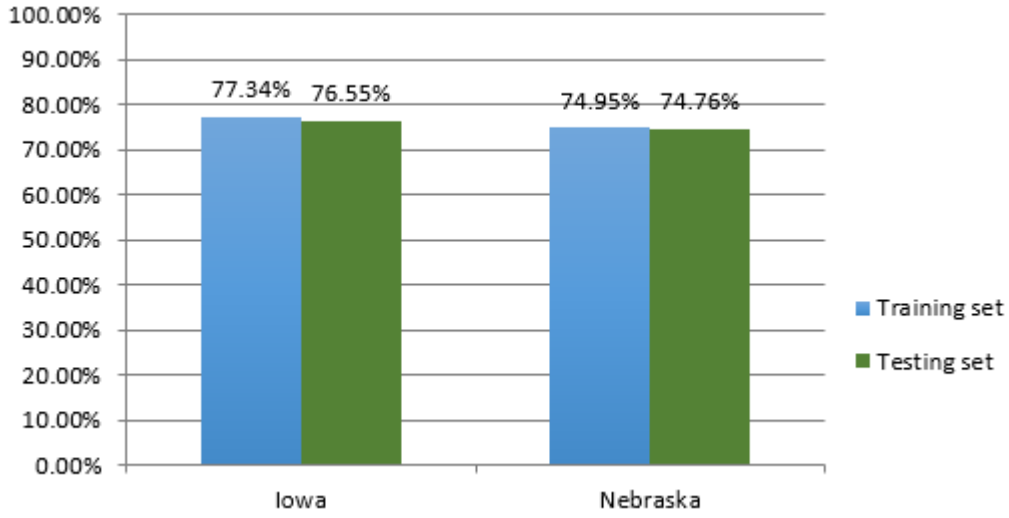

Figure 4.9. Accuracy of Approach 3 : CNN Non-Static

Figure 4.10 and Figure 4.11 shows the detailed accuracy on different topics for Iowa and Nebraska. The accuracy on some topics, like “Foreign Trade” in Nebraska, is 0% because the number of tweets in the training set and the testing set in this topic is small. For example, “Foreign Trade” in Nebraska has 10 for training and only 3 tweets for testing.

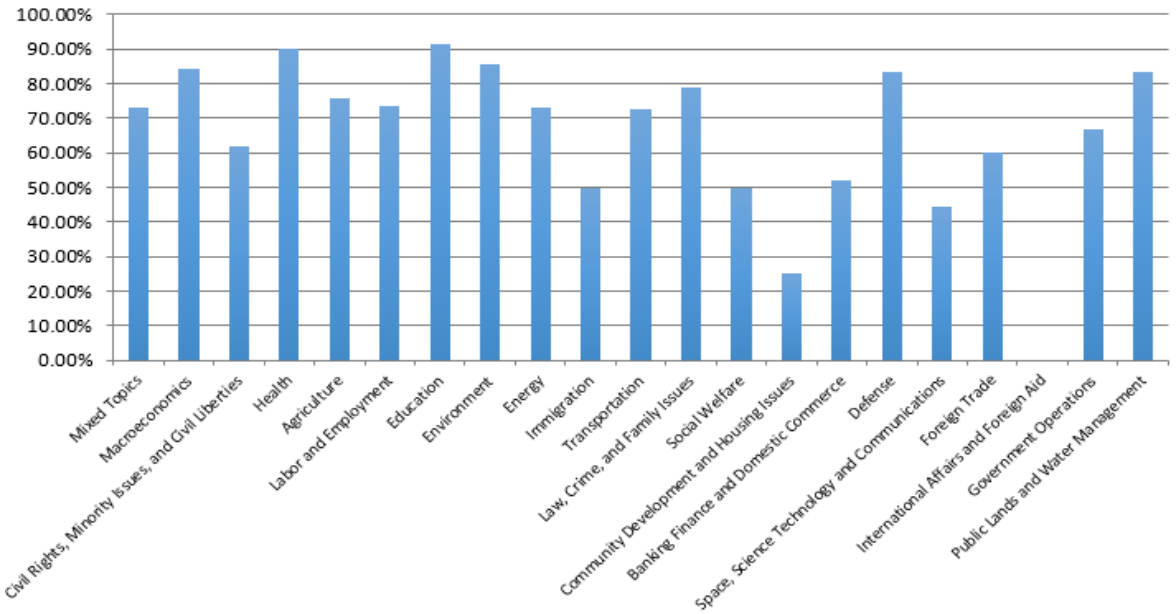

Figure 4.10. Accuracy across Different Topics in Iowa

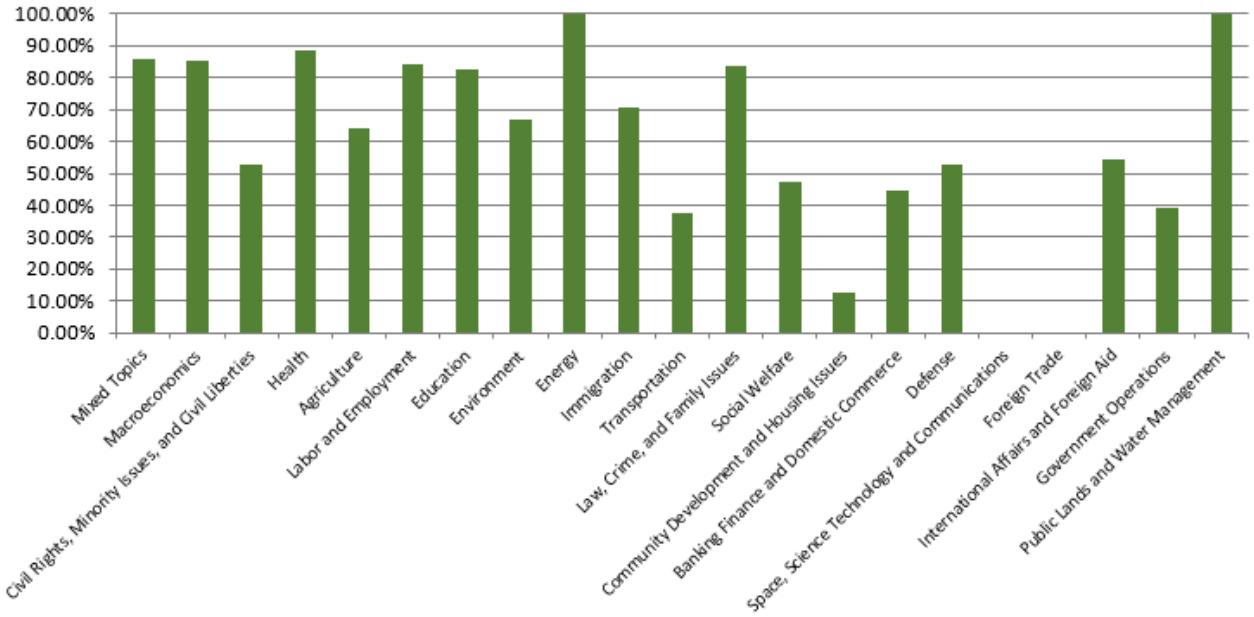

Figure 4.11. Accuracy across Different Topics in Nebraska

Since the CNN approach achieves the best accuracy, we further conducted three experiments to determine whether one classifier built from sample tweets from a set of states is able to provide comparable classification accuracy for tweets from these states compared to a specific classifier for each state.

We experimented with the following.

(1) Building a classifier from the combined training datasets (16,650 tweets in total) of the Iowa and Nebraska to classify the combined testing set (2,515 tweets in total) of Iowa and Nebraska datasets.

(2) Using the classifier built from the training set of the Iowa dataset (12,445 tweets in total) to classify the Nebraska dataset (6,720 tweets in total).

