IJCAI 2016

Proceedings of the Workshop on Scholarly Big Data: AI Perspectives, Challenges, and Ideas

July 9, 2016
New York City, USA

Preface

The IJCAI 2016 Workshop on Scholarly Big Data: AI Perspectives, Challenges, and Ideas is to be held on July 9, 2016 in New York. The workshop's objective is to bring together researchers addressing a wide-range of questions pertaining to mining, managing and searching Scholarly Big Data and using the Social Web lenses to discover new patterns and introduce new metrics.

The workshop program includes two invited talks by researchers who are experts in the fields of data mining, information retrieval, and natural language processing: Prof. C. Lee Giles from the Pennsylvania State University and Dr. Iris Shen from Microsoft Research, Redmond. After a rigorous review process, four long papers were selected for inclusion into the workshop proceedings by the Program Committee.

We hope that these papers summarize novel findings related to scholarly big data and inspire further research interest on this exciting topic. We thank the authors, invited speakers, program committee members, and participants for sharing their
research ideas and valuable time to be part of IJCAI 2016 SBD!

July 1, 2016
Singapore

Cornelia Caragea
Madian Khabsa
Sujatha Das Gollapalli
C. Lee Giles
Alex D. Wade

Program Committee

Hamed Alhoori	Northern Illinois University
Cornelia Caragea	University of North Texas
Doina Caragea	Kansas State University
Sujatha Das Gollapalli	I2R, A*STAR
C. Lee Giles	The Pennsylvania State University
Kazi Hasan	University of Texas at Dallas
Min-Yen Kan	National University of Singapore
Madian Khabsa	Microsoft Research
Rada Milhalcea	University of Michigan
Ani Nenkova	University of Pennsylvania
Yang Peng	Institute for Infocomm Research, A*STAR
Kazunari Sugiyama	National University of Singapore
Niket Tandon	Max Planck Institute for Informatics
Suppawong Tuarob	Mahidol University
Alex Wade	Microsoft Research
Xiaojun Wan	Peking University
Zhaohui Wu	The Pennsylvania State University
Feng Xia	Dalian University of Technology
Fang Yuan	Institute for Infocomm Research, A*STAR

Additional Reviewers

Bekele, Teshome Megersa
Wang, Wei

Table of Contents

(Invited Talk) Introduction to Scholarly Big Data

C. Lee Giles
College of Information Sciences and Technology
The Pennsylvania State University
giles@ist.psu.edu

About the Speaker:
C. Lee Giles is the David Reese Professor of Information Sciences and Technology at the Pennsylvania State University with appointments in the departments of Computer Science and Engineering, and Supply Chain and Information Systems. He is also the Director of the Intelligent Systems Research Laboratory. He was a co-creator of the popular search engine CiteSeer (now CiteSeerx) and related scholarly search engines. He directs the CiteSeer^x project and co-directs the ChemxSeer project at Penn State. Lee's research interests are in intelligent cyber-infrastructure and big data, web tools, specialty search engines, information retrieval, digital libraries, web services, knowledge and information extraction, data mining, entity disambiguation, and social networks. He has published over 300 papers in these areas. He is a fellow of the ACM, IEEE, and INNS. Lee serves on many related conference program committees and has helped organize many related meetings and workshops. He has given many invited and keynote talks and seminars.

(Invited Talk) Microsoft Academic Service and Applications: Challenges and Opportunities

Zhihong (Iris) Shen
Microsoft Research, Redmond, WA
hihosh@microsoft.com

Abstract

In this talk, we will introduce the new release of a Web scale entity graph, which serves as the backbone of Microsoft Academic Service. The architecture of the data pipeline which produces Microsoft Academic Graph will be presented. Challenges and opportunities on various research topics as well as engineering efforts are exploited. In addition, Microsoft Research has opened up this graph dataset to the research community with new APIs to support further research, experimentation, and development. This talk will highlight how the research community can take advantage of these data and APIs to fuel new research opportunities.

About the Speaker:

Zhihong (Iris) Shen is a Senior Data Scientist at Microsoft Research, Redmond, WA. She obtained her Ph.D. in Operations Research at the University of Southern California. She is involved in the new generation of Microsoft Academic Service since 2014 and has been working on various research and engineering problems in the project, such as heterogeneous graph generation, entity recognition/linking, and entity recommendation etc.

Random Forest DBSCAN Clustering for USPTO Inventor Name Disambiguation and Conflation

Kunho Kim[‡], Madian Khabsa[*], C. Lee Giles[†‡]

[‡]Computer Science and Engineering
[†]Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802, USA

[*]Microsoft Research
One Microsoft Way
Redmond, WA 98005, USA

kunho@cse.psu.edu, madian.khabsa@microsoft.com, giles@ist.psu.edu

Abstract

Name disambiguation and the subsequent name conflation are essential for the correct processing of person name queries in a digital library or other database. It distinguishes each unique person from all other records in the database. We study inventor name disambiguation for a patent database using methods and features from earlier work on author name disambiguation and propose a feature set appropriate for a patent database. A random forest was selected for the pairwise linking classifier since they outperform Naive Bayes, Logistic Regression, Support Vector Machines (SVM), Conditional Inference Tree, and Decision Trees. Blocking size, very important for scaling, was selected based on experiments that determined feature importance and accuracy. The DBSCAN algorithm is used for clustering records, using a distance function derived from random forest classifier. For additional scalability clustering was parallelized. Tests on the USPTO patent database show that our method successfully disambiguated 12 million inventor mentions within 6.5 hours. Evaluation on datasets from USPTO PatentsView inventor name disambiguation competition shows our algorithm outperforms all algorithms in the competition.

Introduction

One of the most frequent queries for digital library search system is a person name. An example is to find all relevant records of a particular person. For a patent database, users may want to find the list of patents of a certain inventor. This query can be problematic if there is no unique identifier for each person. In that case, a method must be used to distinguish between person records in the database. This is often referred to as the personal name disambiguation problem.

There are several factors that make this problem hard. Firstly, there can be several different formats for displaying one person's name. For example, one record has the full name "John Doe", while another contains only the iniital of first name, "J. Doe". More importantly, there are some common names that many people share. We can see this problem often with Asian names. Statistics from Wikipedia[1] show that 84.8% of the population have one of the top 100 popular surnames in China, while only 16.4% of common names in United States. For some records the first and last name is reversed, especially for certain groups that put the last name first for their full name. Lastly, typographical errors and foreign characters can also challenge disambiguation. In addition, because of the large number of records, usually millions, manual disambiguation of all records is not feasible (which is even then not perfect) and automated methods have to be used. An automatic name disambiguation algorithm typically consists of two parts. The first is a pairwise linkage classifier that determines whether each pair of records are from the same person or not (Winkler 2014). The second is a clustering algorithm, grouping records for each unique person using the classifier.

Here, we propose to use an author name disambiguation algorithm for the patent inventor database. Our algorithm follows the typical steps of author name disambiguation, but with a newly proposed set of features from patent metadata. Having experimented with several different classifers, we use a random forest classifier to train for pairwise linkage classification and use DBSCAN for clustering for disambiguation. We use the publicly available USPTO database for testing. Recently there was an inventor name disambiguation competition for this database. Raw data is publicly available via the competition's web page[2]. This raw data contains all published US patent grants from 1976 to 2014. Although we didn't participate in the competition, we used the same training and test datasets used in the competition for evaluation. The competition's evaluation results show our algorithm to be superior to other suggested algorithms in the competition. A detailed explanation of dataset and results can be found in results section.

Related Work

Several approaches have been proposed for pairwise linkage classification using different machine learning algorithms. Han et al. (2004) proposed two approaches using a Hybrid

A shorter version of this paper was published in JCDL (Kim, Khabsa, and Giles 2016)

[1]List of common Chinese surnames, in Wikipedia. https://en.wikipedia.org/wiki/List_of_common_Chinese_surname

[2]http://www.dev.patentsview.org/workshop

Naive Bayes and support vector machine(SVM) classifier. Huang et al. (2006) used an online active SVM(LASVM) to boost the speed of SVM classifier. Song et al. (2007) used probabilistic latent semantic analysis(pLSA) and Latent Dirichlet allocation(LDA) to disambiguate names based on publication content. Treeratpituk and Giles (2009) first introduced the random forest (RF) for disambiguation and showed the random forest classifier at the time to have the best accuracy compared to other machine learning based classifiers. Godoi et al. (2013) used an iterative approach to update ambiguous linkage with user feedback. Fan et al. (2011) used graph based framework for name disambiguation, and Hermansson et al. (2013) used graph kernels to calculate similarity based on local neighborhood structure. Instead of using machine learning algorithms, Santana et al. (2014) used domain specific heuristics for classification. Recently, Ventura et al. (2015) applied a random forest classifier with agglomerative clustering for inventor name disambiguation for USPTO database.

For clustering algorithms for disambiguation, Mann and Yarowsky (2003) used a simple agglomerative clustering which still had a transitivity problem. The transitivity problem occurs when there are three records *a*, *b*, *c* and while *a* matches to *b*, *b* matches to *c*, *a* does not match with *c*. Han et al. (2005) used K-spectral clustering which had scaling issues and the K (number of clusters) was heuristically determined. To overcome those problems, Huang et al. (2006) proposed a density-based clustering(DBSCAN) algorithm. Another is a graphical approach using conditional random fields for clustering, using Markov Chain Monte Carlo(MCMC) methods (Wick, Singh, and McCallum 2012). Recently Khabsa et al. (2015) proposed a constraint-based clustering algorithm based on DBSCAN and extended it to handle online clustering.

Disambiguation Process

Patent records have consistent metadata to that of scholarly publications. There exists title, personal information of inventors, such as name, affiliation, etc. Ventura et al. (2015) applied author name disambiguation algorithms to patent records, showing very promising results. Our algorithm follows the same general steps of author name disambiguation.

First we train a pairwise classifier that determines whether each pair of inventor records is same person or not. Second, we apply blocking to the entire records for scaling. Finally, we cluster inventor records from each block separately using the classifier learned from the previous step.

Training Pairwise Classifier

Pairwise classifier is needed to distinguish whether each pair of inventor records is the same person or not. In this section we show what features are used and how we sample the training data. We compare several machine learning classifiers to find the best one for inventor name disambiguation.

Selecting Features We start with the feature set used in Ventura et al. (2015), and test additional features that are used in author disambiguation for scholarly databases. We only kept features that had a meaningful decrease in Gini importance if they were removed. Table 1 shows all features used for the random forest classifier. A detailed explanation of each term is as follows:

Category	Subcategory	Features
Inventor	First name	Exact, Jaro-Winkler, Soundex
	Middle name	Exact, Jaro-Winkler, Soundex
	Last name	Exact, Jaro-Winkler, Soundex, IDF
	Suffix	Exact
	Order	Order comparision
Affiliation	City	Exact, Jaro-Winkler, Soundex
	State	Exact
	Country	Exact
Co-author	Last name	# of name shared, IDF, Jaccard
Assignee	Last name	Exact, Jaro-Winkler, Soundex
Group	Group	Exact
	Subgroup	Exact
Title	Title	# of term shared

Table 1: Features used for the random forest

- Exact: Exact string match, 3 if name matches and both full names, 2 if initial matches and not both full names, 1 if initial not matches and not both full names, 0 if name not matches and both full names.
- Jaro-Winkler: Jaro-Winkler distance (Winkler 1990) of two strings. Jaro-Winkler distance is a variant of Jaro distance. Jaro disatnace d_j of two string s_1 and s_2 is calculated as

$$d_j = \begin{cases} 0 & \text{if } n_m = 0 \\ \frac{1}{3}\left(\frac{n_m}{|s_1|} + \frac{n_m}{|s_2|} + \frac{n_m - \frac{1}{2}n_t}{n_m}\right) & \text{otherwise} \end{cases}$$

  where n_m is number of matching characters, and n_t is number of transpositions. Each character is considered as a match only if they are within distance of a half length of a longer string -1. Jaro-Winkler distance d_{jw} of two strings are calculated using this Jaro distance d_j,

$$d_{jw} = d_j + l_{\text{prefix}} p(1 - d_j)$$

  where l_{prefix} is length of common prefix between two strings (up to 4 characters). p is a scaling factor, we use 0.1.
- Soundex: Convert each string with Soundex algorithm (Knuth 1973) and then do an exact string match giving credit for phonetically similar strings. The basic idea of soundex algorithm is to cluster phonetically similar consonants and convert them with the group number. {b, f, p, v}, {c, g, j, k, q, s, x, z}, {d,t}, {l}, {m,n}, {r} are the 6 groups.
- IDF: Inverse document frequency(calculated by # of records total/# of records with name) of the name, to give more weight to a unique name.
- Order comparison: 2 if both records are first author, 1 if both records are last author, 0 otherwise.
- # of name shared: number of same name shared without considering order.
- # of term shared: number of shared terms appear in both titles, excluding common stop words.

