Classifying Computer Science Papers

Tong Zhou, Yi Zhang, Jianguo Lu
School of Computer Science, University of Windsor
401 Sunset Avenue, Windsor, Ontario N9B 3P4. Canada
Email: {zhou142, zhang18f, jlu}@uwindsor.ca

Abstract

This paper addresses the problem of classifying academic papers. It is a building block in constructing an advanced scholarly search engine, such as in crawling and recommending papers in a particular area. Our goal is to identify the best classification method for scholarly data, to choose appropriate parameters, and to gauge how accurate academic papers can be classified using document content only. In addition, we also want to find out whether the neural network approach, which has been proven very successful in many other areas, can help in this particular problem.

Our experiments are conducted on 160,000 papers from arXiv data set, each is already labeled as either a computer science (CS) paper or a paper in other areas. We experimented with a variety of classification methods, including Multinomial Naive Bayes on unigram and bigram models, and Logistic Regression on distributional representation obtained from sentence2vec.

We find that computer science papers can be identified with high accuracy (F_1 close to 0.95). The best method is the bigram model using Multinomial Naive Bayes method and point-wise mutual information (PMI) as the feature selection method.

Introduction

Academic papers need to be classified for a variety of applications. When we build an academic search engine specializing in computer science (CS), we need to judge whether a document crawled from the Web and online social networks is a CS paper; When recommending papers in certain area, we need to infer whether a paper is on that topic or in the area of a specific researcher. There are numerous techniques to address these problems, but the basic building block is the classification techniques based on the text.

Although text classification has been studied extensively (Aggarwal and Zhai 2012), studies targeting academic papers are limited and inconclusive (Craven, Kumlien, and others 1999) (Kodakateri Pudhiyaveetil et al. 2009) (Lu and Getoor 2003) (Caragea et al. 2011). It is not clear how accurate we can classify academic papers, and what are the best methods. Thus, our research questions are: 1) Can we tell the difference between a CS paper and a non-CS paper? 2) What is the best method for academic paper classification? 3) What are the best parameters for each method? Each classification method has many parameters. Take Naive Bayes (McCallum, Nigam, and others 1998) method for example, there are different models (e.g.,unigram, bigram (Cavnar, Trenkle, and others 1994)), different feature selection methods (e.g., mutual information (MI), χ^2, point-wise mutual information (PMI) (Rogati and Yang 2002) (Yang and Pedersen 1997)), different pre-processing (e.g., stop words, stemming (Aggarwal and Zhai 2012)), systemic bias correction (e.g., length normalization and weight adjustment (Rennie et al. 2003)). Due to the unique characteristics of academic papers, the choosing of correct parameters need to be investigated. 4) Whether the neural network approach helps in this area? Given the recent success of deep learning in many domains, we need to check whether approaches spawn from word2vec (Mikolov et al. 2013) and sentence2vec (Le and Mikolov 2014) can improve the performance.

To find answers to these questions, we conducted a string of experiments on a variety of scholarly data, including citeSeerX (Lawrence, Giles, and Bollacker 1999) and arXiv (Warner 2005) data. This paper reports our result on the arXiv data only. This is because each paper in arXiv is labeled with an area, and the label is considered accurate since it is self-identified by its authors. Our experimental data set is obtained from the most recent arXiv collection after balancing and duplicate removing, which contains 80,000 CS papers as positive class and 80,000 non-CS papers as negative class.

The methods we tested include multinomial Naive Bayes (MNB) on unigram and bigram models, and MNB and logistic regression (Hosmer, Jovanovic, and Lemeshow 1989) on vector representations generated using sentence2vec. We chose these methods for scalability consideration. Other methods, such as the well-known SVM (Joachims 1998), are tried without success. The experiments are carried on two powerful servers with 256 GB memory and 24 core CPU.

Our result shows that CS papers can be classified with high accuracy with large training data. Most methods can achieve an F_1 value above 0.9. The best method is the bigram model using MNB. The out-of-box sentence2vec is inferior to the bigram model by almost 2 percent. Interestingly, removing stop words helps in all the methods, even in sentence2vec, while stemming has limited impact. Histor-

ically, PMI is considered inferior in text classification (Xu et al. 2007). We show that when the feature size is large, it out-performs MI and χ^2.