(3) Using the classifier built on the training set of the Nebraska dataset (6,720 tweets in total) to classify the Iowa dataset (12,445 tweets in total).

Figure 4.12 shows the best accuracy for these experiments with the non-static method. The figure reveals that the accuracy drops significantly if using the classifier built from another state. The classifier built from the training data of the combined Iowa and Nebraska datasets is able to perform well.

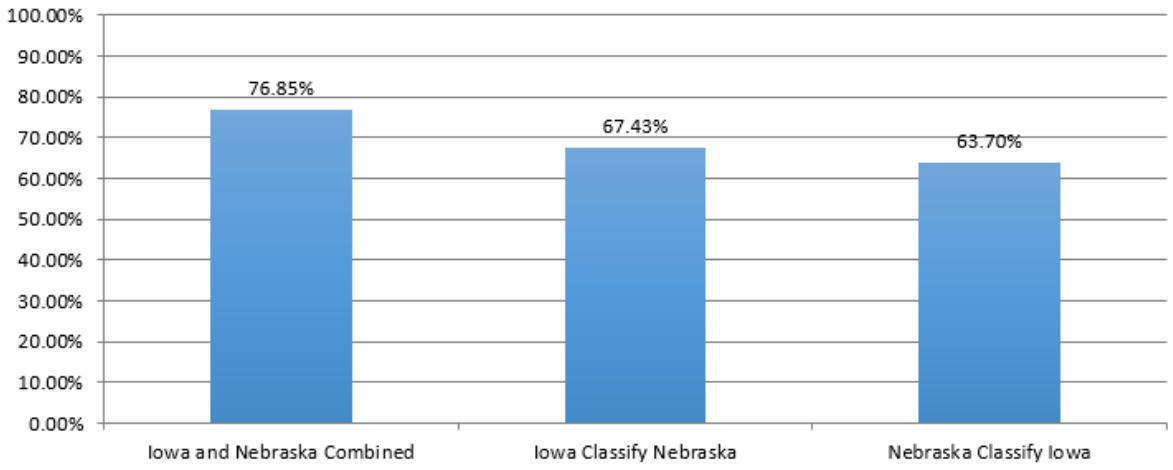

Figure 4.12. Accuracy on Cross-testing

4.5 Experimental Results on Prediction of Top k Topics by CNN

To evaluate whether the CNN method with the current accuracy is of any practical value for political science research, we compared the prediction by the CNN method for the top *k* topics discussed in each month by state's legislatures from December 2014 to November 2015 with the ground truth top *k* topics of the same month in the Iowa dataset and Nebraska dataset. Appendix C shows the prediction of the

number of tweets of the top 7 topics for each state. For instance, in September 2015, 309 tweets were predicted to be in Topic 0 (Mixed Topics), but the ground truth (original result) has the most number of tweets, 274 tweets, in Topic 6. Based on the comparison data in the above period, we found the following:

1). During Feb. 2015, Mar. 2015, Jun. 2015, and Jul. 2015 for Iowa, top 4 topics of the predicted result and the ground truth matched.

2). In Jul. 2015 for Nebraska, top 4 topics matched.

3). For both Iowa and Nebraska, top 5 topics matched in the other months.

4). Topic 0 appears in all the top 5 predicted topics as well as the ground truth.

Topic 0 is the “Mixed Topics” which is the tweet where the human coder cannot decide to assign the tweet on one major topic. The prediction results of the top 4 topics match with the ground truth for all the months under study for both states although the ordering of the match is different in some months. It is promising that the current result is useful for studying the trends of policy agenda topics under discussion in Twitter.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, we explored a new research problem of determining the major policy agenda topic of a given tweet. This problem is challenging since some tweets are ambiguous even when human coders determine them, containing words of more than one topic; there is a limitation of 140 characters per tweet; and there are as many as 21 classes to consider. We investigated three approaches. The experimental results show that our application of Convolutional Neural Network (CNN) provides a significant accuracy improvement about 15% over SVM with linear kernel with TF-IDF---term frequency weighted by the inverse document frequency of the terms, on the same testing set. Besides providing the best accuracy, the CNN method is also stable as the accuracies on the training set and the testing set are quite similar. Furthermore, it achieves good accuracy without manual tuning of hyper-parameters since they are determined automatically by CNN during the training process.

SVM with Topics Grouping and Tweets Merging offers good accuracy on the training set. It shows that lengthening tweets improves the classification accuracy. However, the same level of accuracy is not seen in the testing set because the accuracy due to grouping and merging is only 62.02% at best. If the grouping and merging of the testing set could be improved, the overall performance of SVM with Topics Grouping and Tweets Merging will be improved.

With the CNN method, the *5* hottest topics discussed in one month in a given state can be estimated quite reliably as evidenced in the experimental results and can be used to test different theories about the diffusion of policy agendas across states.

References

[1] Twitter Introduction. Available: https://en.wikipedia.org/wiki/Twitter

[2] Use of Twitter by public figures, Available: https://en.wikipedia.org/wiki/Use_of_Twitter_by_public_figures#Politicians

[3] Top Politicians on Twitter, Available: http://fanpagelist.com/category/politicians/view/list/sort/followers_today/page1

[4] Michael D. Conover, Bruno Goncalves, etal, Predicting the Political Alignment of Twitter Users, 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011, pp. 192-199.

[5] Political fundraiser by Hillary Clinton, Available: https://twitter.com/HillaryClinton/status/735185631148703748

[6] The data used here were originally collected by Frank R. Baumgartner and Bryan D. Jones, with the support of National Science Foundation grant numbers SBR 9320922 and 0111611, and were distributed through the Department of Government at the University of Texas at Austin. Neither NSF nor the original collectors of the data bear any responsibility for the analysis reported here.[Online].

[7] Code book. Available: http://www.policyagendas.org/sites/policyagendas.org/files/Topics_Codebook_2014.pdf

[8] E. S. A. a. J. Wilkerson. Congressional Bills Project. Available: http://congressionalbills.org/

[9] Ragavi Pala Janardhan, Automated Policy Topic Classification of Congressional Bills, Graduate Theses and Dissertations Digital Repository @ Iowa State University 2015.

[10] S. Purpura and D. Hillard, "Automated classification of congressional legislation," in Proceedings of the 2006 international conference on Digital government research, San Diego, California, USA, 2006, pp. 219-225.

[11] Yihua Liao,V. Rao Vemuri,Using Text Categorization Techniques for Intrusion Detection, Proceedings of the 11th USENIX Security Symposium,2002, pp. 51-59.

[12] Li Baoli, Yu Shiwen, and Lu Qin, An Improved k-Nearest Neighbor Algorithm for Text Categorization, Proceedings of the 20th International Conference on Computer Processing of Oriental Languages, Shenyang, China, 2003, pp. 1503-1509.

[13] Oznur Kirmemis and Gulen Toker, Text Categorization Using k-Nearest Neighbour Classification, Middle East Technical University Computer Engineering Department.

[14] Imad Rahal,William Perrizo, An Optimized Approach for KNN Text Categorization using P-trees, 2004 ACM Symposium on Applied Computing, 2004, pp. 613-617.

[15] P. Soucy, G. W. Mineau, A simple KNN algorithm for text categorization, Proceedings IEEE International Conference on Data Mining, 2001,pp. 647 – 648.

[16] Yang Y. and Liu X., A Re-examination of Text Categorization Methods. Proceedings of 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 1999,pp. 42-49.

[17] Thorsten Joachims, Text Categorization with Suport Vector Machines: Learning with Many Relevant Features, Proceedings of the 10th European Conference on Machine Learning,1998, pp. 137-142.