Method	Precision	Recall	F1
Naive Bayes	0.9246	0.9527	0.9384
Logistic Regression	0.9481	0.9877	0.9470
SVM	0.9613	**0.9958**	0.9782
Decision Tree	0.9781	0.9798	0.9789
Conditional Inference Tree	0.9821	0.9879	0.9850
Random Forest	**0.9839**	0.9946	**0.9892**

Table 2: Comparison of different classification methods

Selecting Samples for Training Classifier The existing labeled data from the USPTO database has two challenges in that we cannot directly use all possible pairs as a training set for a classifier. First, the majority of the labeled clusters have only a single record. In the Mixture dataset, these are 3,491 clusters out of 4,956 clusters(70.44%) and in the Common characteristics dataset 26,648 clusters out of 30,745 clusters(86.67%) that have only a single record. Those clusters are not useful as training data because we can only get negative pairs (two records not from the same person) from them. To train a good classifier, we need data that can give both positive and negative examples. As such we removed all those clusters, and used clusters that only have more than 1 record.

Second, there were insufficient informative negative samples from the labeled datasets. Since we need to use blocking for scaling, we want to use only pairs that consist of records from same block, since pairs from different blocks are not going to be examined in the clustering process. But there were few different clusters within each block in the labeled datasets. Since there were fewer negative samples than positive samples and to avoid overfitting, we take samples from a bigger block than the actual blocking size, using *first 3 characters of last name+first name initial* while actual blocking is done with *full last name+first name initial*.

Classifier Selection We experimented with several supervised classifiers using the proposed feature set. We tested with the mixture of two training datasets - Mixture and Common characteristics. A detailed explanation of the datasets are in the result section. Table 2 shows the results with 4-fold cross validation. Tree-based classifiers have a higher accuracy compared to non-tree classifiers such as SVM and logistic regression. Among them, the Random Forest (RF) classifier gives the best accuracy in terms of F1 score. RF is an ensemble classifier that aggregates the votes from decision trees for classification (Breiman 2001). Previous work (Treeratpituk and Giles 2009) showed RF is effective for pairwise linkage classification in scholarly database. This experiments showed that for patent database RF also has the best accuracy. We trained our RF classifier with 100 trees with 5 features tried in each split. We estimated the out-of-bag (OOB) error of the RF to measure the classification quality. OOB error is known to be an unbiased estimation of test set classification error (Breiman 2001). The error rates for Common characteristics and Mixture dataset were 0.05% and 0.07% respectively.

Rank	Feature
1	Last name (Jaro-Winkler)
2	First name (Jaro-Winkler)
3	Last name (Exact)
4	Last name (Soundex)
5	First name (Soundex)
6	Affiliation (Jaro-Winkler)
7	First name (Exact)
8	State (Exact)
9	Middle name (Soundex)
10	Middle name (Exact)

Table 3: Top 10 important features of the random forest with respect to the Gini decrease

Blocking

The USPTO patent database consists of 12 millions of inventor mentions. Due to the limitation of physical memory, we cannot efficiently perform the clustering for the whole database. Blocking is done in preprocessing in order to solve this problem . Records are split into several blocks, based on the blocking function. The function should be carefully selected so that records from the same person are in the same block with high probability (Bilenko, Kamath, and Mooney 2006). Then we perform clustering for each block.

Table 3 shows the top 10 important features of the RF according to the average Gini decrease. The table shows the most important features are from first name and last name. All features from them are in top 10 except for the last name IDF. Thus, for best performance we made a blocking function with a combination of the first name and last name. Figure 1 and Table 4 shows the accuracy and computation time with respect to different block sizes. While precision is steady with different block sizes, recall gets lower as the block size becomes smaller, as does F1. This is because this blocking function splits the potential matches into different blocks. While the accuracy is getting lower, the computation time is reduced due to smaller block sizes. We use *full last name+initial of first name*, which was the blocking function that gives highest accuracy and that each block can be loaded fully into memory.

Clustering Using DBSCAN

We use a density-based clustering algorithm, DBSCAN (Ester et al. 1996) to cluster inventor records. DBSCAN is widely used for disambiguation, because it does not require a prior the number of clusters, and it resolves the transitivity problem (Huang, Ertekin, and Giles 2006).

Using DBSCAN to cluster inventor records, we need to define a distance function for each pair of inventor records. The RF classifier predicts whether each pair of records are from the same person or not with a binary value(0 or 1) output. From the RF, we can get the number of negative/positive votes in its trees. We use the fraction of negative(0) votes of the trees in random forest as the distance function (Treeratpituk and Giles 2009). The final resulting clusters from DBSCAN algorithm are the result of the disambiguation.

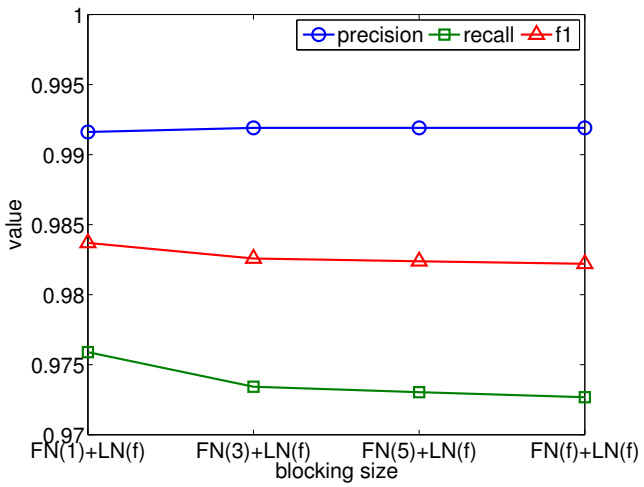


Figure 1: Evaluation of different blocking size. FN denotes the first name and LN denotes the last name. The number (n) denotes first n characters used for blocking. If n is f, full name was used.

Block	FN(1)+LN(f)	FN(3)+LN(f)	FN(5)+LN(f)	FN(f)+LN(f)
Time	6h 30min	5h 49min	5h 27min	5h 17min

Table 4: Computation time comparision for different block size

Parallelization

We use parallelization of GNU Parallel (Tange and others 2011) to utilize all cores available for clustering (Khabsa, Treeratpituk, and Giles 2014). Our work consumes memory proportional to the total number of records in the block. Due to the limitation of our physical memory, we cannot completely utilize all cores at a time if block size is too large. As such, we grouped all blocks with respect to total number of records.

The machine we use for the experiment has about 40GB memory available, and 12 cores(runs up to 24 threads simultaneously) at best. The first group consists of blocks that have less than 500 records, and we run 24 threads maximum simultaneously. The second group consists of blocks that have between 500 and 5,000 records and we run 12 threads maximum. The last group consists of blocks that have more than 5,000 records and we run 6 threads maximum.

Results

We tested our algorithm on the USPTO patent database. We used the same evaluation datasets of USPTO Patentsview inventor named disambiguation competition to compare the results. The test dataset includes ALS, ALS common, IS, E&S, and Phase2. The ALS and ALS common datasets are from Azoulay et al. (2007), which consists of inventors from the Association of Medical Colleges(AAMC) Faculty Roster. ALS common is a subset of the ALS dataset with common popular names. The IS dataset is from Trajtenberg and Shiff (2008), containing Israeli inventors in USPTO database. E&S dataset is from Ge et al. (2016) and consists of patents from engineers and scientists. Phase2 is a random mixture of previous datasets. The training dataset includes the Mixture and Common characteristics datasets. Mixture dataset is random mixture of IS and E&S dataset, and Common characteristics dataset is a subsample of E&S dataset which was subsampled according to the match characteristics of the USPTO database, in terms of the mean number of inventors per patent and percentage of missing assignees.

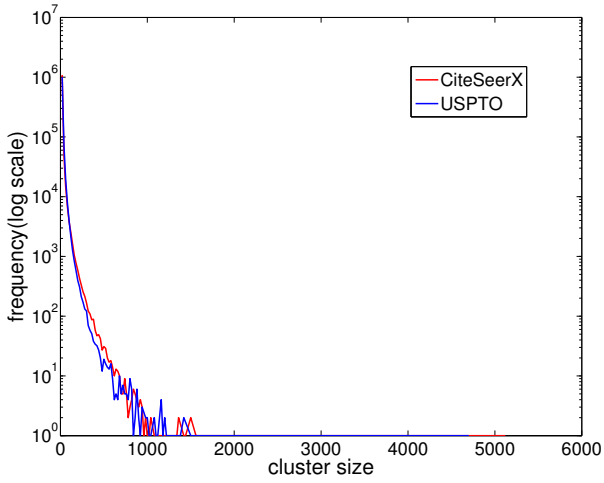


Figure 2: Frequency of each cluster size for the CiteSeerX and USPTO database

The result of our disambiguated USPTO database shows a similar tendency to previous disambiguation studies of scholarly databases. Figure 2 shows the cluster frequency of each cluster size of CiteSeerX[3] and USPTO database after disambiguation. For both databases, small clusters have high frequency and big clusters are rare with a long tail. A total of 1.11 million clusters are produced by inventor name disambiguation and the average number of patent mentions per individual inventor is 4.93. For CiteSeerX database, the average number is 6.07.

For further evaluation, we measured pairwise precision, recall, and F1 score with definitions:

$$\text{Pairwise Precision} = \frac{\text{\# of correctly matched pairs}}{\text{\# of all matched pairs by algorithm}}$$

$$\text{Pairwise Recall} = \frac{\text{\# of correctly matched pairs}}{\text{\# of pairs in manually labeled dataset}}$$

$$\text{Pairwise F1 Score} = 2 \cdot \frac{\text{Pairwise Precision} \cdot \text{Pairwise Recall}}{\text{Pairwise Precision} + \text{Pairwise Recall}}$$

Table 5 shows the results for each training and test dataset. Results were slightly better with the Common characteristics dataset, as expected from OOB error of RF. This is because common characteristics dataset has more samples and

[3] http://citeseerx.ist.psu.edu

Test Set	Training Set	Precision	Recall	F1 Score
ALS	Mixture	0.9963	0.9790	0.9786
	Common	0.9960	0.9848	0.9904
ALS common	Mixture	0.9841	0.9796	0.9818
	Common	0.9820	0.9916	0.9868
IS	Mixture	0.9989	0.9813	0.9900
	Common	0.9989	0.9813	0.9900
E&S	Mixture	0.9992	0.9805	0.9898
	Common	0.9995	0.9810	0.9902
Phase2	Mixture	0.9912	0.9760	0.9836
	Common	0.9916	0.9759	0.9837

Table 5: Disambiguation evaluation

Test Set	F1(Ours)	F1(Winner)
ALS	0.9904	0.9879
ALS common	0.9868	0.9815
IS	0.9900	0.9783
E&S	0.9902	0.9835
Phase2	0.9837	0.9826
Average($\pm$stddev.)	0.9882$\pm$0.0029	0.9827$\pm$0.0035

Table 6: Comparison with the competition winner

is subsampled according to match the characteristics of the whole USPTO database. We can also see that the recall is relatively lower compare to the precision. Blocking affects the recall, as it can remove some potential matches. Since we have a trade-off between efficiency and recall in our algorithm, blocking needs to be further improved for higher recall. Table 6 shows F1 score comparison between our work and the best result from the competition for each test dataset. The winner of the competition used a pre-defined distance metric and Markov Chain Monte Carlo(MCMC) based clustering method inspired from (Wick, Singh, and McCallum 2012). Note that our algorithm has the best performance on all datasets. The P value with one-tailed Wilcoxon test is 0.03125, which indicates that the improvement of our algorithm is statistically significant at the 0.05 level. We can see from the results that the DBSCAN algorithm with the RF classifier used in scholarly disambiguation is also effective for inventor name disambiguation in a patent database.