Related work

Classifying Academic Papers

Classification of academic papers have been studied in small scale with mixed results. (Craven, Kumlien, and others 1999) used Naive Bayes algorithm to classify biomedical articles. They gathered a corpus of 2,889 abstracts from the MEDLINE database and concluded that Naive Bayes reached higher classification precision than their baseline algorithm. When recall = 25%, the precision for baseline algorithm is 44% and the precision for Naive Bayes algorithm is 70%. However, even though they claimed the good performance for their result, 70% precision is rather low probably due to the small training data size and poor selection of the parameters.

(Kodakateri Pudhiyaveetil et al. 2009) categorized CiteSeer (an online scholarly database) papers into 268 different categories based on the ACM CCS class definition. They used k-NN algorithm to train the classifier with 268 classes while each class has 10 sampled papers. For each test paper, they output the top K predicted classes. However, they didn't use systematic evaluation scheme (e.g. cross validation scheme) to evaluate the performance of the classifier.

(Lu and Getoor 2003) conducted text classification on multiple data sets by using the logistic regression algorithm. They experimented with three kinds of data sets: Cora (4,187 papers), CiteSeer (3,600 papers) and WebKB (700 web pages). Each data set has multiple classes. They used stemming and stop words removal to pre-process the paper text, and used 3-fold cross validation and F_1 measure to evaluate the classifier. By using text only as features to train the classifier, the highest F_1 for Cora is 0.643, for CiteSeer is 0.551 and for WebKB is 0.832. Still, the problem is the small data size (less than 5,000 docs).

Also using Cora and CiteSeer data sets, (Caragea et al. 2011) discussed lowering feature dimensionality to simplify the complexity of classifiers. Under SVM and Logistic Regression, they experimented with three different schemes: Mutual Information algorithm, topic models, and feature abstraction. Both of them can reach a better classification accuracy compared with using all features, and feagture abstraction performs the best (Cora: 79.88, CiteSeer: 72.85).

Classification and Feature Selection Algorithms

Apart from academic papers, most of the previous researchers conducted more thorough text classification experiments on benchmark data sets such as Reuters-21578. (Yang and Liu 1999) thoroughly compares the performances of five different text classification algorithms: SVM, k-NN, LLSF (Linear Least Squares Fit), NNet (Neural Network) and NB (Naive Bayes). Measured by micro average F_1, they concluded the performance ranking for the five algorithms as $SVM > kNN \gg \{LLSF, NNet\} \gg NB$. Although (Yang and Liu 1999) concluded that more sophisticated algorithms such as SVM can outperform Naive Bayes, the time and space complexity of SVM are much higher than Naive Bayes. Hence, Naive Bayes is still very popular, especially in text classification where feature dimensionality is much higher. Moreover, previous researchers have proved that even though the independent assumption of Naive Bayes can be unrealistic in most of the text classification scenarios, Naive Bayes can still perform surprisingly well (McCallum, Nigam, and others 1998). Authors in (McCallum, Nigam, and others 1998) illustrated two models for Naive Bayes: Bernoulli Model and Multinomial Model, and concluded that Multinomial Model is better for text classification. Another paper (Rennie et al. 2003) described improvements on Multinomial Naive Bayes, using TF-iDF weighting and length normalization to balance the feature weights.

As for feature selection, (Rogati and Yang 2002) compared and concluded the most efficient filter feature selection algorithms for text classifiers. They concluded that χ^2-statistic (CHI) consistently outperformed other feature selection criteria for multiple classification algorithms.

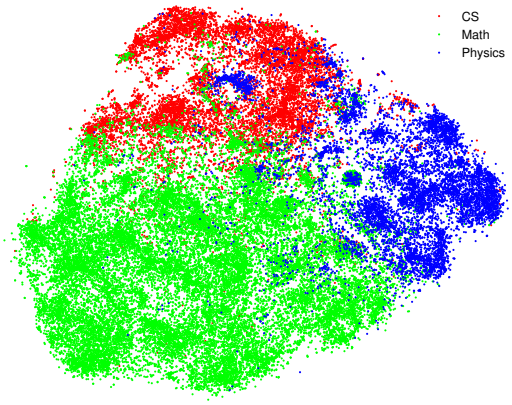

Figure 1: Three classes of documents. Vectors with 100 dimensions are trained using Sentence2vec. Then they are reduced to two dimension using t-SNE.