[18] Li Baoli, Chen Yuzhong, and Yu Shiwen, A Comparative Study on Automatic Categorization Methods for Chinese Search Engine. In: Proceedings of the Eighth Joint International Computer Conference. Hangzhou: Zhejiang University Press, 2002, pp. 117-120.

[19] Manning C. D. and Schutze H., Foundations of Statistical Natural Language Processing. 1999, Cambridge: MIT Press.

[20] Mengen Chen, Xiaoming Jin, Short Text Classification Improved by Learning Multi-Granularity Topics. Proceeding IJCAI'11 Proceedings of the Twenty-Second international joint conference on Artificial Intelligence, 2011,Vol(3), pp. 1776-1781.

[21] X.Wang, R.Chen. Short Text Classification Using Wikipedia Concept Based Document Representation. International Conference on Information Technology and Applications (ITA), 2013, pp. 471-474.

[22] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks, Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies, 2015.

[23] Rie Johnson and Tong Zhang.Semi-supervised convolutional neural networks for text categorization via region embedding. Advances in Neural Information Processing Systems, 2015, pp. 919-927.

[24] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, Maryland, 2014 Vol(1), pp. 655-665.

[25] Maxime Oquab, Leon Bottou, Ivan Laptev, Josef Sivic,Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1717-1724.

[26] Dan C. Cires¸an, Ueli Meier,Flexible, High Performance Convolutional Neural Networks for Image Classification,Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence,2011, pp.1237-1242.

[27] Karen Simonyan, Andrew Zisserman,Very Deep Convolutional Networks for Large-Scale Image Recognition,International Conference on Learning Representations,2015, available: http://arxiv.org/abs/1409.1556

[28] Y. Kim. Convolutional neural networks for sentence classification. Proceedings of EMNLP, 2014.

[29] D. E. Johnson,F. J. Oles, T. Zhang, T. Goetz, A decision-tree-based symbolic rule induction system for text categorization, IBM Systems Journal, 2002, Vol(41), pp. 428 – 437.

[30] Opinion polarity detection subtask of the MPQA dataset. Available : http://mpqa.cs.pitt.edu/

[31] Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao, Recurrent Convolutional Neural Networks for Text Classification, Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence,2015, pp. 2267-2273.

[32] Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner, Isabell M. Welpe, Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media, 2010, pp. 178-185.

[33] J. Bollen, A. Pepe, and H. Mao, “Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena,” Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM 2011), Barcelona, Spain, 2011, pp. 450-453.

[34] K. Durant and M. Smith, Predicting the political sentiment of web log posts using supervised machine learning techniques coupled with feature selection, Proceeding of the 8th International Workshop on Knowledge Discovery on the Web (WebKDD). Springer, 2007, pp. 187-206.

[35] N. Diakopoulos and D. Shamma, “Characterizing debate performance via aggregated Twitter sentiment,” in Proc. of the 28th International Conference on Human Factors in Computing Systems (CHI). ACM, 2010, pp. 1195-1198.

[36] Adam Bermingham, Alan F. Smeaton, On using Twitter to monitor political sentiment and predict election results, AI meets Psychology (SAAIP) Workshop at the International Joint Conference for Natural Language Processing (IJCNLP), 2011, pp. 2-10.

[37] Pennebaker, J.; Chung, C.; and Ireland, M. 2007. The development and psychometric properties of LIWC2007. Austin, TX, 2007.

[38] Erik Tjong Kim Sang,Johan Bos, Predicting the 2011 dutch senate election results with Twitter, Proceedings of the Workshop on Semantic Analysis in Social Media,2011,pp. 53-60.

[39] Akshat Bakliwal, Jennifer Foster, etal, Sentiment Analysis of Political Tweets: Towards an Accurate Classifier, Proceedings of the Workshop on Language in Social Media ,2013, pp. 49-58.

[40] Michael D. Conover, Bruno Goncalves, etal, Predicting the Political Alignment of Twitter Users, 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011, pp. 192-199.

[41] Sheetal Kaul, Agenda detector: labeling tweets with political policy agenda, Graduate Theses and Dissertations Digital Repository @ Iowa State University 2015.

[42] StringToWordVector in Weka Available: http://weka.sourceforge.net/doc.dev/weka/filters/unsupervised/attribute/StringToWordVector.html