Our disambiguation is much faster with parallelization. We used Intel Xeon X5660@2.80GHz machine with 12 cores and 40GB memory available in an idle state, configured with RHEL 6. The disambiguation process takes about 6.5 hours to finish for both training sets. Currently we cannot fully utilize all the CPUs for certain blocks that contain large number of records, because of memory limitations. Better way of blocking such as (Bilenko, Kamath, and Mooney 2006) is needed for efficient memory usage, for fast performance and scalability. This remains as a future work.

Conclusions

We present a machine learning based algorithm for inventor name disambiguation for patent database. Motivated by the feature set of author name disambiguation for scholarly databases, we devised a proposed feature set that showed a significant low OOB error rate, 0.05% at minimum. Based on experiments with several machine learning classifiers, we use random forest classifier to determine whether each pair of inventor records are from a the same inventor or not. Disambiguation is done by using DBSCAN clustering algorithm. We define distance function of each pair of inventor records as the ratio of votes in random forest classifier. In addition to make the algorithm scalable, we use blocking and parallelization, scheduling threads based on the size of blocks. Evaluation results with the dataset from USPTO PatentsView inventor name disambiguation competition shows our algorithm outperforms all algorithms submitted to the competition in comparable running time.

Currently our algorithm is memory bounded since a great deal of memory is used to store inventor information needed for calculating features. This becomes a bottleneck when we parallelize the algorithm, since some of the block is huge due to the popularity of certain names. In future work, one could explore a better method for blocking for efficient memory usage. It would be interesting to see if other methods using graph or link data could be incorporated for better performance.

Acknowledgments

We gratefully acknowledge Evgeny Klochikhin and Ahmad Emad for assistance in the evaluation of the dataset used in USPTO PatentsView inventor name disambiguation competition and partial support from the National Science Foundation.

References

Azoulay, P.; Michigan, R.; and Sampat, B. N. 2007. The anatomy of medical school patenting. *New England Journal of Medicine* 357(20):2049–2056.

Bilenko, M.; Kamath, B.; and Mooney, R. J. 2006. Adaptive blocking: Learning to scale up record linkage. In *Proceedings of the 6th IEEE International Conference on Data Mining(ICDM'06)*, 87–96.

Breiman, L. 2001. Random forests. *Machine learning* 45(1):5–32.

Ester, M.; Kriegel, H.-P.; Sander, J.; and Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'96)*, volume 96, 226–231.

Fan, X.; Wang, J.; Pu, X.; Zhou, L.; and Lv, B. 2011. On graph-based name disambiguation. *Journal of Data and Information Quality (JDIQ)* 2(2):10.

Ge, C.; Huang, K.-W.; and Png, I. 2016. Engineer/scientist careers: Patents, online profiles, and misclassification bias. *Strategic Management Journal* 37(1):232–253.

Godoi, T. A.; Torres, R. d. S.; Carvalho, A. M.; Gonçalves, Marcos A .and Ferreira, A. A.; Fan, W.; and Fox, E. A. 2013. A relevance feedback approach for the author name disambiguation problem. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries(JCDL'13)*, 209–218.

Han, H.; Giles, C. L.; Zha, H.; Li, C.; and Tsioutsiouliklis, K. 2004. Two supervised learning approaches for

name disambiguation in author citations. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries(JCDL'04)*, 296–305.

Han, H.; Zha, H.; and Giles, C. L. 2005. Name disambiguation in author citations using a k-way spectral clustering method. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries(JCDL'05)*, 334–343.

Hermansson, L.; Kerola, T.; Johansson, F.; Jethava, V.; and Dubhashi, D. 2013. Entity disambiguation in anonymized graphs using graph kernels. In *Proceedings of the 22nd ACM International Conference on information & knowledge management(CIKM'13)*, 1037–1046.

Huang, J.; Ertekin, S.; and Giles, C. L. 2006. Efficient name disambiguation for large-scale databases. In *Proceedings of the 10th European Conference on Principle and Practice of Knowledge Discovery in Databases(PKDD'06)*, 536–544.

Khabsa, M.; Treeratpituk, P.; and Giles, C. L. 2014. Large scale author name disambiguation in digital libraries. In *IEEE International Conference on Big Data*, 41–42.

Khabsa, M.; Treeratpituk, P.; and Giles, C. L. 2015. Online person name disambiguation with constraints. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries(JCDL'15)*, 37–46.

Kim, K.; Khabsa, M.; and Giles, C. L. 2016. Inventor name disambiguation for a patent database using a random forest and dbscan. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries(JCDL'16)*.

Knuth, D. E. 1973. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley.

Mann, G. S., and Yarowsky, D. 2003. Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, volume 4, 33–40. Association for Computational Linguistics.

Santana, A. F.; Gonçalves, M. A.; Laender, A. H.; and Ferreira, A. 2014. Combining domain-specific heuristics for author name disambiguation. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries(JCDL'14)*, 173–182.

Song, Y.; Huang, J.; Councill, I. G.; Li, J.; and Giles, C. L. 2007. Efficient topic-based unsupervised name disambiguation. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries(JCDL'07)*, 342–351.

Tange, O., et al. 2011. Gnu parallel-the command-line power tool. *The USENIX Magazine* 36(1):42–47.

Trajtenberg, M., and Shiff, G. 2008. *Identification and mobility of Israeli patenting inventors*. Pinhas Sapir Center for Development.

Treeratpituk, P., and Giles, C. L. 2009. Disambiguating authors in academic publications using random forests. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries(JCDL'09)*, 39–48.

Ventura, S. L.; Nugent, R.; and Fuchs, E. R. 2015. Seeing the non-stars:(some) sources of bias in past disambiguation approaches and a new public tool leveraging labeled records. *Research Policy*.

Wick, M.; Singh, S.; and McCallum, A. 2012. A discriminative hierarchical model for fast coreference at large scale. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics(ACL'12)*, 379–388. Association for Computational Linguistics.

Winkler, W. E. 1990. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods*, 354–359. American Statistical Association.

Winkler, W. E. 2014. Matching and record linkage. *Wiley Interdisciplinary Reviews: Computational Statistics* 6(5):313–325.

Temporal Quasi-Semantic Visualization and Exploration of Large Scientific Publication Corpora

Victor Andrei and **Ognjen Arandjelović**
University of St Andrews, St Andrews KY16 9SX, United Kingdom

Abstract

The huge amount of information in the form of the rapidly growing corpus of scholarly literature presents a major bottleneck to research advancement. The use of artificial intelligence and modern machine learning techniques has the potential of overcoming some of the associated challenges. In this paper we introduce as our main contribution a visualization tool which enables a researcher to analyse large longitudinal corpora of scholarly literature in an intuitive, quasi-semantic fashion. The tool allows the user to search for particular topics, track their temporal interdependencies (e.g. ancestral or descendent topics), and examine their dominance within the corpus across time. Our visualization builds upon a temporal topic model capable of extracting meaningful information from large longitudinal corpora, and of tracking complex temporal changes within it. The framework comprises: (i) discretization of time into epochs, (ii) epoch-wise topic discovery using a hierarchical Dirichlet process based model, and (iii) a temporal similarity graph which allows for the modelling of complex topic changes. Unlike previously proposed methods our algorithm distinguishes between two groups of particularly challenging and pertinent topic evolution phenomena: topic splitting and speciation, and topic convergence and merging, in addition to the more widely recognized emergence and disappearance, and gradual evolution. Evaluation is performed on a public medical literature corpus concerned with the highly pertinent condition: the so-called metabolic syndrome.

1 Introduction

Recent years have witnessed a remarkable convergence of two broad trends. The first of these concerns information i.e. data – rapid technological advances coupled with an increased presence of computing in nearly every aspect of daily life, have for the first time made it possible to acquire and store massive amounts of highly diverse types of information. Concurrently and in no small part propelled by the described environment, research in artificial intelligence – in machine learning, data mining, and pattern recognition, in particular – has reached a sufficient level of methodological sophistication and maturity to process and analyse the collected data, with the aim of extracting novel and useful knowledge. Application domains pertaining to health care and emergency situations have attracted a significant amount of attention. For example, heterogeneous data collected and stored in the form of large scale longitudinal electronic health records (EHRs) is increasingly recognized as a promising target for knowledge extraction algorithms [Arandjelović, 2015b,a; Vasiljeva and Arandjelović, 2016b,c,a; Christensen and Ellingsen, 2016; Xu *et al.*, 2016], as has the diverse information content shared across social media platforms [Abel *et al.*, 2011; Agarwal *et al.*, 2011; Baucom *et al.*, 2013; Beykikhoshk *et al.*, 2014; Bollen *et al.*, 2011].

It is insightful to observe that the research community itself stands to benefit from this by means of retrospection and introspection [Blei and Lafferty, 2007; Andrei and Arandjelović, 2016]. In particular, a potential obstacle to innovation and research lies in the amount of information which needs to be organized, contextualized, and understood within the research literature corpus. This is especially the case in fields associated with a particularly fast pace of innovation and publication such as medicine and computer science. The amount of published research in these fields is immense and its growth is only continuing to accelerate, posing a clear challenge to a researcher. Even restricted to a specified field of research, the amount of published data and findings makes it impossible for a human to survey the entirety of relevant publications exhaustively which inherently leads to the question of what kind of important information or insight may go unnoticed or insufficiently appreciated.

1.1 Key challenges and our contributions

On the broad scale there are two challenges that must be overcome in order to facilitate the type of analysis argued for in the previous section. The first of these is the technical problem of knowledge extraction itself. The complex and highly heterogeneous nature of data of interest requires a sufficiently nuanced analytical framework which is capable of inferring the wide range of changes and interactions between topics over time [Beykikhoshk *et al.*, 2015b]. Our algorithm which addresses this challenge is described in Section 2. The second major challenge concerns the human-machine gap, that is, the

problem of being able to present the extracted information to a non-expert user in a manner which is intuitive and which allows the user to search, navigate, and explore information within a large longitudinal data set in a semantically meaningful manner [Chaney and Blei, 2012]. Section 3.3 describes some of the key functionalities of the visualization tool we developed which achieves this. The tool is freely available upon request.

1.2 Previous work

Unsurprisingly, the challenge of semantic knowledge extraction from large document corpora has already attracted significant research attention [Blei and Lafferty, 2006a; Andrei and Arandjelović, 2016]. Most of it has focused on so-called 'static' document collections. This means that such collections are treated as sets without any associated sequential ordering or temporal information [Blei and Lafferty, 2006a]. Under this model the documents are said to be exchangeable [Blei and Lafferty, 2006b].

A limitation of most models described in the existing literature lies in their assumption that the data corpus is static. Here the term 'static' is used to describe the lack of any associated temporal information associated with the documents in a corpus – the documents are said to be exchangeable [Blei and Lafferty, 2006b]. However, research articles are added to the literature corpus in a temporal manner and their ordering has significance. Consequently the topic structure of the corpus changes over time [Dyson, 2012; Rodriguez *et al.*, 2014; Beykikhoshk *et al.*, 2015a]: new ideas emerge, old ideas are refined, novel discoveries result in multiple ideas being related to one another thereby forming more complex concepts or a single idea multifurcating into different 'sub-ideas' etc. The premise in the present work is that documents are not exchangeable at large temporal scales but can be considered to be at short time scales, thus allowing the corpus to be treated as *temporally locally static*.

2 Proposed approach

In this section we describe the algorithm used to extract knowledge from longitudinal document collections. For the sake of completeness we begin by reviewing the relevant theory underlying Bayesian mixture models suitable for the analysis of static corpora. We then explain how the proposed algorithm builds upon and employs these static models to extract nuanced temporal changes to the topic structure. In Section 3.3 we describe a tool we developed to visualize this complex web of interactions in a manner which allows the user to explore the extracted data in an intuitive, quasi-semantic manner.

2.1 Bayesian mixture models

The structure of mixture models makes them inherently suitable for the modelling of heterogeneous data whereby heterogeneity is taken to mean that observable data is generated by more than one 'process' (also referred to as a 'source'). The key challenges lie in the lack of observability of the correspondence between specific data points and their sources, and the lack of *a priori* information on the number of sources [Richardson and Green, 1997].