Data

The most recent arXiv collection contains 840,218 papers. Each paper includes title and abstract, and is labeled with a discipline (e.g. CS, Math, Physics, etc.). An overview of the data can be depicted by Fig. 1. Each point is a vector representing a paper in areas of Computer Science, Math or Physics.

We observed that duplicates exist in the arXiv data set, and removed the duplicates by comparing their URLs. After removing duplicates, there are 84,172 CS papers and 575,043 non-CS papers. Non-CS corresponds to all the other disciplines such as Math, Physics. We use random sampling to equalize the amount of CS and non-CS papers to solve the data set imbalanced problem (mostly reduce the size of non-CS class). Our final experimental data set contains sampled 80,000 CS and 80,000 non-CS papers. Most of the non-CS papers belong to Math (29,899). Before ex-

tracting feature sets for different methods, we tokenize them where each token contains only alphanumeric letters. Each token is also case folded.

Methods

We used NLTK (Bird 2006) English Stop Words List to filter stop words, and Porter stemmer (Porter 1980) to do the stemming.

In unigram and bigram models, the feature sizes are in the order of 10^6. Thus, only MNB is tested for scalability reasons. We also tried Bernoulli NB method (BNB), which is inferior to MNB. Since BNB is well-known to be inferior for long text classification (McCallum, Nigam, and others 1998), we do not report BNB method in this paper.

Sentence2vec (Le and Mikolov 2014) is a deep learning approach to learning the embeddings of sentence from the training datasets. It has two models. One is the Distributed Memory Model of Paragraph Vectors (PV-DM), which trains a sentence vector along with the word vectors to predict the missing content. In this model, paragraph vector represents the missing information from the current context and can act as a memory of the topic of the paragraph. The second model is Distributed Bag of Words version of Paragraph Vector (PV-DBOW), in which the sentence vector is trained to predict the words in a small window. Combined with negative sampling, sentence2vec can update the over 2,000 embedding per second per core. Sentence2vec have several parameters for both models, the most important parameters are window-size, negative sample size, and the demission of the vectors. After series tests, we find the combination of PV-DM model with vector dimension = 100, window size = 10, negative sample = 5 gives the best embeddings for classification. Thus, we keep this setup in the following experiment.

In MNB for text classification, feature weight needs to be adjusted (Rennie et al. 2003). TF-iDF and length normalization are taken into consideration to compute feature weights instead of simple frequency count. We tested the feature weight normalization using the following equation that is given in (Rennie et al. 2003):

$$f'_i = \log(1 + f_i) \cdot \log \frac{N}{\sum_d \delta_{id}}$$
$$f''_i = \frac{f'_i}{\sqrt{\sum_k (f'_k)^2}} \quad (1)$$

The first equation that calculates f'_i is the TF-iDF normalization. TF is the frequency of a feature in a document, and is normalized using logarithm. iDF gives more weight to features that have lower document frequency. The second equation that calculates f''_i is length normalization, the function is to eliminate the effect of length variation to the weight of features. Finally, normalized feature weight uses f''_i instead of f_i as the final weight of features that input into the classification system.

		SW	ST	SW + ST	OT
MNB	Precision	0.9021	0.8999	0.8972	0.8992
	Recall	0.9150	0.9077	0.9146	0.9049
	F_1	**0.9085**	0.9038	0.9058	0.9021
	Time	0:01:38	0:01:37	0:01:37	0:01:38
LR	Precision	0.9269	0.9228	0.9241	0.9035
	Recall	0.9318	0.9295	0.9295	0.9290
	F_1	**0.9293**	0.9261	0.9268	0.9262
	Time	0:03:49	0:03:51	0:03:47	0:03:48

Table 1: Classification results using setence2vec representation. Vector dimension is 100. Best F_1 is achieved using logistic regression. Precision, recall, and F_1 are average of 10 runs.