[43] TF-IDF, https://en.wikipedia.org/wiki/Tf–idf

APPENDIX A

SORUCE CODE OF REST API

```
//SimilarState.java
package edu.iastate.tweet;
import java.util.ArrayList;
import javax.xml.bind.annotation.XmlAccessOrder;
import javax.xml.bind.annotation.XmlAccessorOrder;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

@XmlType(name = "SimilarState", propOrder = {
           "state",
           "year",
           "month",
           "k",
           "similarity"
       })

public class SimilarState {
       String State;
       String year;
       String month;
       int k;
       ArrayList<SimState> similarity;
       public String getState() {
              return State;
       }
       public void setState(String state) {
              State = state;
       }
       public String getYear() {
              return year;
       }
       public void setYear(String year) {
              this.year = year;
       }
       public String getMonth() {
              return month;
       }
       public void setMonth(String month) {
              this.month = month;
       }
       public int getK() {
              return k;
       }
       public void setK(int k) {
              this.k = k;
       }
       public ArrayList<SimState> getSimilarity() {
              return similarity;
```

```
	}
	public void setSimilarity(ArrayList<SimState> similarity) {
		this.similarity = similarity;
	}
}


//SimState.java
package edu.iastate.tweet;

import javax.xml.bind.annotation.XmlAccessOrder;
import javax.xml.bind.annotation.XmlAccessorOrder;
import javax.xml.bind.annotation.XmlType;
public class SimState implements Comparable<SimState>{
	double cosineSimilarity;
	String StatePair;
	public String getCurrMonth() {
		return currMonth;
	}

	public void setCurrMonth(String currMonth) {
		this.currMonth = currMonth;
	}

	public String getCurrYear() {
		return currYear;
	}

	public void setCurrYear(String currYear) {
		this.currYear = currYear;
	}

	String currMonth;
	String currYear;
	double[] vector;

	public SimState() {
		vector = new double[20];
	}
	public double getCosineSimilarity() {
		return cosineSimilarity;
	}
	public void setCosineSimilarity(double cosineSimilarity) {
		this.cosineSimilarity = cosineSimilarity;
	}
	public String getStatePair() {
		return StatePair;
	}
	public void setStatePair(String statePair) {
		StatePair = statePair;
	}
	public double[] getVector() {
		return vector;
	}
	public void setVector(double[] vector) {
```

```
			this.vector = vector;
		}
		@Override
		public int compareTo(SimState o1) {
			// TODO Auto-generated method stub
			if(this.cosineSimilarity==o1.cosineSimilarity)
				return 0;
			else if(this.cosineSimilarity>o1.cosineSimilarity)
				return -1;
			else
				return 1;
		}
}


//SimilarTopic.java
package edu.iastate.tweet;

import java.util.ArrayList;
import java.util.HashMap;

import javax.xml.bind.annotation.XmlAccessOrder;
import javax.xml.bind.annotation.XmlAccessorOrder;
import javax.xml.bind.annotation.XmlType;

class StateTopic implements Comparable<StateTopic>{
		String year;
		String month;
		String state;
		double percentTweets;

		public String getState() {
			return state;
		}
		public void setState(String state) {
			this.state = state;
		}
		public String getYear() {
			return year;
		}
		public void setYear(String year) {
			this.year = year;
		}
		public String getMonth() {
			return month;
		}
		public void setMonth(String month) {
			this.month = month;
		}
		public double getPercentTweets() {
			return percentTweets;
		}
		public void setPercentTweets(double percentTweets) {
			this.percentTweets = percentTweets;
		}
```

```
	@Override
	public int compareTo(StateTopic o) {
		// TODO Auto-generated method stub
		if(this.percentTweets==o.percentTweets)
			return 0;
		else if(this.percentTweets<o.percentTweets)
			return 1;
		else
			return -1;
	}

}
@XmlType(propOrder = {
		"topic",
		"year",
		"month",
		"k",
		"topstates"
	})

public class SimiliarTopic {
	String topic;
	String year;
	String month;
	int k;
	ArrayList<StateTopic> topstates;

	public ArrayList<StateTopic> getTopstates() {
		return topstates;
	}
	public void setTopstates(ArrayList<StateTopic> topstates) {
		this.topstates = topstates;
	}
	public int getK() {
		return k;
	}
	public void setK(int k) {
		this.k = k;
	}
	public String getTopic() {
		return topic;
	}
	public void setTopic(String topic) {
		this.topic = topic;
	}
	public String getYear() {
		return year;
	}
	public void setYear(String year) {
		this.year = year;
	}
	public String getMonth() {
		return month;
	}
	public void setMonth(String month) {
```

```
			this.month = month;
		}
}

// tweetSimilarState.java
package edu.iastate.tweet;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import javax.ws.rs.DELETE;
import javax.ws.rs.DefaultValue;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

@Path("/v1/policyagendas/tweets")
//This class will be used for getting similar States and similar topics, and the
end point is specified in this class.
public class TweetsSimilarState
{
	@GET
	@Path("/similarStates")
	@Produces(MediaType.APPLICATION_JSON)
    // this function is used for getting similar states, the parameters of this
end point are set here.
    public SimilarState findSimilarState(@QueryParam("state") String
state,@QueryParam("year") Integer year,@QueryParam("month") Integer
month,@DefaultValue("5") @QueryParam("k") int k,@DefaultValue("false")
@QueryParam("distinct") boolean distinct,@DefaultValue("12")
@QueryParam("limitmonth") Integer limitmonth)
	{
		DatabaseOperations dbOperations=new DatabaseOperations();
		return dbOperations.getSimilarStates(k,year, month,
state,distinct,limitmonth);
	}

	@GET
	@Path("/similarTopics")
	@Produces(MediaType.APPLICATION_JSON)
    // this function is used for getting similar topic, the parameters of this
end point are set here.
    public SimiliarTopic findSimilarTopic(@QueryParam("topic") Integer
topic,@QueryParam("year") Integer year,@QueryParam("month") Integer
month,@DefaultValue("5") @QueryParam("k") int k)
	{
		DatabaseOperations dbOperations=new DatabaseOperations();
		return dbOperations.getSimilarTopic(k,topic,year, month);

	}
}
```

```
//DatabaseOperations.java
package edu.iastate.tweet;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;

public class DatabaseOperations
{
	public SimiliarTopic getSimilarTopic(int k,int topic,int year, int month)
	{
		if(k<=0)
		    k=5;
		String SQL = null;
		String abbreviationSQL = null;
		String abbreviation =null;
		int topicNumber = topic;

		SimiliarTopic res = new SimiliarTopic();

		res.setK(k);
		res.setTopic(String.valueOf(topic));
		res.setMonth(String.valueOf(month));
		res.setYear(String.valueOf(year));

		SQL = "SELECT distinct state FROM pd.tweetsforrest";
		ArrayList<String> stateList =executeQueryForString(SQL);
		ArrayList<StateTopic> statetopic = new ArrayList<StateTopic>();
		ArrayList<StateTopic> topstates = new ArrayList<StateTopic>();
		for(String s : stateList)
		{
			String currAbbr = null;
			abbreviationSQL="SELECT abbreviation FROM pd.states where name
= '"+s+"'";
			currAbbr=executeQueryForGettingAbbrivation(abbreviationSQL);
			StateTopic currTopic = new StateTopic();
			HashMap<String,Integer> map = new HashMap<String,Integer>();
			SQL = "SELECT actual_major as topicNumber,count(actual_major)
as numberOfTweets FROM pd.tweetsforrest where actual_major!=0 and state='"+s+"'
and year='"+String.valueOf(year)+"' and month='"+String.valueOf(month)+"' group
by topicNumber";
			map = executeQueryForGettingStateTopics(SQL);

			currTopic.setMonth(String.valueOf(month));
			currTopic.setYear(String.valueOf(year));
```

```
                    double[] vector = new double[20];

                    Iterator it = map.keySet().iterator();
                    int totalTweetsNumber = 0;
                    while(it.hasNext())
                    {

                           String topicString = (String)it.next();

                           int tNumber = Integer.parseInt(topicString);
                           if(tNumber>11)
                                  tNumber = tNumber-1;
                           tNumber=tNumber-1;
                           int tweetsNumber = map.get(topicString);
                           totalTweetsNumber+=tweetsNumber;
                           vector[tNumber]=(double)tweetsNumber;
                    }

                    for(int i=0;i<vector.length;i++)
                    {
                           if(totalTweetsNumber==0)
                                  vector[i]=0;
                           else

      vector[i]=100*vector[i]/(double)totalTweetsNumber;
                    }
                    int tempTopicNumber=0;
                    if(topicNumber>11)
                           tempTopicNumber=topicNumber-1;
                    else
                           tempTopicNumber=topicNumber;
                    tempTopicNumber-=1;
                    DecimalFormat df = new DecimalFormat("#.##");

      currTopic.setPercentTweets(Double.parseDouble(df.format(vector[tempTopicNum
ber])));

      //currTopic.setPercentTweets(Double.parseDouble(String.format("%.2f",vector
[tempTopicNumber])));
                    currTopic.setState(currAbbr);
                    topstates.add(currTopic);
             }

             //Collections.sort(topstates);
             Collections.sort(topstates);
             ArrayList<StateTopic> temp = new ArrayList<StateTopic>();
             for(int i=0;i<k;i++)
             {
                    temp.add(topstates.get(i));
             }


             res.setTopstates(temp);

             return res;
```

```
	}
	public SimilarState getSimilarStates(int k,int year, int month, String
state,boolean distinct,int limitmonth)
	{
		if(k<=0)
			k=5;
		if(limitmonth<=0)
			limitmonth=12;
		String SQL = null;
		String abbreviationSQL = null;
		String abbreviation =null;
		SimilarState res = new SimilarState();
		res.setK(k);
		res.setMonth(String.valueOf(month));
		if(state.length()==2)
		{
			res.setState(state);
			abbreviation=state;
		}
		else
		{
			abbreviationSQL="SELECT abbreviation FROM pd.states where name
= '"+state+"'";

	abbreviation=executeQueryForGettingAbbrivation(abbreviationSQL);
		}
		res.setYear(String.valueOf(year));
		SQL = "SELECT distinct state FROM pd.tweetsforrest";
		ArrayList<String> stateList =executeQueryForString(SQL);
		ArrayList<SimState> simState = new ArrayList<SimState>();
		double[] stateVector = new double[20];
		System.out.println("limit month is: "+limitmonth);
		int count=limitmonth;
		for(String s : stateList)
		{
			String currAbbr = null;
			abbreviationSQL="SELECT abbreviation FROM pd.states where name
= '"+s+"'";
			currAbbr=executeQueryForGettingAbbrivation(abbreviationSQL);


			int currMonth= month;
			int currYear = year;
			for(int l=0;l<count;l++)
			{
				SimState currState = new SimState();
				SQL = "SELECT actual_major as
topicNumber,count(actual_major) as numberOfTweets FROM pd.tweetsforrest where
actual_major!=0 and state='"+s+"' and year='"+String.valueOf(currYear)+"' and
month='"+String.valueOf(currMonth)+"' group by topicNumber";
				HashMap<String,Integer> map = new
HashMap<String,Integer>();
				map = executeQueryForGettingStateTopics(SQL);
				currState.setStatePair(abbreviation+","+currAbbr);
				double[] vector = new double[20];
```

```
                        Iterator it = map.keySet().iterator();
                        int totalTweetsNumber = 0;
                        while(it.hasNext())
                        {
                               String topicString = (String)it.next();
                               int topicNumber = Integer.parseInt(topicString);
                               if(topicNumber>11)
                                      topicNumber = topicNumber-1;
                               topicNumber=topicNumber-1;
                               int tweetsNumber = map.get(topicString);
                               totalTweetsNumber+=tweetsNumber;
                               vector[topicNumber]=(double)tweetsNumber;
                        }
                        for(int i=0;i<vector.length;i++)
                        {
                               if(totalTweetsNumber==0)
                                      vector[i]=0.00;
                               else
                               {


      vector[i]=100*vector[i]/(double)totalTweetsNumber;
                                      DecimalFormat df = new
DecimalFormat("#.##");

      vector[i]=Double.parseDouble(df.format(vector[i]));
                                      }
                        }
                        currState.setCurrMonth(String.valueOf(currMonth));
                        currState.setCurrYear(String.valueOf(currYear));
                        currState.setVector(vector);
                        simState.add(currState);
                        if(currAbbr.equals(abbreviation) && currMonth==month &&
currYear == year)
                        {

                               for(int i=0;i<vector.length;i++)
                                      stateVector[i]=vector[i];
                        }
                        currMonth=currMonth-1;
                        if(currMonth==0)
                        {
                               currMonth = 12;
                               currYear = currYear-1;
                        }
                  }
            }
            //System.out.println(simState.size());
            for(int i=0;i<simState.size();i++)
            {
                  double[] sVector = simState.get(i).getVector();
                  double similarity = calcCosineSimilarity(stateVector,sVector);
                  DecimalFormat df = new DecimalFormat("#.##");
                  similarity=Double.parseDouble(df.format(similarity));
```

```
                    System.out.println(simState.get(i).getStatePair()+" |
"+simState.get(i).getCurrYear()+" | "+simState.get(i).getCurrMonth()+" |
"+similarity);
                    simState.get(i).setCosineSimilarity(similarity);
             }
             Collections.sort(simState);
             ArrayList<SimState> temp = new ArrayList<SimState>();
             if(distinct)
             {
                    int sCount = 0;
                    HashSet<String> set = new HashSet<String>();
                    for(int i=0;i<simState.size();i++)
                    {
                           if(sCount>=k)
                                  break;
                           String statePair = simState.get(i).getStatePair();
                           if(!set.contains(statePair))
                           {
                                  sCount++;
                                  set.add(statePair);
                                  temp.add(simState.get(i));
                           }
                    }
             }else
             {
                    for(int i=0;i<k;i++)
                    {
                           temp.add(simState.get(i));
                    }
             }
             res.setSimilarity(temp);

             return res;
      }
      private String executeQueryForGettingAbbrivation(String sql)
      {
             String res=null;
             Connection connection=null;
             try
             {
                    DbConnection database= new DbConnection();
                    connection = database.getConnection();
                    System.out.println(sql);
                    if(connection==null)
                           System.out.println("Connections is null");
                    PreparedStatement ps = connection.prepareStatement(sql);
                    ResultSet rs = ps.executeQuery();
                    //res=rs.getString(1);
                    while(rs.next())
                    {
                           res=rs.getString(1);
                    }
             }

             catch(Exception e)
```

```
		{
			e.printStackTrace();
		}
		finally
		{
			try
			{
				connection.close();
			}
			catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
		return res;
	}
	private double calcCosineSimilarity(double[] a,double[] b)
	{
		double res=0;
		double ab=0,aa=0,bb=0;
		boolean same=true;
		for(int i=0;i<a.length;i++)
		{
			ab+=a[i]*b[i];
			aa+=a[i]*a[i];
			bb+=b[i]*b[i];
			if(a[i]!=b[i])
				same=false;
		}
		if(same)
			return 1;
		if(aa==0 || bb==0)
			return 0;
		res = ab/(Math.sqrt(aa)*Math.sqrt(bb));
		return res;
	}
```