Bayesian non-parametric methods place priors on the infinite-dimensional space of probability distributions and provide an elegant solution to the aforementioned modelling problems. Dirichlet Process (DP) in particular allows for the model to accommodate a potentially infinite number of mixture components [Ferguson, 1973]:

$$p\left(x|\phi_{1:\infty},\phi_{1:\infty}\right)=\sum_{k=1}^{\infty}\pi_k f\left(x|\phi_k\right). \tag{1}$$

where $\mathrm{DP}\left(\gamma,H\right)$ is defined as a distribution of a random probability measure G over a measurable space $(\Theta,\mathcal{B})$, such that for any finite measurable partition $(A_1,A_2,\ldots,A_r)$ of Θ the random vector $\left(G\left(A_1\right),\ldots,G\left(A_r\right)\right)$ is a Dirichlet distribution with parameters $\left(\gamma H\left(A_1\right),\ldots,\gamma H\left(A_r\right)\right)$. A Dirichlet process mixture model (DPM) is obtained by associating different mixture components with atoms ϕ_k, and assuming $x_i|\phi_k \overset{iid}{\sim} f\left(x_i|\phi_k\right)$ where $f\left(.\right)$ is the kernel of the mixing components.

Hierarchical DPMs

While the DPM is suitable for the clustering of exchangeable data in a single group, many real-world problems are more appropriately modelled as comprising multiple groups of exchangeable data. In such cases it is desirable to model the observations of different groups jointly, allowing them to share their generative clusters. This "sharing of statistical strength" emerges naturally when a hierarchical structure is implemented.

The DPM models each group of documents in a collection using an infinite number of topics. However, it is desired for multiple group-level DPMs to share their clusters. The hierarchical DP (HDP) [Teh *et al.*, 2006] offers a solution whereby base measures of group-level DPs are drawn from a corpus-level DP. In this way the atoms of the corpus-level DP are shared across the documents; posterior inference is readily achieved using Gibbs sampling as described by Teh *et al.* [2006].

2.2 Modelling topic evolution over time

We now show how the described HDP based model can be applied to the analysis of temporal topic changes in a *longitudinal* data corpus.

Owing to the aforementioned assumption of a temporally locally static corpus we begin by discretizing time and dividing the corpus into epochs. Each epoch spans a certain contiguous time period and has associated with it all documents with timestamps within this period. Each epoch is then modelled separately using a HDP, with models corresponding to different epochs sharing their hyperparameters and the corpus-level base measure. Hence if n is the number of epochs, we obtain n sets of topics $\boldsymbol{\phi}=\left\{\boldsymbol{\phi}_{t_1},\ldots,\boldsymbol{\phi}_{t_n}\right\}$ where $\boldsymbol{\phi}_t=\{\theta_{1,t},\ldots,\phi_{K_t,t}\}$ is the set of topics that describe epoch t, and K_t their number.

Topic relatedness

Our goal now is to track changes in the topical structure of a data corpus over time. The simplest changes of interest include the emergence of new topics, and the disappearance of

others. More subtly, we are also interested in how a specific topic changes, that is, how it evolves over time in terms of the contributions of different words it comprises. Lastly, our aim is to be able to extract and model complex structural changes of the underlying topic content which result from the interaction of topics. Specifically, topics, which can be thought of as collections of memes [Leskovec *et al.*, 2009], can merge to form new topics or indeed split into more nuanced memetic collections. This information can provide valuable insight into the refinement of ideas and findings in the scientific community, effected by new research and accumulating evidence.

The key idea behind our tracking of simple topic evolution stems from the observation that while topics may change significantly over time, changes between successive epochs are limited. Therefore we infer the continuity of a topic in one epoch by relating it to all topics in the immediately subsequent epoch which are sufficiently similar to it under a suitable similarity measure – we adopt the well known Bhattacharyya distance (BHD):

$$\rho_{\text{BHD}}(p,q) = -\ln \sum_i \sqrt{p(i)q(i)} \tag{2}$$

where $p(i)$ and $q(i)$ are two probability distributions. This approach can be seen to lead naturally to a similarity graph representation whose nodes correspond to topics and whose edges link those topics in two epochs which are related. Formally, the weight of the directed edge that links $\phi_{j,t}$, the j-th topic in epoch t, and $\phi_{k,t+1}$ is $\rho_{\text{BHD}}\left(\phi_{j,t}, \phi_{k,t+1}\right)$ where ρ_{BHD} denotes the BHD. Alternatives to the BHD, such as the Hellinger distance [Hellinger, 1909] are possible but we found no compelling theoretical reason nor a meaningful (or indeed significant) empirical difference to motivate its use over the BHD.

In constructing a similarity graph a threshold is used to eliminate automatically weak edges, retaining only the connections between sufficiently similar topics in adjacent epochs. Then the disappearance of a particular topic, the emergence of new topics, and gradual topic evolution can be determined from the structure of the graph. In particular if a node does not have any edges incident to it, the corresponding topic is taken as having emerged in the associated epoch. Similarly if no edges originate from a node, the corresponding topic is taken to vanish in the associated epoch. Lastly when exactly one edge originates from a node in one epoch and it is the only edge incident to a node in the following epoch, the topic is understood as having evolved in the sense that its memetic content may have changed.

A major challenge to the existing methods in the literature concerns the detection of topic merging and splitting. Since the connectedness of topics across epochs is based on their similarity what previous work describes as 'splitting' or indeed 'merging' does not adequately capture these phenomena. Rather, adopting the terminology from biological evolution, a more accurate description would be 'speciation' and 'convergence' respectively. The former is illustrated in Figure 1(a) whereas the latter is entirely analogous with the time arrow reversed. What the conceptual diagram shown illustrates is a slow differentiation of two topics which originate from the same 'parent'. Actual topic splitting, which does not have a biological equivalent in evolution, and which is conceptually illustrated in Figure 1(b) cannot be inferred by measuring topic similarity. Instead, in this work we propose to employ the Kullback-Leibler divergence (KLD) for this purpose. The divergence $\rho_{\text{KLD}}(p,q)$ is asymmetric and it measures the amount of information lost in the approximation of the probability distribution $p(i)$ with $q(i)$. KLD is defined as follows:

$$\rho_{\text{KLD}}(p,q) = \sum_i p(i) \ln \frac{p(i)}{q(i)} \tag{3}$$

It can be seen that a high penalty is incurred when $p(i)$ is significant and $q(i)$ is low. Hence, we use the BHD to track gradual topic evolution, speciation, and convergence, while the KLD (computed both in forward and backward directions) is used to detect topic splitting and merging.

Automatic temporal relatedness graph construction
Another novelty of the work first described in this paper concerns the building of the temporal relatedness graph. We achieve this almost entirely automatically, requiring only one free parameter to be set by the user. Moreover the meaning of the parameter is readily interpretable and understood by a non-expert, making our approach highly usable.

Our methodology comprises two stages. Firstly we consider all inter-topic connections present in the initial fully connected graph and extract the empirical estimate of the corresponding cumulative density function (CDF). Then we prune the graph based on the operating point on the relevant CDF. In other words if F_ρ is the CDF corresponding to a specific initial, fully connected graph formed using a particular similarity measure (BHD or KLD), and $\zeta \in [0,1]$ the CDF operating point, we prune the edge between topics $\phi_{j,t}$ and $\phi_{k,t+1}$ iff $\rho(\phi_{j,t}, \phi_{k,t+1}) < F_\rho^{-1}(\zeta)$.

3 Evaluation and discussion

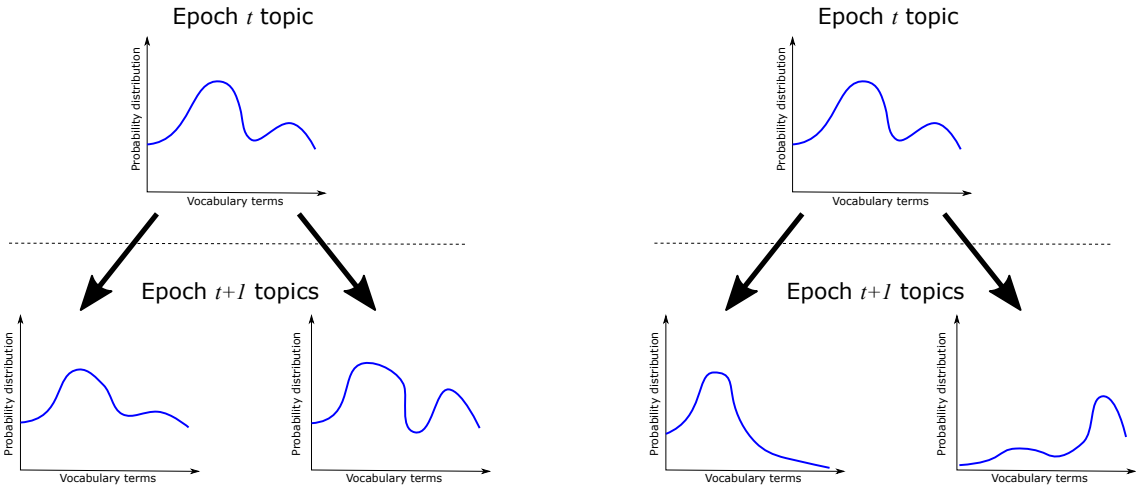
We now analyse the performance of the proposed framework empirically on a large real world data set.

3.1 Evaluation data

Herein we adopt the data set first described by Beykikhoshk *et al.* [2016] which is freely available (currently upon request) from `https://oa7.host.cs.st-andrews.ac.uk`. For full detail the reader is referred to the original publication; herein we summarize the main feature of the data set.

Raw data was collected using the PubMed interface to the US National Library of Medicine. Scholarly articles on the metabolic syndrome (MetS) and written in English were retrieved by searching with the keyphrase "metabolic syndrome". The earliest publication found was that by Berardinelli *et al.* [1953]. A corpus of 31,706 publications matching the search criteria was collected, with the chronologically last matching one having been indexed by PubMed on the 10th Jan 2016.

Pre-processing
Raw data collected from PubMed is in the form of freeform text. To prepare it for automatic analysis a series of 'preprocessing' steps were required. Broadly speaking, the goal



(a) Topic speciation (b) Topic splitting

Figure 1: Unlike previously proposed methods, our algorithm explicitly distinguishes between two important topic evolution phenomena: (a) topic speciation and (b) topic splitting. Analogous phenomena in the form of, respectively topic convergence and topic merging are similarly obtained using BHD and KLD with the associated time arrow reversed.

of pre-processing is to remove words which are largely uninformative, reduce dispersal of semantically equivalent terms, and thereafter select terms which are included in the vocabulary over which topics are learnt.

Soft lemmatization using the WordNet® lexicon [Miller, 1995] was performed first in order to normalize for word inflections. No stemming was performed to avoid semantic distortion often effected by heuristic rules used by stemming algorithms. After lemmatization and the removal of so-called stop-words, approximately 3.8 million terms were obtained when term repetitions are counted, and 46,114 when only unique terms are considered. The vocabulary was constructed by selecting the most frequent terms which explain 90% of the energy in a specific corpus, resulting in a vocabulary containing 2,839 terms.

3.2 Understanding the underlying model

Before describing the visualization framework we developed in Section 3.3, it is important to understand the structure of the model our visualization builds upon, including the nature and the complexity of the extracted topic information. Hence we begin by describing a set of quantitative experiments which illustrates some of the key differences of our temporal topic extraction algorithm.

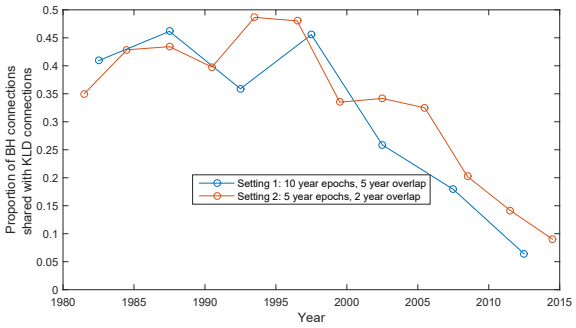
We first demonstrate how our use of the two topic relatedness measures (BHD and KLD) effectively captures different aspects of topic relatedness. To obtain a quantitative measure we looked at the number of inter-topic connections formed in respective graphs both when the BHD is used as well as when the KLD is applied instead. The results were normalized by the total number of connections formed between two epochs, to account for changes in the total number of topics across time. Our results are summarized in Figure 2. A significant difference between the two graphs is readily evident – across the entire timespan of the data corpus, the number of Bhattacharyya distance based connections also formed through the use of the KLD is less than 40% and in most cases less than 30%. An even greater difference is seen when the proportion of the KLD connections is examined – it is always less than 25% and most of the time less than 15%.