		SW	ST	SW + ST	OT
sentence2vec		**0.9085**	0.9038	0.9058	0.9021
unigram	Un-NL	**0.9326**	0.9293	0.9289	0.9288
	NL	**0.9354**	0.9317	0.9314	0.9312
bigram	Un-NL	**0.9460**	0.9425	0.9440	0.9424
	NL	**0.9467**	0.9449	0.9448	0.9444

Table 2: F_1 for various classification methods. Removing stop words (SW) improves the performance consistently for all classification methods. Stemming (ST) is not necessary.

Experiments

Impact of stop words and stemming

Table 1 and Table 2 show the performance on variations of text pre-processing. Four models are created: **SW** is only removing stop words, **ST** is only stemming, **SW + ST** is removing stop words + stemming, **OT** is keeping original text. Table 1 is for sentence2vec approach, and 2 includes the unigram and bigram models . We can see that removing stop words improves the F_1 value consistently across all methods, while effect of stemming is very limited. Table 1 also shows that under sentence2vec, Logistic Regression (LR) outperforms Multinomial Naive Bayes (MNB) with more than 2% F_1 measure value. Furthermore, from Table 2 we can see that under unigram and bigram, models with normalized feature weight consistently perform better than un-normalized (original frequency count as feature weight) models.

Impact of Training Data Size

Fig. 2 shows the F_1 value as a function of training data size. We can see that the performance improves with the data size in general as expected. What is interesting is bigram model increases with a faster pace, thereby it outperforms all other methods when the data size is large. s2v approach is better than bigram model when data size is small. But its improvement tapers off with the growth of data.

Panel (B) in Fig. 2 shows the impact of normalization. For both bigram and unigram models, normalization plays a minor role. It can be explained by the fact that the document size (title + abstract) are similar, and the dependency between tokens are similar across different classes.

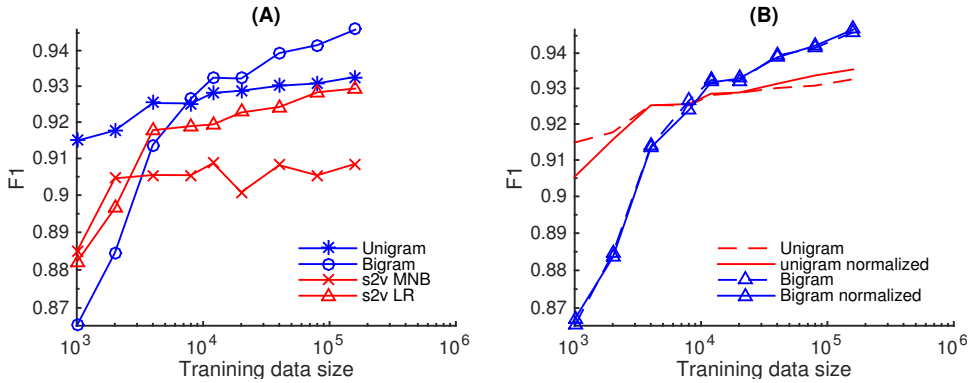

Figure 2: Impact of training data size on F_1. (A) Both unigram and bigram model out-perform sentence2vec when training data is large; (B) Normalization plays a minor role in this data.

		Size = 1,000		Size = 160,000	
		MNB	LR	MNB	LR
sentence2vec		0.8848	0.8821	0.9085	0.9293
unigram	Un-NL	0.9148	-	0.9326	-
	NL	0.9054	-	0.9354	-
bigram	Un-NL	0.8653	-	0.9460	-
	NL	0.8671	-	0.9467	-

Table 3: Classification results with the increase of training data size.

unigram			bigram		
Rank	Name	CTF	Rank	Name	CTF
1	quantum	**2856** / 24576	1	magnetic field	**28** / 2924
2	**algorithm**	**41238** / 4843	2	**state art**	**4422** / 279
3	field	**4523** / 20706	3	field theory	**46** / 2204
4	**network**	**31142** / 4014	4	two dimensional	**679** / 3150
5	**performance**	**23800** / 2081	5	**log n**	**3705** / 262
6	**algorithms**	**23178** / 2032	6	**polynomial time**	**3364** / 156
7	**based**	**47442** / 12647	7	x ray	**65** / 1781
8	spin	**248** / 10336	8	**paper propose**	**3348** / 234
9	0	**5897** / 19455	9	ground state	**10** / 1515
10	equation	**1193** / 11352	10	**o n**	**3631** / 441
11	**information**	**28191** / 4953	11	black hole	**68** / 1492
12	**networks**	**25829** / 4076	12	boundary conditions	**94** / 1483
13	**learning**	**16895** / 965	13	**real world**	**2849** / 244
14	**problem**	**40677** / 11110	14	su 2	**2** / 1223
15	magnetic	**247** / 8610	15	phase transition	**247** / 1733

Table 4: Top Selected Features by χ^2

Table 3 also lists the statistics of average F_1 measure values corresponding to Fig. 2.