APPENDIX B

TWITTER HANDLES USED IN OUR STUDY

handle	State
@AZSenateGOP	Arizona
@AZSenateGOP	Arizona
@SenatorAbleser	Arizona
@Allen4AZSenate	Arizona
@LelaAlstonAz	Arizona
@NancyBarto	Arizona
@Barton4Az	Arizona
@CarlyleBegay	Arizona
@reginaldbolding	Arizona
@SonnyBorrelli	Arizona
@Rusty4Congress	Arizona
@BoyerAZ	Arizona
@RepMarkCardenas	Arizona
@HeatherCarterAZ	Arizona
@Charlene4House	Arizona
@recobbforazrep	Arizona
@douglas_coleman	Arizona
@Dalessandro4AZÂ Protected Tweets	Arizona
@jeffdial	Arizona
@AdamDriggs	Arizona
@SteveFarleyAZ	Arizona
@MarkFinchem	Arizona
@DrRandyFriese	Arizona
@RosannaGabaldon	Arizona
@katiehobbs	Arizona
@RepLarkin	Arizona
@DebbieLesko	Arizona
@Sen_LindaÂ	Arizona
@PhilLovas	Arizona
@StefanieMach	Arizona
@McCuneDavis4D30	Arizona
@KateMcGeeAZ	Arizona
@SenBarbMcGuire	Arizona
@mendezforaz	Arizona
@JDMesnard	Arizona
@DrEricMeyer	Arizona

@SenRobertMeza	Arizona
@Miranda4Arizona	Arizona
@ElectDarin	Arizona
@SteveMontenegro	Arizona
@JillNorgaard	Arizona
@RepOtondo	Arizona
@votewarren	Arizona
@SenStevePierce	Arizona
@pratt4az	Arizona
@RepQuezada29	Arizona
@Rios_Rebecca	Arizona
@AZ_Rep_Sherwood	Arizona
@SteeleForHouse	Arizona
@_DavidStevens	Arizona
@azrepbobthorpe	Arizona
@KellyTownsend11	Arizona
@MichelleUgenti	Arizona
@cecivelasquez_	Arizona
@kelliwardaz	Arizona
@JeffWeninger	Arizona
@BruceWheelerAZ	Arizona
@bob_worsley	Arizona
@KimberlyYeeAZ	Arizona
@FLSenateGOP	Florida
@FlaDems	Florida
@FLSenate	Florida
@FloridaGOP	Florida
@RepJanetAdkins	Florida
@Larry_Ahern	Florida
@RepAlbritton	Florida
@SenatorAltman	Florida
@Artiles118	Florida
@BryanAvilaFL	Florida
@AaronPBean	Florida
@lizbethkb	Florida
@loriberman	Florida
@HalseyBeshears	Florida
@mbileca	Florida
@RepJimBoyd	Florida
@JeffreyBrandes	Florida
@oscarjb2	Florida
@jasonbrodeur	Florida
@DougBroxson	Florida