To get an even deeper insight into the contribution of the two relatedness measures, we examined the corresponding topic graphs before edge pruning. The plot in Figure 3 shows the variation in inter-topic edge strengths computed using the BHD and the KLD (in forward and backward directions) – the former as the x coordinate of a point corresponding to a pair of topics, and the latter as its y coordinate. The scatter of data in the plot corroborates our previous observation that the two similarity measures indeed do capture different aspects of topic behaviour.

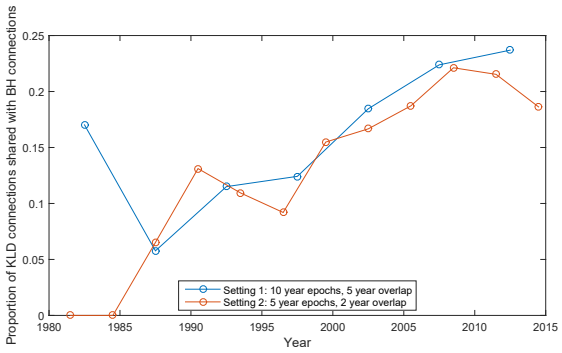
3.3 Interactive exploration and visualization

Our final contribution comprises a web application which allows users to upload and analyse their data sets using the proposed framework. A screenshot of the initial window of the application when a data set is loaded is shown in Figure 4. Topics are visualized as coloured blobs arranged in rows, each row corresponding to a single epoch (with the time arrow pointing downwards). The size of each blob is proportional to the popularity of the corresponding topic within its epoch. Each epoch (that is to say, the set of topics associated with a single epoch) is coloured using a single colour different from the neighbouring epochs for easier visualization and navigation. Line connectors between topics denote temporal topic connections i.e. the connections in the resultant temporal relatedness graph which, as explained in the previous section, depending on the local graph structure encode topic evolution, merging and splitting, and convergence and speciation.

The application allows a range of powerful tasks to be per-



(a) BHD-KLD normalized overlap



(b) KLD-BHD normalized overlap

Figure 2: The proportion of topic connections shared between the BHD and the KLD temporal relatedness graphs, normalized by (a) the number of BHD connections, and (b) the number of KLD connections, in an epoch.

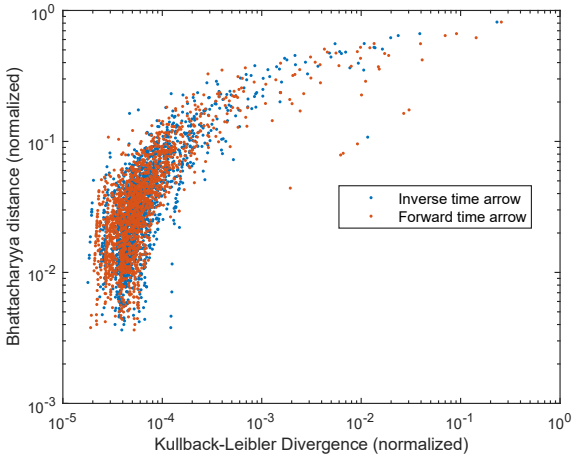


Figure 3: Relationship between inter-topic edge strengths computed using the BHD and the KLD before the pruning of the respective graphs.

formed quickly and in an intuitive manner. For example, the user can search for a given topic using keywords (and obtain a ranked list), trace the origin of a specific topic backwards in time, or follow its development in the forward direction, examine word clouds associated with topics, display a range of statistical analyses, or navigate the temporal relatedness graph freely. Some of these capabilities are showcased in Figure 5. In particular, the central part of the screen shows a selected topic and the strongest ancestral and descendent lineages. The search box on the left hand side can be used to enter multiple terms which are used to retrieve and rank topics by quality of fit to the query. Finally, on the right hand side relevant information about the currently selected topic is summarized: its most popular terms are both visualized in the form of a colour coded word cloud, as well as listed in order in plain text underneath. Additional graph navigation options include magnification tools accessible from the bottom of the screen, whereas translation is readily performed by simply dragging the graph using a mouse or a touchpad. The application and its code are freely available upon request.

4 Summary and Conclusions

In this work we addressed some of the challenges faced by researchers in the task of analysing, organizing, and searching through large corpora of scientific literature. We described a framework for the visualization of a recently introduced complex temporal topic model. The model is based on non-parametric Bayesian techniques which is able to extract and track complex, semantically meaningful changes to the topic structure of a longitudinal document corpus and is the first such model capable of differentiating between two types of topic structure changes, namely topic splitting and what we termed topic speciation. Built upon this model is a sophisticated web based visualization tool which enables a researcher to analyse literature in an intuitive, quasi-semantic fashion. The tool allows the user to search for particular topics, track their temporal interdependencies (e.g. ancestral or descendent topics), and examine their dominance within the corpus across time. Experiments on a large corpus of medical literature concerned with the metabolic syndrome was used to illustrate our contributions.

References

F. Abel, Q. Gao, G. J. Houben, and K. Tao. Analyzing user modeling on Twitter for personalized news recommendations. *In Proc. International Conference User Modeling, Adaptation and Personalization*, pages 1–12, 2011.

A. Agarwal, B. Xie, I Vovsha, O. Rambow, and R. Passonneau. Sentiment analysis of Twitter data. *In Proc. Workshop on Language in Social Media*, pages 30–38, 2011.

V. Andrei and O. Arandjelović. Identification of promising research directions using machine learning aided medical literature analysis. *In Proc. International Conference of the IEEE Engineering in Medicine and Biology Society*, 2016.

O. Arandjelović. Discovering hospital admission patterns using models learnt from electronic hospital records. *Bioinformatics*, 31(24):3970–3976, 2015.

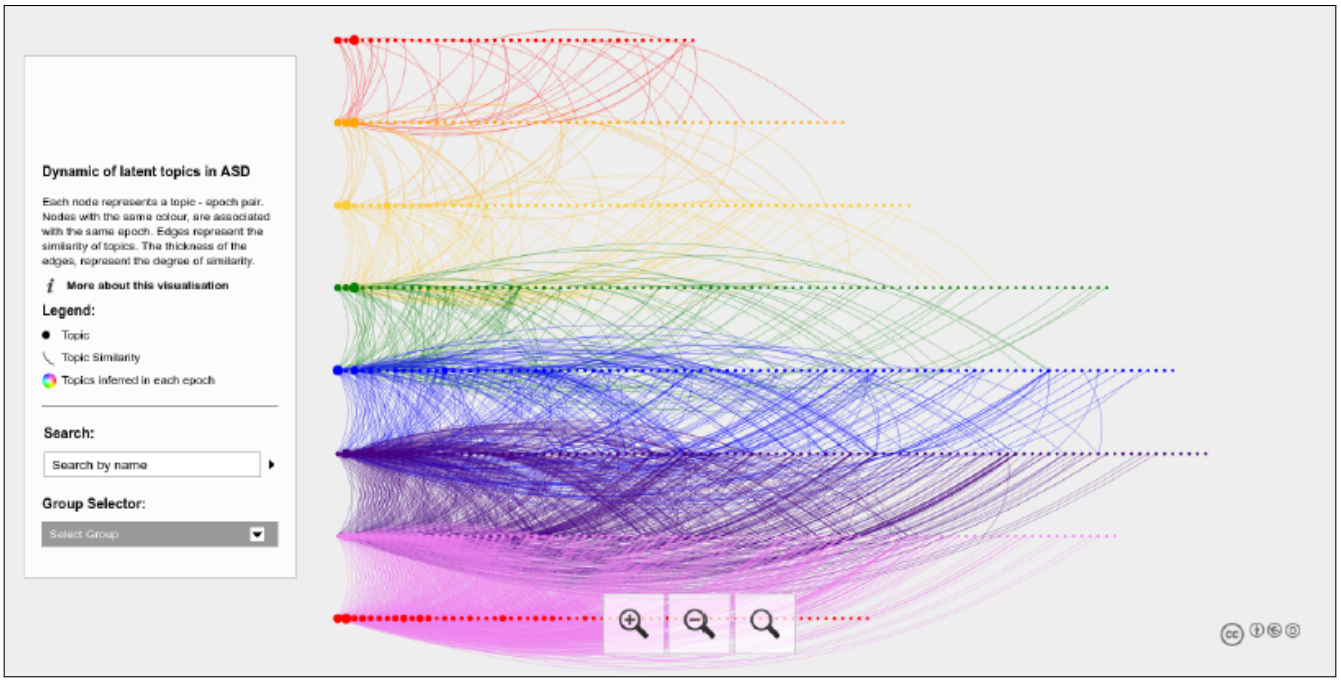


Figure 4: Screenshot of the initial window of the application we developed for free public analysis of custom data sets using the method described in the present paper. Topics are visualized as coloured blobs arranged in rows, each row corresponding to a single epoch (with the time arrow pointing downwards). The size of each blob is proportional to the popularity of the corresponding topic within its epoch. Each epoch is coloured using a single colour different from the neighbouring epochs. Line connectors between topics denote temporal topic connections across the temporal relatedness graph and encode topic evolution, merging and splitting, and convergence and speciation.

O. Arandjelović. Prediction of health outcomes using big (health) data. *In Proc. International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2543–2546, August 2015.

E. Baucom, A. Sanjari, X. Liu, and M. Chen. Mirroring the real world in social media: Twitter, geolocation, and sentiment analysis. *In Proc. International Workshop on Mining Unstructured Big Data Using Natural Language Processing*, pages 61–68, 2013.

W. Berardinelli, J. G. Cordeiro, D. de Albuquerque, and A. Couceiro. A new endocrine-metabolic syndrome probably due to a global hyperfunction of the somatotrophin. *Acta Endocrinologica*, 12(1):69–80, 1953.

A. Beykikhoshk, O. Arandjelović, D. Phung, S. Venkatesh, and T. Caelli. Data-mining Twitter and the autism spectrum disorder: a pilot study. *In Proc. IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, pages 349–356, 2014.

A. Beykikhoshk, O. Arandjelović, D. Phung, and S. Venkatesh. Overcoming data scarcity of Twitter: using tweets as bootstrap with application to autism-related topic content analysis. *In Proc. IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, pages 1354–1361, August 2015.

A. Beykikhoshk, O. Arandjelović, D. Phung, S. Venkatesh, and T. Caelli. Using Twitter to learn about the autism community. *Social Network Analysis and Mining*, 5(1):5–22, 2015.

A. Beykikhoshk, O. Arandjelović, D. Phung, and S. Venkatesh. Discovering topic structures of a temporally evolving document corpus. *arXiv preprint*, page 1512.08008, 2016.

D. Blei and J. Lafferty. Correlated topic models. *Advances in Neural Information Processing Systems*, 18:147, 2006.

D. Blei and J. Lafferty. Dynamic topic models. *In Proc. IMLS International Conference on Machine Learning*, pages 113–120, 2006.

D. Blei and J. Lafferty. A correlated topic model of Science. *Annals of Applied Statistics*, 1(1):17–35, 2007.

J. Bollen, H. Mao, and A. Pepe. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *In Proc. International Conference on Weblogs and Social Media*, pages 450–453, 2011.

A. J. B. Chaney and D. M. Blei. Visualizing topic models. *In Proc. International Conference on Web and Social Media*, 2012.

B. Christensen and G. Ellingsen. Evaluating model-driven development for large-scale EHRs through the openEHR approach. *Int J Med Inform*, 89:43–54, 2016.