Impact of feature size

We rank features based on their scores produced by PMI, MI and χ^2 respectively. In Fig. 3 we can clearly see the plotted top ranked few features are all bias in only one specific class, either CS or non-CS. X-axis represents the CTF (Class Term Frequency) in CS class, Y-axis represents the CTF in non-CS class. In Fig. 3 (A) (B), red nodes represent the top 3,000 ranked unigram features, and the blue nodes are all the remaining features. We can see that all of the red nodes are deviated from $y = x$, which means the CTF in one class are significantly larger than the other class. Accord-

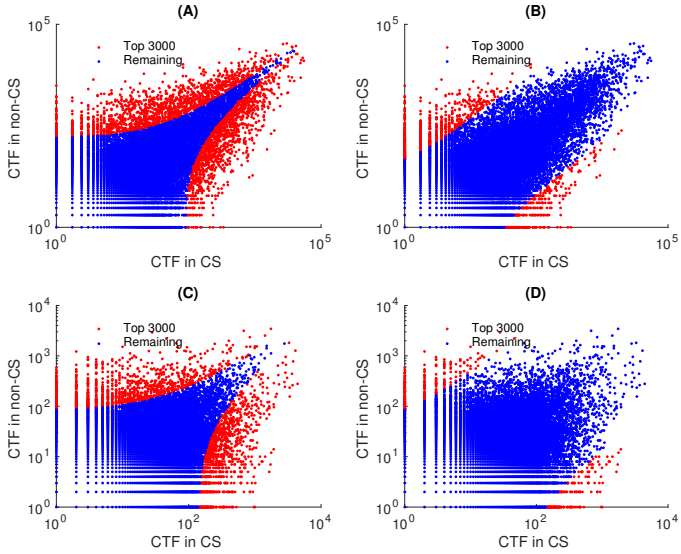

Figure 3: Top features selected by different methods. Panel (A) and (C): χ^2; (B) and (D): PMI. (A) and (B) are unigram models; (C) and (D) are bigram models.

unigram			bigram		
Rank	Name	CTF	Rank	Name	CTF
1	supersymmetric	**0** / 1828	1	yang mills	**0** / 1151
2	mev	**0** / 1586	2	gauge theories	**0** / 714
3	chiral	**1** / 3074	3	su 3	**0** / 680
4	pion	**0** / 1161	4	non perturbative	**0** / 636
5	supersymmetry	**0** / 1035	5	spin orbit	**0** / 621
6	branes	**0** / 1003	6	**sum rate**	**946** / 0
7	mesons	**0** / 948	7	quantum gravity	**0** / 554
⋮	⋮	⋮	⋮	⋮	⋮
11	**beamforming**	**1439** / 0	23	**mimo systems**	**680** / 0
23	**precoding**	**949** / 0	24	**outage probability**	**679** / 0
28	**multicast**	**891** / 0	28	**access control**	**665** / 0
31	**cnn**	**842** / 0	29	**logic programs**	**665** / 0
34	**p2p**	**764** / 0	35	**ieee 802**	**590** / 0
37	**qos**	**1431** / 1	36	**interference alignment**	**580** / 0

Table 5: Top Selected Features by PMI

ingly, Fig. 3 (C) (D) shows the distribution of the bigram top 3,000 ranked features and remaining features. Table 4 shows the top 15 selected features by χ^2. Bold features in Table 4 have higher CTF in CS class. We can see χ^2 tend to select big/popular words have high total occurrences but still have significant different occurrences in the two classes, and the ratio of CS features and non-CS features tend to be equal. However, PMI tend to select very exclusive words, we can see from Table 5 that top ranked features have almost 0 CTF in the other class. And top CS features ranked lower than non-CS features, since non-CS papers have more highly occurred exclusive words.