@DwightBullard	Florida
@ColleenLBurton	Florida
@mattcaldwell_fl	Florida
@RepCampbellD108	Florida
@Gwyn_ClarkeReed	Florida
@ClemensFL	Florida
@NeilCombee	Florida
@richardcorcoran	Florida
@Cortes_FL	Florida
@CortesBob	Florida
@fredcostello	Florida
@SteveCrisafulli	Florida
@RepJanetCruz	Florida
@CharlieDeanSD5	Florida
@SenNancyDetert	Florida
@MarioDB	Florida
@josefelixdiaz	Florida
@RepMannyDiazJr	Florida
@RepBobbyDuBose	Florida
@DwightDudleyFL	Florida
@DaneEagle	Florida
@EricEisnaugle	Florida
@gregevers	Florida
@jay_fant	Florida
@RepFitzenhagen	Florida
@anitere_flores	Florida
@ErikFresenFL	Florida
@RepFullwood	Florida
@mattgaetz	Florida
@BillGalvano	Florida
@SenReneGarcia	Florida
@RepJosephGeller	Florida
@GibsonForSenate	Florida
@TGoodsonFLHD50	Florida
@JamesGrantFL	Florida
@denisegrimsley	Florida
@RepBillHager	Florida
@Gayle_Harrell	Florida
@Shawnfor63	Florida
@Hays2010	Florida
@DorothyHukill	Florida
@BlaiseIngoglia	Florida
@repclayingram	Florida

@Kristin_Jacobs	Florida
@ArtheniaJoyner	Florida
@DavidMKerner	Florida
@ChrisLatvala	Florida
@JackLatvala	Florida
@SenatorJohnLegg	Florida
@MaryLynnMagar	Florida
@GwenMargolis	Florida
@debbie_mayfield	Florida
@Rep_McBurney	Florida
@kionnemcghee	Florida
@larrymetz	Florida
@MillerFlorida14	Florida
@AmandaMurphyFL	Florida
Â ?@joenegronflÂ	Florida
@RepJNunez	Florida
@marleneotoole	Florida
@MarkPafford	Florida
@Kathleen4SWFL	Florida
@keithperry21	Florida
@KathleenMPeters	Florida
@CaryPigman	Florida
@PilonForFlorida	Florida
@TeamPlakon	Florida
@voteCoachP	Florida
@ElectLizPorter	Florida
@BPowellforRep	Florida
@rep_pritchett	Florida
@JakeRaburn	Florida
@KevinRader	Florida
@HollyRaschein	Florida
@DanRaulersonFL	Florida
@RepMichelle	Florida
@Paul_Renner	Florida
@david4florida	Florida
@isayray	Florida
@JoseJavierJJR	Florida
@PatRooneyJr	Florida
@darrylrouson	Florida
@MariaSachs	Florida
@dsantiago457	Florida
@SenDavidSimmons	Florida
@WiltonSimpson	Florida

@IrvSlosberg	Florida
@SenChrisSmith	Florida
@JimmieTSmith	Florida
@SenDarrenSoto	Florida
@RossSpano	Florida
@ChrisSprowls	Florida
@kellistargel	Florida
@gregsteube	Florida
@Vote4Cyndi	Florida
@RepCharlieStone	Florida
@John_Thrasher	Florida
@VicTorres_FL	Florida
@RepCTrujillo	Florida
@TeamRepWatson	Florida
@RepAlanWilliams	Florida
@RitchWorkman	Florida
@repdanayoung	Florida
@ILSRCC	Illinois
@ILSenateGOP	Illinois
@EdwardJAcevedo	Illinois
@pamelaalthoff	Illinois
@StateRepAmmons	Illinois
@Andersson4Rep	Illinois
@AndradeRep40	Illinois
@RepArroyo	Illinois
@jasonbarickman	Illinois
@scott_m_bennett	Illinois
@49JBT	Illinois
@danielbiss	Illinois
@Dan_Brady	Illinois
@Brady4Illinois	Illinois
@Melinda4Senate	Illinois
@JohnCabello	Illinois
@RepKellyCassidy	Illinois
@lindachapalavia	Illinois
@jamesclayborne	Illinois
@SenatorConnelly	Illinois
@StateRepCrespo	Illinois
@VPTommy	Illinois
@electCD	Illinois
@SRDrury	Illinois
@SenatorDanDuffy	Illinois
@RepKenDunkin	Illinois

@Repevans33	Illinois
@StateRepSara	Illinois
@RepLauraFine	Illinois
@repcurrie	Illinois
@RepLaShawnFord	Illinois
@RepJackFranks	Illinois
@RobynGabel	Illinois
@WillGuzzardi	Illinois
@HarmonForSenate	Illinois
@RepDavidHarris	Illinois
@Napoleon_Harris	Illinois
@HastingsforIL	Illinois
@RepHernandez	Illinois
@SenatorHolmes	Illinois
@SenatorHunter	Illinois
@ToiHutchinson	Illinois
@JeanneIves	Illinois
@il29cand	Illinois
@Senator14	Illinois
@RepDwightKay	Illinois
@RepKifowit	Illinois
@KoehlerIL	Illinois
@DanKotowski	Illinois
@lahooddarin	Illinois
@StateRepLouLang	Illinois
@LLCoolK_4	Illinois
@DaveLuechtefeld	Illinois
@AndyManar	Illinois
@StateRepManley	Illinois
@ILSenMartinez	Illinois
@robertmartwick	Illinois
@mccann_sam	Illinois
@SenatorMcCarter	Illinois
@kmcconnaughay33	Illinois
@PatMcGuire43	Illinois
@JohnGMulroe	Illinois
@SenMattMurphy	Illinois
@ElaineNekritz	Illinois
@noland4congress	Illinois
@chrisnybo	Illinois
@JimOberweis	Illinois
@RepPritchard	Illinois
@chrisradogno	Illinois

@KwameRaoul	Illinois
@PamelaForUs	Illinois
@SenatorRezin	Illinois
@DaleRighter	Illinois
@RepBobRita	Illinois
@SenChapinRose	Illinois
@RonSandack	Illinois
@SenatorSandoval	Illinois
@SenSilverstein	Illinois
@ElgieSims	Illinois
@Smiddy4IL71	Illinois
@repsosnowski	Illinois
@StateRepSoto	Illinois
@SteveStadelman	Illinois
@HeatherSteans	Illinois
@Sen1Dave	Illinois
@RepThapedi	Illinois
@reptryon	Illinois
@ArthurLTurner	Illinois
@mikeunes	Illinois
@SenatorVanPelt	Illinois
@LitesaWallace	Illinois
@GrantWehrli	Illinois
@RepChrisWelch	Illinois
@RepAnnWilliams	Illinois
@Sam4Rep	Illinois
@IowaSenate	Iowa
@IASenateGOP	Iowa
@akoabdulsamad	Iowa
@ChazAllen2013	Iowa
Anderson Marti	Iowa
@ChipBaltimoreIA	Iowa
@BruceBearinger	Iowa
@LizBennettIowa	Iowa
@RepDebBerry	Iowa
@SenatorBertrand	Iowa
@showbiz8	Iowa
@JoeBolkcom	Iowa
@darrbran	Iowa
@JoshuaByrnes	Iowa
@GaryCarlsonGOP	Iowa
@VoteJakeChapman	Iowa
@petercownie	Iowa

@JeffDanielsonÂ	Iowa
@SenatorDix	Iowa
@NancyDunkel	Iowa
@RandyFeenstra	Iowa
@Abby4IowaHouse	Iowa
@JohnForbes4IA	Iowa
@RepRuthAGaines	Iowa
@PatGrassley	Iowa
@ChrisHagenow	Iowa
@HallForIowa	Iowa
@RepCurtHanson	Iowa
@hart4senate	Iowa
@DaveHeaton85	Iowa
@jake_highfill	Iowa
@SenatorRobHogg	Iowa
@RepDaveJacoby	Iowa
@PJochum	Iowa
@rjorgensen79	Iowa
@kaufmannGOP	Iowa
@electkelley	Iowa
@Kevinkoester	Iowa
@bkressig	Iowa
@LandonForHouse	Iowa
@marymascher	Iowa
@LizMathis1	Iowa
@CharlieMcConkey	Iowa
Â @mccoyforsenate	Iowa
@brianjmeyer	Iowa
@helenmiller49	Iowa
@janet4iowa	Iowa
@dawnpettengill	Iowa
@RepPrichard	Iowa
@RizerForHouse	Iowa
@waltrogersforIA	Iowa
@Ruff4StateRep	Iowa
@SenSchneider	Iowa
@Jason_Schultz	Iowa
@marksmithiowa	Iowa
Â @SenRobySmith	Iowa
@Senator_Sodders	Iowa
@StaedArt	Iowa
@QStanerson	Iowa
@SSteckman	Iowa