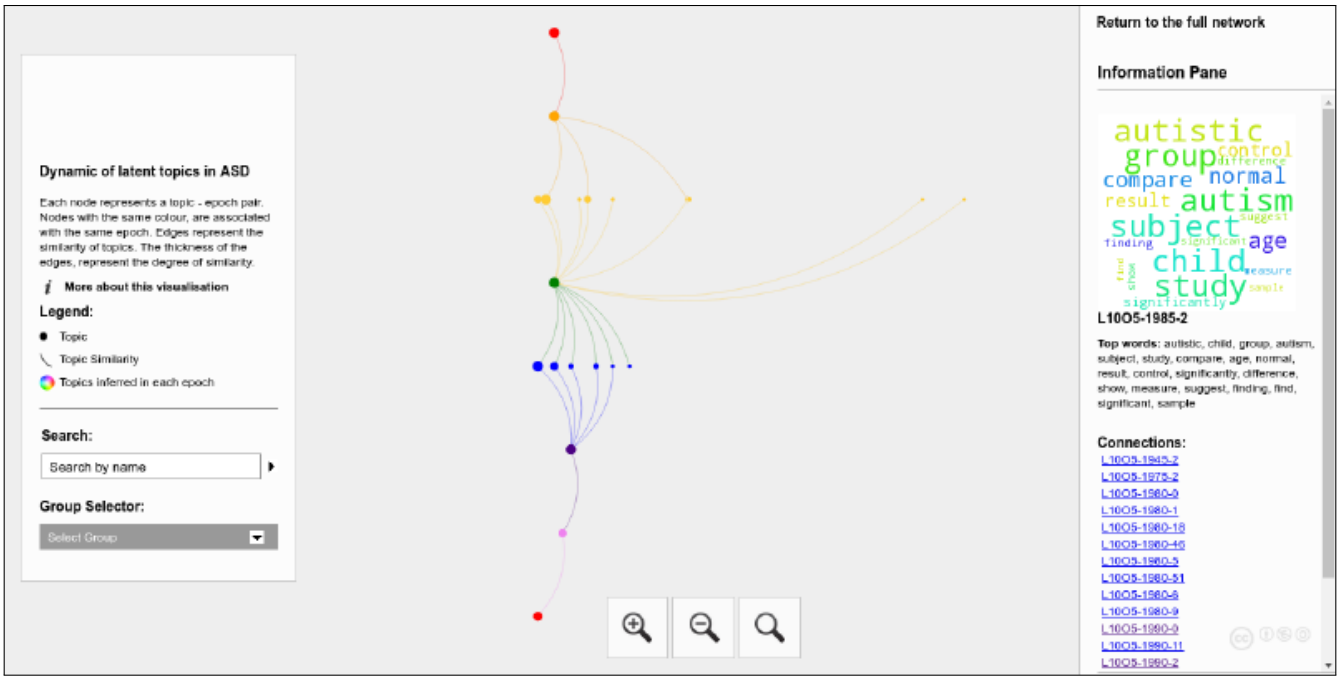


Figure 5: Illustration of some of the capabilities of the developed application: (i) the central part of the screen shows a selected topic and the strongest ancestral and descendent lineages, (ii) the search box on the left hand side can be used to enter multiple terms which are used to retrieve and rank topics, and (iii) on the right hand side relevant information about the currently selected topic is summarized. Magnification tools are accessible from the bottom of the screen, whereas translation can be performed using a simple dragging motion.

F. J. Dyson. Is science mostly driven by ideas or by tools? *Science*, 338(6113):1426–1427, 2012.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, pages 209–230, 1973.

E. Hellinger. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 136:210–271, 1909.

J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. *In Proc. ACM/SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 497–506, 2009.

G. A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

S. Richardson and P. J. Green. On Bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society*, 59(4):731–792, 1997.

M. G. Rodriguez, J. Leskovec, D. Balduzzi, and B. Schölkopf. Uncovering the structure and temporal dynamics of information propagation. *Network Science*, 2(1):26–65, 2014.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

I. Vasiljeva and O. Arandjelović. Automatic knowledge extraction from EHRs. *In Proc. International Joint Conference on Artificial Intelligence Workshop on Knowledge Discovery in Healthcare Data*, 2016.

I. Vasiljeva and O. Arandjelović. Prediction of future hospital admissions – what is the tradeoff between specificity and accuracy? *In Proc. International Conference on Bioinformatics and Computational Biology*, 2016.

I. Vasiljeva and O. Arandjelović. Towards sophisticated learning from EHRs: increasing prediction specificity and accuracy using clinically meaningful risk criteria. *In Proc. International Conference of the IEEE Engineering in Medicine and Biology Society*, 2016.

L. Xu, D. Wen, X. Zhang, and J. Lei. Assessing and comparing the usability of Chinese EHRs used in two Peking University hospitals to EHRs used in the US: A method of RUA. *Int J Med Inform*, 89:32–42, 2016.

Identifying Academic Papers in Computer Science Based on Text Classification

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 the arXiv data set. Each paper in arXiv 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 from MEDLINE database. On a corpus of 2,889 abstracts, they reported a precision of 0.70 when the recall is 0.25. (Kodakateri Pudhiyaveetil et al. 2009) categorized CiteSeer papers into 268 categories based on the ACM CCS class definition. They used the k-NN algorithm to train their classifier with 268 classes, each class has 10 sampled papers. For each test paper, they output the top K predicted classes. They did not 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 feature 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.

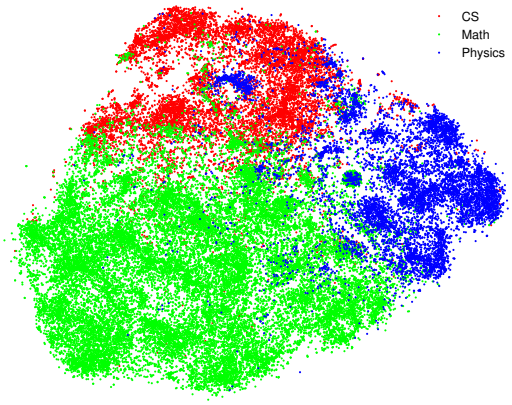


Figure 1: Three classes of documents. Vectors with 100 dimensions are trained using Sentence2vec. Then they are reduced to two dimension using t-SNE.

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 extracting 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 learn the embeddings of sentence from the training data sets. 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 over 2,000 embeddings per second per CPU 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.

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.

		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.

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.

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

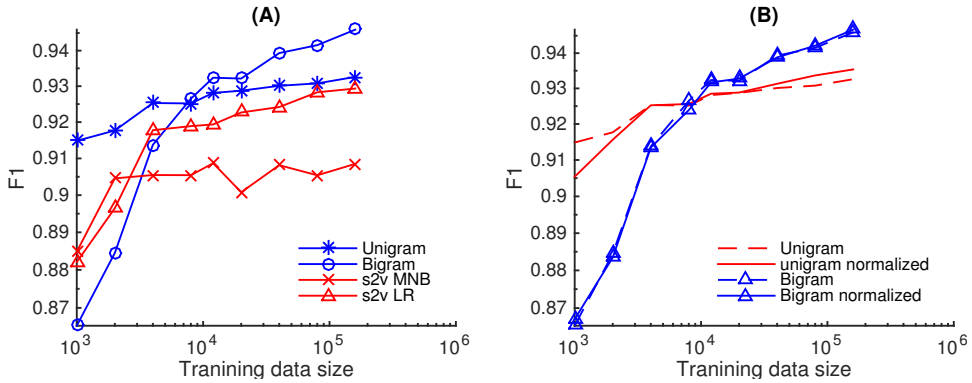


Figure 2: Impact of training data size on F_1. (A) Both unigram and bigram model out-perform sentence2vec when training data is large; (B) Normalization plays a minor role in this data.

		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.

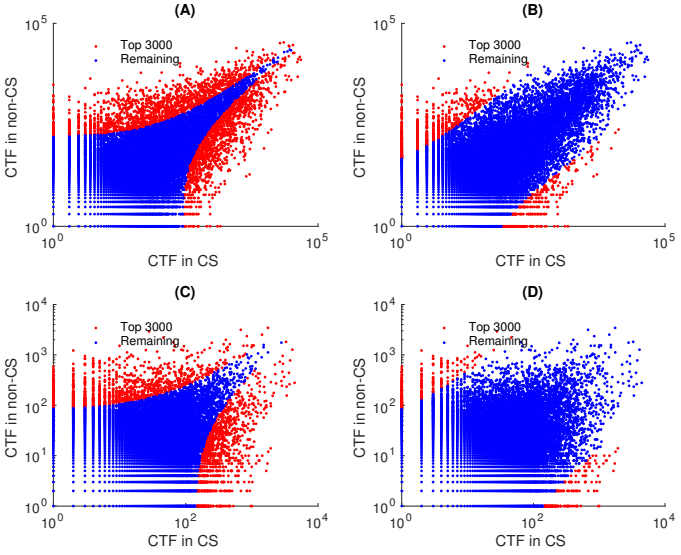


Figure 3: Top features selected by different methods. Panel (A) and (C): χ^2; (B) and (D): PMI. (A) and (B) are unigram models; (C) and (D) are bigram models.

unigram			bigram		
Rank	Name	CTF	Rank	Name	CTF
1	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

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

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. Accordingly, 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.

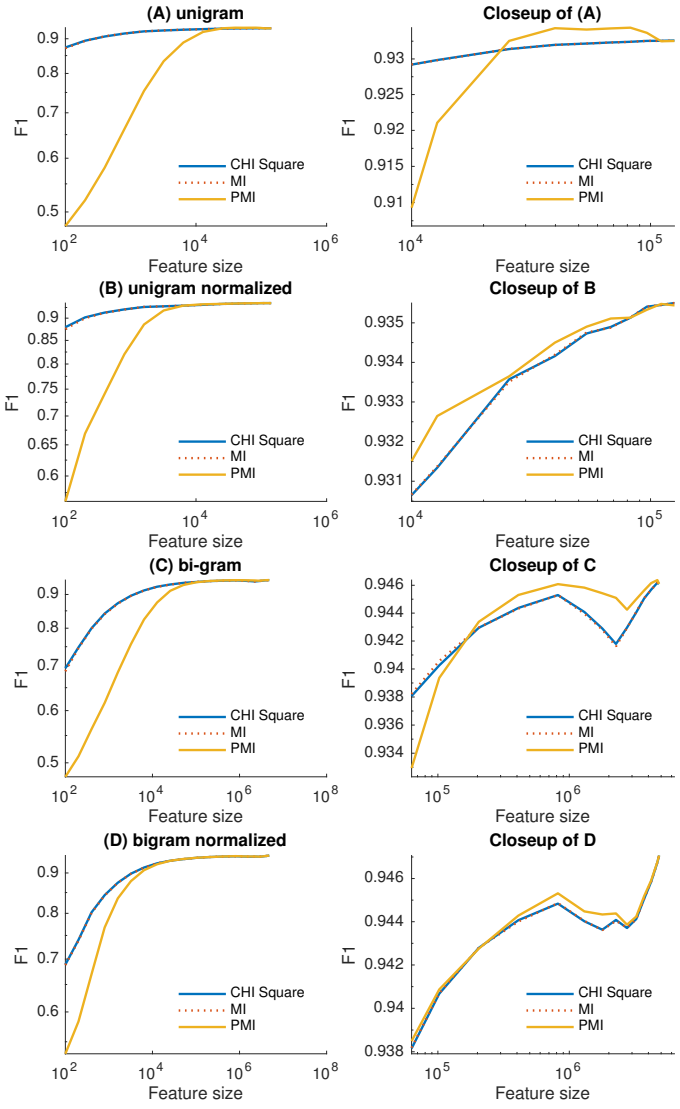


Figure 4: Impact of feature size for unigram and bigram models, with combination of text normalization. (A) Unigram; (B) unigram normalized; (C) bigram; (D) bigram normalized.

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 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.

Near-duplicated Documents in CiteSeerX

Yi Zhang and **Jianguo Lu**
School of Computer Science, University of Windsor
401 Sunset Avenue, Windsor, Ontario N9B 3P4. Canada

Abstract

Academic literatures, especially those in the field of computer science, are often posted multiple times on the Web. Scholarly index engines, such as Google Scholar and CiteSeerX, crawl such documents from the open web as well as publishers. To improve the quality of the search result, there is a need to detect and coalesce duplicate or very similar (hereafter called near-duplicate) papers. Near-duplicate detection is computationally expensive. Pair-wise comparison of millions of papers is not feasible even for the most advanced machines. We combine SimHash and Jaccard similarity to discover near-duplicate documents in a CiteSeerX data set, which contains 2,118,122 full-text academic papers. We observe that 12% documents in CiteSeerX have near-duplicates with Jaccard similarity larger than 0.9. Then we study the near-duplicates and summarize six leading causes. We also compare these near-duplicates with those appeared only once on the web. We find that the citation count grows almost linearly with the number of duplications.