Fig. 4 demonstrates the impact of feature size on classification performance when using three different kinds of feature selection algorithms: PMI, MI and χ^2, and their combinations with bigram and unigram models. Two feature weighting schemes, normalized feature weight and simple frequency count, are tested. We select top k ranked features for classifier and change k (X-axis) to draw the performance curves. We can see that classification performance (F_1) increases with feature size for most cases. PMI increases faster than χ^2 and MI. Fig. 4 (A) (B) shows the unigram classification results. We can see that when $k > 30000$, the classification results for PMI under frequency count model are better than using all features; more features can let PMI performs better than χ^2 and MI for all the four models. Moreover, normalized feature weight improves the performances for both PMI, MI and χ^2. Fig. 4 (C) (D) shows the bigram classification results. both PMI, MI and χ^2 reaches the best classification results when using all features. When gradually decreasing features, firstly χ^2 and MI drops lower than PMI, but then PMI drops below χ^2 and MI. In addition, for bigram features, an interesting phenomenon is that from feature size 10^6 to all features (5.2 millions), F_1 first drop then increase greatly to the highest value with all features. The final increasing trend is caused by the increasing of TN (True Negative) value that pull the classification precision higher.

Discussions and Conclusions

When classifying research papers in computer science, we find that most classification methods can reach an F_1 value as high as 0.9. The best method is MNB on bigram language model, which obtained an F_1 value close to 0.95. The out-of-box neural network approach does not perform as well as bigram model. On sentence2vec representation, neither logistic regression nor Naive Bayes can compete with bigram model. Other classification methods, including SVM, are also tested on smaller datasets because of the scalability issue of these algorithms. SVM performs similar with Logistic Regression on sentence2vec representation.

For multinomial Naive Bayes text classification, it was long believed that PMI is not a good candidate for feature selection. On the contrary to this believe, we show that PMI is better than χ^2 and MI. This is probably because of the size of our training data is bigger– in Fig. 4 we can see that PMI is inferior until the feature size exceeds 10^4.

This paper also shows that stop word removing improves the performance for all the methods, including bag of words model, bigram model, and various classification methods