@SallyStutsman	Iowa
@RobTaylorIowa	Iowa
@phyllisthede	Iowa
@jackwhitver	Iowa
@CindyWinckler	Iowa
@MattWindschitl	Iowa
@zaunforcongress	Iowa
@DanZumbach	Iowa
@missourigop	Missouri
@NebraskaDems	Nebraska
@NEGOP	Nebraska
@katejbolz	Nebraska
@LydiaBrasch	Nebraska
@ColbyCoash	Nebraska
@joni_craighead	Nebraska
@SenCrawford	Nebraska
@SenLauraEbke	Nebraska
captaincorn	Nebraska
@wnta12	Nebraska
@SenatorKenHaar	Nebraska
@MattHansenNE	Nebraska
@burkeharr	Nebraska
@senhilkemann	Nebraska
@SaraForNebraska	Nebraska
@BillKintner	Nebraska
@RickKolowski	Nebraska
@KoltermanforLeg	Nebraska
@SenatorBobKrist	Nebraska
@JohnKuehnDVM	Nebraska
@TysonLarson	Nebraska
@SenBLindstrom	Nebraska
@votemccollister	Nebraska
@BeauRMcCoy	Nebraska
@heathmello	Nebraska
@Adam_Morfeld	Nebraska
@JohnMurante	Nebraska
@NordquistNE	Nebraska
@Patty4Nebraska	Nebraska
@Senator_Riepe	Nebraska
@ken_schilz	Nebraska
@SenSchnoor	Nebraska
@JohnStinner	Nebraska
@DanWatermeier	Nebraska

@NYSenDems	New York
@NewYorkGOP	New York
@SoDakDems	South Dakota
@sdgop	South Dakota
@VoteBolin	South Dakota
@angiebuhl	South Dakota
@Krisdistrict32	South Dakota
@BlakeCurd	South Dakota
@FredDeutsch	South Dakota
@jasonfrerichs	South Dakota
@BrockGreenfield	South Dakota
@donhaggar	South Dakota
@jennahaggar	South Dakota
@haverly_terri	South Dakota
@PhyllisHeineman	South Dakota
@riedholien	South Dakota
@berniehunhoff	South Dakota
@RepDanKaiser	South Dakota
@kevinck04	South Dakota
@IsaacLatterell	South Dakota
@danlederman	South Dakota
@micksd	South Dakota
@JeffJeffmonroe	South Dakota
@scottmunsterman	South Dakota
@RepStaceNelson	South Dakota
@anovstrup	South Dakota
@dnovstrup	South Dakota
@parsley4senate	South Dakota
@PartridgeSD	South Dakota
@DebPetersForSD	South Dakota
@BruceRampelberg	South Dakota
@SDSenLeader	South Dakota
@RayRingforHouse	South Dakota
@LeeSchoenbeck	South Dakota
@SoholtDeb	South Dakota
@SenatorSolano	South Dakota
@JimStalzer	South Dakota
@Billie_Sutton	South Dakota
@larrytidemann	South Dakota
@SteveWestra	South Dakota
@wiikfor4	South Dakota
@MarkWilladsen	South Dakota
@MathewWollmann	South Dakota

@LarryZikmund	South Dakota
@WisDems	Wisconsin
scotte_allen	Wisconsin
TylerAugust	Wisconsin
JoanBallweg	Wisconsin
PeterWBarca	Wisconsin
TheOtherMandela	Wisconsin
TereseBerceau	Wisconsin
JanetBewley4WI	Wisconsin
RepJillBillings	Wisconsin
RepJanel	Wisconsin
RepRobBrooks	Wisconsin
Jbrostoff	Wisconsin
TimCarpenterMKE	Wisconsin
Considine1Dave	Wisconsin
RepDaveCraig	Wisconsin
MaryCzaja	Wisconsin
SenDarling	Wisconsin
JonErpenbach	Wisconsin
PaulFarrowWi	Wisconsin
SenFitzgerald	Wisconsin
RepGenrich	Wisconsin
Goyke4Assembley	Wisconsin
RickGudex	Wisconsin
SenatorDaveHansen	Wisconsin
NikiyaQharris	Wisconsin
standwithSheila	Wisconsin
daveheaton85	Wisconsin
RepHesselbein	Wisconsin
GordonHintz	Wisconsin
CodyHorlacher	Wisconsin
Robert_Hutton	Wisconsin
JohnJagler	Wisconsin
AdamJarchow28	Wisconsin
RepJorgensen	Wisconsin
robbkahl	Wisconsin
chriskapenga	Wisconsin
samanthakerkman	Wisconsin
RepKessler	Wisconsin
DanKnodlforWI	Wisconsin
RepDeanKnudson	Wisconsin
DaleKooyenga	Wisconsin
JesseforWI	Wisconsin

Skrug75	Wisconsin
MikeKuglitsch	Wisconsin
BobKulp	Wisconsin
ChrisJLarson	Wisconsin
JulieMLassa	Wisconsin
RepAmy31	Wisconsin
SenMarklein	Wisconsin
RepCoryMason	Wisconsin
WISenatorMiller	Wisconsin
NickMilroy	Wisconsin
terrymoulton	Wisconsin
davemurphy56	Wisconsin
AdamNeylon	Wisconsin
VoteToddNovak	Wisconsin
rep89	Wisconsin
RepTodOhnstad	Wisconsin
petersentweets	Wisconsin
Jerry4senate	Wisconsin
RepSondy	Wisconsin
RepDanielRiemer	Wisconsin
WIRepKeithRipp	Wisconsin
SenRisser	Wisconsin
RepJessie	Wisconsin
MikeRohrkaste	Wisconsin
Roth4wisconsin	Wisconsin
RepSanfelippo	Wisconsin
Repsargent	Wisconsin
michaelschraa	Wisconsin
RepShankland	Wisconsin
SenShilling	Wisconsin
RepChrisSinicki	Wisconsin
RepSpiros	Wisconsin
RepSpreitzer	Wisconsin
repsteffen	Wisconsin
jimsteineke	Wisconsin
Amanda4Assembly	Wisconsin
LisaSubeck	Wisconsin
sweak1	Wisconsin
ChrisTaylorWI	Wisconsin
SenTaylor	Wisconsin
Thiesfeldt	Wisconsin
tittlassembly	Wisconsin
Tranel4assembly	Wisconsin

VineoutK	Wisconsin
VorpagelFor27	Wisconsin
SpeakerVos	Wisconsin
LeahVukmir	Wisconsin
DanaJwachs	Wisconsin
Vanwaggaard	Wisconsin
RepWeatherston	Wisconsin
RepJocasta	Wisconsin

APPENDIX C

COMPARISON OF THE TOP FIVE TOPICS

December 2014
Iowa:
predict result(top 7):[6=274, 0=213, 1=174, 3=162, 5=90, 11=89, 19=64]
original result(top 7):[6=257, 0=217, 3=172, 1=163, 5=93, 19=91, 11=75]

predict top 5: 0,6,1,3,5
original top 5: 6,0,3,1,5
Matched!!!