Introduction

Academic literatures, particularly the ones in computer science, often occur multiple times on the Web. Researchers may post drafts on their personal websites. Then, the same paper may occur in arXiv, or proceedings of conferences, etc. In addition, such documents may occur in course web pages. Each version may differ slightly, mostly in the publisher's information, or slight difference in content. Documents with slight difference are called near-duplicates. The threshold for the similarity depends on the application. This paper regards two documents are near-duplicates if their Jaccard similarity of their trigrams exceeds 0.9.

Detecting near-duplicates is an essential component of a search engine. Although the detection of near-duplicate of web pages has been studied extensively (Broder 1997; Broder et al. 1997), there are only a few works focus on the academic literatures (). The questions we want to answer are: 1) How many academic literatures on the web are near-duplicates? How similar are they? 2) What cause these near-duplicates? 3) What are the patterns of such multiple postings? 4) Whether multiple occurrences correlate to the citation number of the documents?

To answer these questions, we need to conduct an experiment on a large scale dataset. Our experiment is conducted on CiteSeerX datasets, which contain over two million full-text academic literatures. When detecting the near-duplicates, we want to tolerate slight difference between documents. Comparing the similarity in such a big dataset is computationally expensive. Thus, an efficient algorithm is required. In this paper, we first evaluate the state-of-art SimHash algorithm on the academic literatures. For web pages, it is reported to set Hamming distance $k = 3$ to achieve 75% accuracy. We find that the accuracy is higher in general for research papers. We also notice that the recall of SimHash can be as high as 99% when retrieving the near-duplicated literatures. By combining SimHash and Jaccard similarity, we successfully discovered 271,906 distinct near-duplicates with high accuracy, which contribute 12.84% of the CiteSeerX dataset. By studying these near-duplicates, we summarize 6 leading causes. And we also compare the categories and publishing years of the near-duplicated documents with the one appeared only once on the web.

We also observe that paper with more near-duplicates are cited more often. Moreover, it is interesting to see the citation counts grow almost linearly with duplicate occurrences.

Literature Review

Broder et. al. studied the near-duplicates on the AltaVista search engine by calculating the MinHash of the documents (Broder 1998). After that, numerous improvements have been proposed (Fetterly et al. 2003; Henzinger 2006; Hajishirzi, Yih, and Kolcz 2010).

SimHash is one of the most widely used near-duplicates detection algorithms, which is first introduced by Manku et. al. in 2007 (Manku, Jain, and Das Sarma 2007). After testing on three billion documents, the authors reported that the optimal Hamming distance of SimHash is three, which is the break-even point of precision and recall. Later on, Sood et. al. (Sood and Loguinov 2011) studied the recall of SimHash. They pointed out that SimHash can be faster and less space consumption by sacrificing a small percentage of recall.

Although near-duplicates detection is well studied, most of the existing works mainly focus on general documents, especially the web page crawled by search engine. Only a few works mention about the near-duplicates in academic literatures. In 2013, Williams et. al. used SimHash to re-

move duplicate documents in CiteSeerX(Williams and Giles 2013) and obtained F-score 0.91. Later on, they released a website called SimSeerX(Williams, Wu, and Giles 2014) for locating near-duplicate papers. However, they did not summarize the duplicate literatures.

Compared with the existing works, we give a better view of the near-duplicates in academic literatures. Our experiment discovers the near-duplicated documents with high accuracy. Most importantly, we analyse the features of the near-duplicates.

Near-Duplicates Detection

If two documents share terms in large quantities, we call them near-duplicates. A common technique for near-duplicate detection is to break documents into a sequence of consecutive tokens called n-grams (Broder et al. 1997). Then the similarity between the documents can be measured by Jaccard similarity. Given two documents, the Jaccard similarity between their shingles A and B is defined as

$$JS(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

When the Jaccard similarity of these two documents $JS(A,B)$ is larger than a threshold t, where t is a number that close to 1, we say these two documents are near-duplicates.

Generally speaking, Jaccard similarity is a good measure of the similarity of two sets (Henzinger 2006; Broder et al. 1997). Thus, in this paper, we treat Jaccard similarity as the ground true similarity between documents.

However, calculating Jaccard similarity directly is very costly, especially when we want to find near-duplicates in a large-scaled dataset. Sampling based techniques, such as MinHash(Broder 1997), has been proposed to reduce the computation time. SimHash, proposed by Manku et. al (Manku, Jain, and Das Sarma 2007), is one of the most widely applied near-duplicates detection algorithm. It maps the tokens of a document into a fixed bit fingerprint. Then the similarity can be estimated by the Hamming distance of the fingerprint. The algorithm can be described as follow:

- Split each document into n-grams (here we use trigrams);
- Initialize a K-dimensional zero vector V;
- Get a K-bit hash value for each trigram;
- For each hash value, if the i-th hash value is 1, then the i-th-bit of V increases by 1; if the i-th-bit hash value is 0, then the i-th bit of V decreases by 1;
- Then normalize V by marking the positive vectors as 1, and all the other vectors as 0. Thus, the fingerprint V can be stored as a K-bit integer.

After mapping each document into a K-bit fingerprint, where $K = 64$ in most applications, the similarity between two documents can be estimated by the Hamming distance of their SimHash fingerprints. Instead of computing all the possible pairs in the dataset, SimHash hash the fingerprints into certain tables to reduce the computation time. According to Pigeonhole principle(Herstein 2006), if n items are put into m containers, with $n > m$, then at least one container must contain more than one item. Thus, we can build an index for the SimHash values by splitting the K-bit fingerprint into $k+1$ sub-fingerprint. When we need to find the near-duplicates of a document, we can find all the candidates by fetching out all the related documents that have the same sub-fingerprints.

Parameters

SimHash requires a detection threshold – Hamming distance k between the similar documents. Two documents can be treated as near-duplicates if the Hamming distance between their fingerprints is less or equal to k. Most existing works use $k = 3$, which is first reported by Manku when detecting the near-duplicates among web pages (Manku, Jain, and Das Sarma 2007). However, scholarly documents are different from web pages. For example, web pages crawled from different websites may share the advertisement in the text; Academic documents do not overlap a lot due to the restriction of copyright; Academic documents are normally longer than web pages.

Next, we need to find out the performance of SimHash algorithm on scholarly literatures. In this paper, we collect a sample of 20,000 possible duplicated documents from CiteSeerX by matching their metadata. Then, we break the documents into trigrams and compute the pair-wised Hamming distances of the SimHash and the corresponding Jaccard similarities.

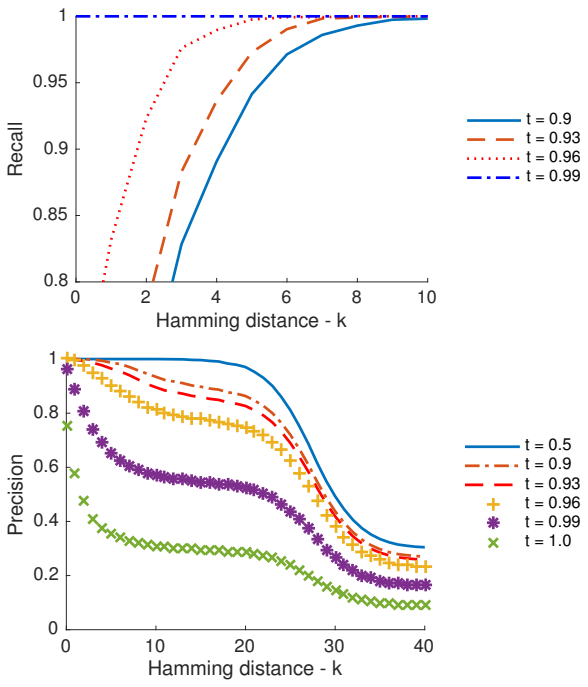


Figure 1: Precision and Recall with different threshold t

Fig. 1 shows Hamming distance threshold k against precision and recall. From the top panel, we observe that given a fixed detecting threshold t, the higher k is, the higher recall

we can achieve. For example, suppose we want to find near-duplicated documents that overlap 90%, when $k = 3$ the recall is about 75%, which means about 25% near-duplicates can not be found. In our work, we want to exam the features of near-duplicated academic literatures, thus, we need to get as many duplicated documents as we can. Therefore, a high recall is demand, which means we need to set k as high as possible.

However, a higher k may leads to two problems. The first is the precision. A higher k means less accuracy. Bottom panel of Fig. 1 shows that the precision decreases when a higher k is selected. The rate of deterioration accelerates when k becomes larger. The second, but larger problem is the time complexity. Fig. 2 shows the run time against k for different size of the datasets. Note that it is the execution time on a powerful server. Runtime increases exponentially with the growth of k. For a smaller data set that contains 20,000 documents, such growth of time is tolerable. For the dataset which contains 2 million documents, it needs around 6 hours to run if $k = 8$ and needs many days to finish if $k = 20$. Thus, in the later experiment, we set $k = 8$.

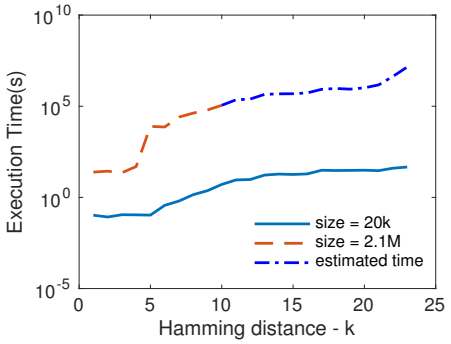


Figure 2: Runtime against Hamming distance k on different sizes of dataset

Next, we need to set a detecting threshold t for Jaccard similarity. Figure 3 shows precision and recall in different t with $k = 8$. As we can see, to guarantee the high recall of the near duplicates, particularly when documents do not overlap a lot, we need to set t as high as possible. But, a larger t could result in less number of near-duplicates. For example, when $t = 0.99$, we find 0.83 million pairs of near-duplicates. While $t = 0.90$, the number grows to 0.9 7 million. Our goal is to find as many near-duplicated documents as we can. In our work, we set $t = 0.9$ to balance the recall and the number of near-duplicates. With such setup we have a recall equals to 0.985 and accuracy of SimHash is 0.9581%. Meanwhile, to further boost the accuracy of the results, we calculate the Jaccard similarity of the document pairs captured by SimHash, and then save those have Jaccard similarity larger than detection threshold t.

Experiment and Results

Experiment Setup

CiteSeerX (Giles, Bollacker, and Lawrence 1998; Bollacker, Lawrence, and Giles 1998) is a scholarly index engine which

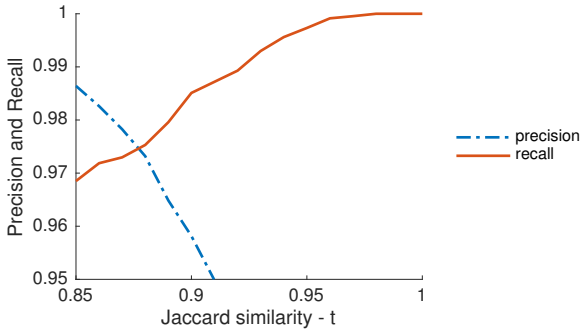


Figure 3: Relation between Jaccard similarity and precision / recall when $k = 8$

contains over 2 million full text academic literatures that are crawled from the open Web. It provides an OAI collection system that allows researchers download CiteSeerX data. The CiteSeerX data contains metadata, PDF files and corresponding text files that are extracted automatically using Prescript (Giles, Bollacker, and Lawrence 1998). In our experiment, we collect all the available text files (2,118,112 in total) from CiteSeerX OAI, then break these files into trigrams. Go-Language has been used to support parallelism computation. The program is executed on a PowerEdge R720 server that has 24 cores and 256GB memory.

Results

With detecting Hamming distance $k = 8$, the program takes 6 hours and 37 minutes to finish. In total, we discover 604,596 pairs of near-duplicates, which contains 364,930 distinct documents. The near-duplicates have been separated by their Hamming distance and Jaccard similarity in Tab. 1 and Tab. 2. From Tab. 1 we can see that most near-duplicated pairs have Hamming distance of 0. While it is interesting to see that they distribute evenly among other values. Meanwhile, most pairs concentrate in the range 0.9-1 as shows in Tab. 2, which means two documents are either very close or very different.