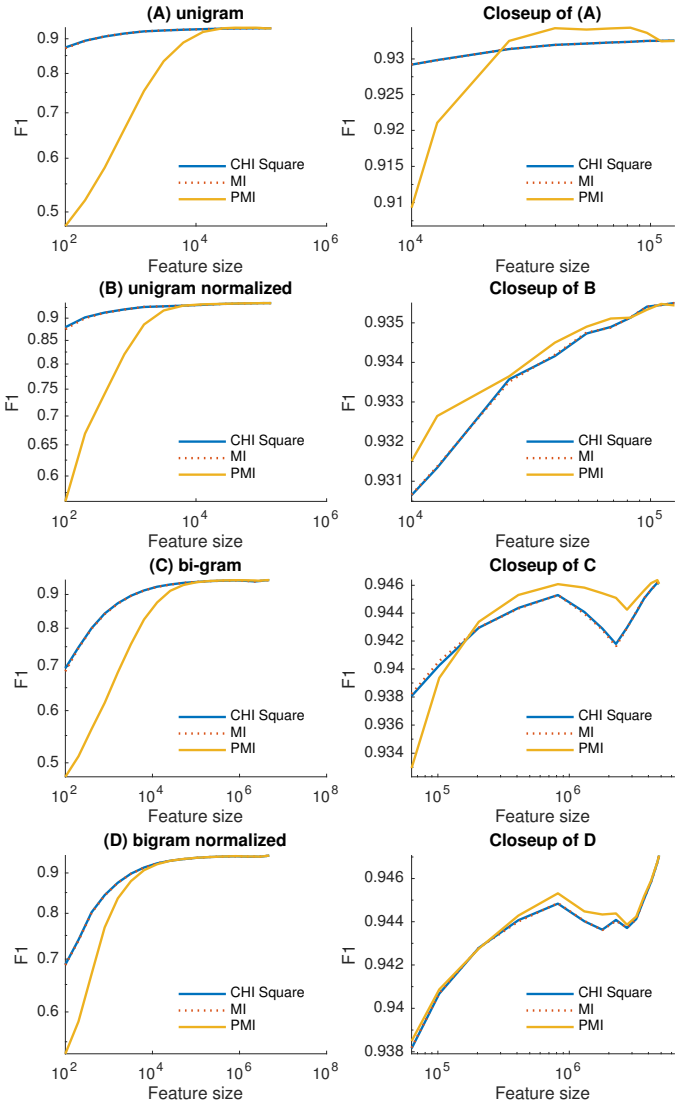

Figure 4: Impact of feature size for unigram and bigram models, with combination of text normalization. (A) Unigram; (B) unigram normalized; (C) bigram; (D) bigram normalized.

on distributional vector representation of documents. On the other hand, stemming has limited impact on the performance.

It is surprising to see that academic papers can be classified with high accuracy based on content only. We also tried to classify papers in narrow areas, such as papers in conferences VLDB, SIGMOD, and ICSE, each class trained on two thousand of papers. We also observed high accuracy in these experiments. Among VLDB and ICSE, the F_1 is above 0.98 because these two conferences focus on very different topics, one in database, the other in software engineering. What is surprising is that among VLDB and SIGMOD, which are both database conferences, the F_1 value is also above 0.88. We believe that if we augment the data with the citation and co-author networks, the accuracy could be even higher.

With such high accuracy, we can envision numerous applications in the pipeline. We are building an academic search engine in the area of computer science. When crawling the data from the Web and online social networks, we can judge whether a document is a computer science paper; when conducting author disambiguation, we can determine whether a paper is written by a certain person or a group of researchers or a community of academics; when recommending papers, we can classify the paper according to a researcher's profile.

Acknowledgement

The research is supported by NSERC Discovery grant (RGPIN-2014-04463).

References

Aggarwal, C. C., and Zhai, C. 2012. A survey of text classification algorithms. In *Mining text data*. Springer. 163–222.

Bird, S. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, 69–72. Association for Computational Linguistics.

Caragea, C.; Silvescu, A.; Kataria, S.; Caragea, D.; and Mitra, P. 2011. Classifying scientific publications using abstract features. American Association for Artificial Intelligence.

Cavnar, W. B.; Trenkle, J. M.; et al. 1994. N-gram-based text categorization. *Ann Arbor MI* 48113(2):161–175.

Craven, M.; Kumlien, J.; et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, 77–86.

Hosmer, D. W.; Jovanovic, B.; and Lemeshow, S. 1989. Best subsets logistic regression. *Biometrics* 1265–1270.

Joachims, T. 1998. *Text categorization with support vector machines: Learning with many relevant features*. Springer.

Kodakateri Pudhiyaveetil, A.; Gauch, S.; Luong, H.; and Eno, J. 2009. Conceptual recommender system for citeseerx. In *Proceedings of the third ACM conference on Recommender systems*, 241–244. ACM.

Lawrence, S.; Giles, L. C.; and Bollacker, K. 1999. Digital libraries and autonomous citation indexing. *Computer* 32(6):67–71.

Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Lu, Q., and Getoor, L. 2003. Link-based classification. In *ICML*, volume 3, 496–503.

McCallum, A.; Nigam, K.; et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, 41–48. Citeseer.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program* 14(3):130–137.

Rennie, J. D.; Shih, L.; Teevan, J.; Karger, D. R.; et al. 2003. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, volume 3, 616–623. Washington DC).

Rogati, M., and Yang, Y. 2002. High-performing feature selection for text classification. In *Proceedings of the eleventh international conference on Information and knowledge management*, 659–661. ACM.

Warner, S. 2005. The arxiv: Fourteen years of open access scientific communication. In *Free Culture and the Digital Library Symposium Proceedings 2005*, 56.

Xu, Y.; Jones, G. J.; Li, J.; Wang, B.; and Sun, C. 2007. A study on mutual information-based feature selection for text categorization. *Journal of Computational Information Systems* 3(3):1007–1012.

Yang, Y., and Liu, X. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 42–49. ACM.

Yang, Y., and Pedersen, J. O. 1997. A comparative study on feature selection in text categorization. In *ICML*, volume 97, 412–420.