Nebraska:
predict result(top 7): [0=243, 3=118, 1=81, 5=75, 11=46, 6=27, 2=23]
original result(top 7):[0=165, 3=122, 1=84, 5=78, 11=50, 19=33, 6=30]

predict top 5: 0,3,1,5,11
original top 5: 0,3,1,5,11
Matched!!!

January 2015
Iowa:
predict result(top 7):[6=297, 0=296, 1=154, 3=153, 5=93, 11=75, 19=74]
original result(top 7):[6=277, 0=237, 3=162, 1=151, 5=96, 19=93, 11=77]

predict top 5: 0,6,1,3,5
original top 5: 6,0,3,1,5
Matched!!!

Nebraska:
predict result(top 7): [0=208, 3=126, 1=91, 5=74, 11=54, 6=42, 19=37]
original result(top 7):[0=178, 3=118, 1=94, 5=78, 11=51, 19=41, 6=39]

predict top 5: 0,3,1,5,11
original top 5: 0,3,1,5,11
Matched!!!

February 2015
Iowa:
predict result(top 7):[0=337, 6=278, 3=150, 1=144, 5=81, 11=73, 19=68]
original result(top 7):[6=283, 0=272, 3=161, 1=140, 19=91, 5=80, 11=67]

predict top 5: 0,6,3,1,5
original top 5: 6,0,3,1,19

top 4 Matched!!!

Nebraska:
predict result(top 7): [0=242, 3=144, 1=93, 5=81, 11=47, 6=26, 14=20, 2=20]
original result(top 7):[0=205, 3=134, 1=93, 5=77, 11=49, 6=27, 19=26, 2=25]

predict top 5: 0,3,1,5,11
original top 5: 0,3,1,5,11
Matched!!!

March 2015
Iowa:
predict result(top 7):[6=294, 0=285, 3=148, 1=133, 19=92, 5=88, 11=75]
original result(top 7):[6=283, 0=255, 3=147, 1=126, 5=98, 19=94, 11=74]

predict top 5: 6,0,3,1,19
original top 5: 6,0,3,1,5
top 4 Matched!!!

Nebraska:
predict result(top 7): [0=227, 3=140, 5=86, 1=70, 11=47, 6=37, 2=26]
original result(top 7):[0=174, 3=144, 5=76, 1=71, 11=42, 6=39, 19=36]

predict top 5: 0,3,1,5,11
original top 5: 0,3,1,5,11
Matched!!!

April 2015
Iowa:
predict result(top 7):[0=298, 6=290, 3=148, 1=124, 5=92, 19=91, 11=78]
original result(top 7):[6=273, 0=258, 3=148, 1=132, 5=92, 11=91, 19=83]

predict top 5: 0,6,3,1,5
original top 5: 6,0,3,1,5
Matched!!!

Nebraska:
predict result(top 7): [0=203, 3=136, 1=84, 5=56, 11=47, 6=30, 9=24]
original result(top 7):[0=179, 3=130, 1=76, 5=60, 11=49, 19=30, 9=27]

predict top 5: 0,3,1,5,11
original top 5: 0,3,1,5,11
Matched!!!

May 2015
Iowa:

predict result(top 7):[6=319, 0=318, 3=148, 1=134, 5=92, 11=70, 19=65]
original result(top 7):[6=307, 0=274, 3=164, 1=136, 5=99, 19=83, 11=65]

predict top 5: 6,0,3,1,5
original top 5: 6,0,3,1,5
Matched!!!

Nebraska:
predict result(top 7): [0=223, 3=146, 1=95, 5=65, 11=59, 6=27, 2=23]
original result(top 7):[0=186, 3=142, 1=100, 5=62, 11=54, 19=38, 6=27]

predict top 5: 0,3,1,5,11
original top 5: 0,3,1,5,11
Matched!!!

June 2015
Iowa:
predict result(top 7):[0=308, 6=274, 1=156, 3=147, 5=81, 19=78, 11=78]
original result(top 7):[6=275, 0=265, 1=144, 3=140, 19=98, 5=82, 11=78]

predict top 5: 0,6,1,3,5
original top 5: 6,0,1,3,19
top 4 Matched!!!

Nebraska:
predict result(top 7): [0=200, 3=136, 1=91, 5=88, 11=48, 6=26, 2=25]
original result(top 7):[0=171, 3=131, 1=86, 5=85, 11=47, 19=39, 6=31]

predict top 5: 0,3,1,5,11
original top 5: 0,3,1,5,11
Matched!!!

July 2015
Iowa:
predict result(top 7):[0=317, 6=273, 3=142, 1=122, 5=74, 11=71, 19=60]
original result(top 7):[0=260, 6=255, 3=141, 1=126, 19=84, 11=83, 5=70]

predict top 5: 0,6,3,1,5
original top 5: 6,0,3,1,19
Top 4 Matched!!!

Nebraska:
predict result(top 7): [0=215, 3=112, 1=99, 5=88, 11=49, 6=31, 19=28]
original result(top 7):[0=180, 3=114, 1=88, 5=84, 19=48, 11=43, 6=42]

predict top 5: 0,3,1,5,11

original top 5: 0,3,1,5,19
Top 4 Matched!!!

August 2015
Iowa:
predict result(top 7):[0=287, 6=286, 3=148, 1=142, 5=93, 11=84, 19=69]
original result(top 7):[0=282, 6=282, 3=139, 1=123, 5=92, 11=83, 19=67]

predict top 5: 0,6,3,1,5
original top 5: 6,0,3,1,5
Matched!!!

Nebraska:
predict result(top 7): [0=212, 3=121, 1=103, 5=94, 11=46, 19=24, 6=24]
original result(top 7):[0=171, 3=115, 1=96, 5=82, 11=52, 19=38, 2=26]

predict top 5: 0,3,1,5,11
original top 5: 0,3,1,5,11
Matched!!!

September 2015
Iowa:
predict result(top 7):[0=309, 6=275, 3=166, 1=116, 5=95, 11=78, 19=69]
original result(top 7):[6=274, 0=253, 3=164, 1=121, 5=91, 19=79, 11=74]
predict top 5: 0,6,3,1,5
original top 5: 6,0,3,1,5
Matched!!!

Nebraska:
predict result(top 7): [0=241, 3=122, 5=96, 1=84, 11=55, 6=29, 2=25]
original result(top 7):[0=175, 3=130, 1=89, 5=84, 11=47, 19=42, 6=29]

predict top 5: 0,3,5,1,11
original top 5: 0,3,1,5,11
Matched!!!

October 2015
Iowa:
predict result (top 7):[6=306, 0=301, 3=136, 1=124, 5=100, 11=95, 19=82]
original result(top 7):[6=285, 0=263, 3=134, 1=125, 5=96, 19=93, 11=93]

predict top 5: 6,0,3,1,5
original top 5: 6,0,3,1,5
Matched!!!

Nebraska:

predict result(top 7): [0=230, 3=120, 5=75, 1=68, 11=57, 2=28, 15=22]
original result(top 7):[0=175, 3=123, 5=73, 1=69, 11=61, 19=42, 2=25]

predict top 5: 0,3,5,1,11
original top 5: 0,3,5,1,11
Matched!!!

November 2015
Iowa:
predict result(top 7): [0=347, 6=262, 1=158, 3=149, 11=74, 5=74, 19=63]
original result(top 7):[0=289, 6=264, 3=161, 1=124, 11=81, 19=80, 5=71]

predict top 5: 0,6,1,3,11
original top 5: 0,6,3,1,11
Matched!!!

Nebraska:
predict result(top 7): [0=22, 3=9, 1=3, 6=3, 19=2, 9=2, 5=1]
original result(top 7):[0=16, 3=10, 1=4, 6=2, 19=2, 15=2, 9=2]

predict top 5: 0,3,1,6,19
original top 5: 0,3,1,6,19
Matched!!!