Hamming distance	pairs #	distinct documents #
0	397,112	79,461
1	19,767	35,666
2	23,165	42,147
3	24,441	43,903
4	25,437	45,998
5	26,483	47,368
6	27,237	48,576
7	29,298	51,362
8	31,656	54,790
sum	604,596	364,930

Table 1: Near-duplicates distribution by SimHash with $k = 8$

According to our previous evaluation, we can estimate the population of near-duplicates showed in Fig. 4. Because dif-

JS range	Pairs #	distinct documents #
0.9 - 1.0	524,888	270,906
0.8 - 0.9	51,161	61,847
0.7 - 0.8	19,826	22,491
0.6 - 0.7	6,164	6,518
0.5 - 0.6	1,491	1,576
0 - 0.5	1,066	1,592
sum	604,596	364,930

Table 2: Captured pairs and documents grouped by Jaccard similarity

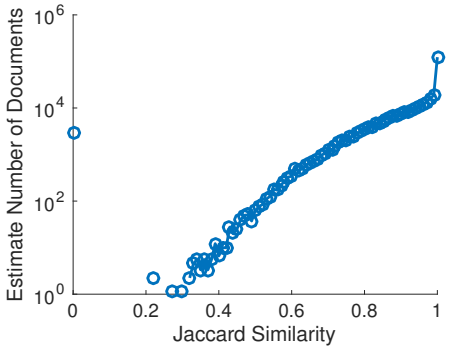


Figure 4: Estimated population of duplicated documents against Jaccard similarity

ferent Jaccard similarity ranges have different recall values, the estimations are augmented with different factors.

Discussion

Why do these near-duplicates appear multiple time on the Web? What cause the near-duplicates? To answer these questions, we randomly select the near-duplicates pairs and manually verified them. We summarize 6 leading causes of the near-duplicates as follow: 1) Same file publishes in different web pages. 2) Different stylesheet. Some literatures are written with LaTex. Different stylesheet, especially the format of citations and references, can cause slightly difference in the compiled PDF files. These near-duplicates often share large Jaccard similarities. 3) Different versions of documents. Some documents, books or reference manuals, have different published versions. 4) Error pages caused by the incorrect crawling process. CiteSeerX crawls the documents from the Internet. When crawling a web page, the server may not response correctly. Thus, an error page returns and is stored in the CiteSeerX database. These error pages from same website carry the same information, thus resulting in near-duplicates. 5) Extraction error. PDF is a common document format. CiteSeerX uses Prescript to extract raw text from PDF. However, not every PDF can be extracted correctly. Some PDF files have been extracted into short text files which contains only certain keywords (Introduction, abstract, References et.). 6) Class assignments also appeared in the CiteSeerX. These documents are not exactly academic literatures and should be removed from the collection.

We also notice that near-duplicates caused by incorrect crawling and extracting process. These documents sometime not share large similarities and are shorter than academic literatures. Meanwhile, they appear around one to two hundred times, which makes them easy to be detected and cleaned. It is possible to train a classifier to identify such near-duplicates, which we leave as the future work.

Next, we investigate the distribution of the duplicate occurrences showed in Fig. 5. The top panel in Fig. 5 shows the frequency of duplicate occurrences. From the plot, we can see that the distribution of duplicate occurrences follows a power-law. More than 100,000 documents only duplicate once (duplicate occurrence = 2) and more than 10,000 documents have duplicate occurrences = 3. Only a few documents have hundreds of near-duplicates.

The bottom panel is the duplicate occurrences against their ranks. It shows that the top one duplicated 807 times. The next repeats 195 times. We find that these files contain only one character. After that, there are a group of "nonsense" documents that are generated artificially to test Google Scholar's crawling and indexing strategy. They are repeated around one hundred times. These type of near-duplicates may be caused by incorrect crawling processing. Here, we removed these near-duplicates in the following observation.

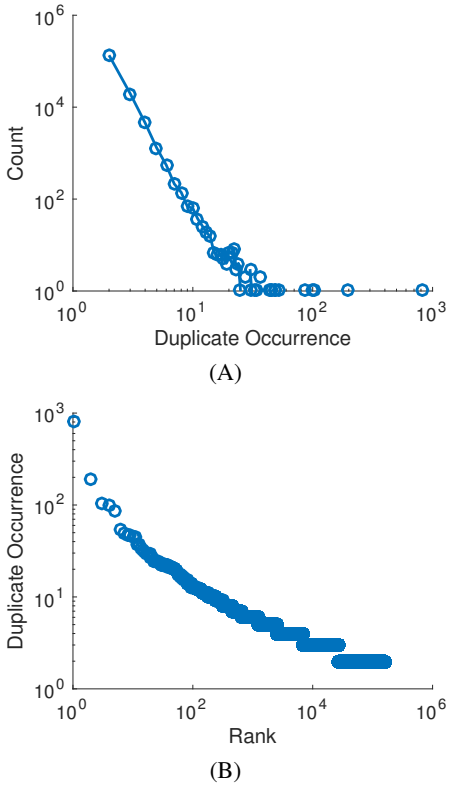


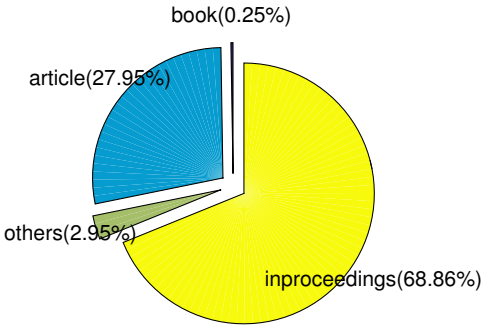
Figure 5: Distribution of near-duplicates.

Table 3 lists the top-10 duplicated documents in CiteSeerX. The most duplicated one is "Linearity and the pi-

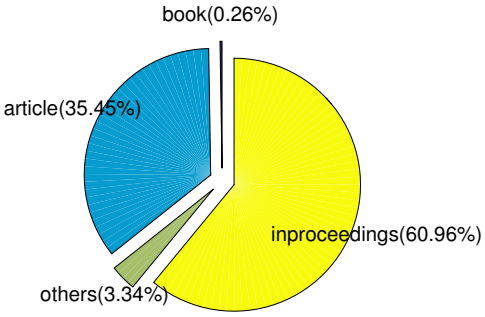
calculus" by Kobayashi, which repeats 49 times. Following is the "Serverless Network File Systems" and "Application Performance and Flexibility on Exokernel Systems". Interestingly, we notice that the duplicated documents not just include published books or papers, but also have some documents for the industry.

Then we list the difference of the document types between duplicated and non-duplicated literatures. Figure 6 shows the media types of the duplicated literatures. CiteSeerX split the document into 7 categories, and we keep the three major components and put the rest into "others". From the figure, we can see that "in-proceedings" contributes 68.86% in the duplicated documents and 60.96% in the non-duplicated ones.

The percentages of "books" of duplicated and non-duplicated documents are nearly the same(0.25%). While "article" contributes 27.95% in the duplicated documents and 35.45% in the non-duplicated ones. Category "others" in the top is 2.95%, which is slightly smaller than the one in the bottom.



(A) Duplicated documents



(B) Non-duplicated documents

Figure 6: Categories of duplicated and non-duplicated documents.

Next, we study the features of the near-duplicates. The top panel of Fig. 7 shows the distribution of publish years of duplicated and non-duplicated documents. It is interesting to see that they share the similar distribution. Documents that are published in earlier years have a higher chance to be seen, but they do not have many near-duplicates. The bottom panel shows the box plot of duplicate occurrence against publish years. We notice that older documents intend to have more near-duplicates.

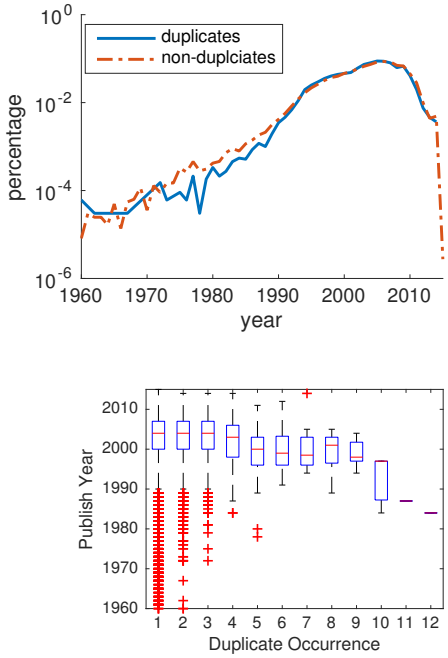


Figure 7: Publish years of documents.

Citation number is another important index to evaluate the quality of the literatures. We extract the citation count from the CiteSeerX metadata and show the relation between duplicate occurrence and citation count in Fig. 8. Panel A shows the average citation counts over duplicate occurrences. Note that there are a few outliers due to the sparse of the data when duplicate occurrences are large. According to the duplicate distribution in Figure 5(A), there are only about 20 data points for duplicate occurrences that are larger than 9. Thereby we ignore those sparse data and focus on the duplicate occurrences that are less than 9 in Panel B. From panel B we can see that the citation count grows linearly with the duplicate occurrence with Pearson Correlation 0.8894, particularly when the data points are abundance. Panel C is the box plot to show the dispersion of the data.

Conclusion

In this paper, we used SimHash and Jaccard similarity to detect near-duplicates in CiteSeerX. We found that SimHash needs to be set appropriate to balance the computational cost and accuracy. By combining SimHash and Jaccard similarity, we successfully retrieved most of the near-duplicates with Jaccard similarity larger than 0.9. We reported CiteSeerX has 12.79% documents duplicated more than once. This finding calls for further work to clean the CiteSeerX data for the construction scholarly search engines.

We also analyzed the near-duplicates, studied their features and made the observation of the relationship between duplicate occurrences with different features. The most interesting observation is that the citation count grows almost linearly with duplicate occurrence. This observation is important for both researchers and practitioners in search en-

Rank	Occurrence	Type	Document name
1	49	paper	Linearity and the pi-calculus
2	16	paper	Serverless Network File Systems
3	15	paper	Application Performance and Flexibility on Exokernel Systems
4	14	paper	A Fast File System for UNIX*
5	13	paper	A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols
6	13	article	Hints for Computer System Design
7	13	article	Security Architecture for the Internet Protocol
8	12	paper	Password Security: A Case History
9	12	paper	End-To-End Arguments in System Design
10	12	article	SWI-Prolog - Reference Manual

Table 3: Top-10 duplicated documents

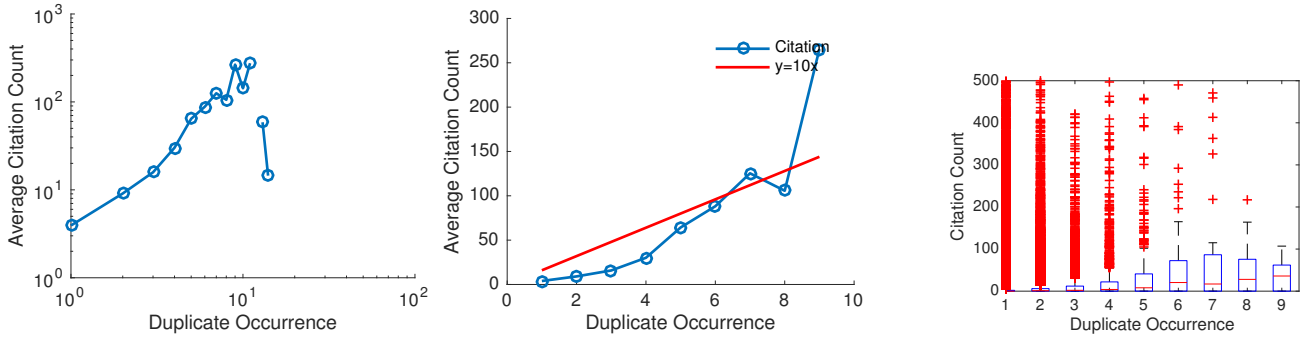


Figure 8: Citation count against duplicate occurrence. (A) average citation count against duplicate occurrence; (B) Zoom-in of plot (A) ; (C) box plot.

gine industry. The gems of scientific works are often publicly available on the web in multiple locations. For researchers, it will be awarded if your publications are listed publicly on the Web. For the construction of academic search engines, we should crawl the open Web to find the gems in science.

Acknowledgement

This work is supported by NSERC Discovery program and Cross Border Institute. We would like to thank Zhou Tong for providing the citation count on the CiteSeerX dataset.

References

Bollacker, K. D.; Lawrence, S.; and Giles, C. L. 1998. CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS '98, 116–123. ACM.

Broder, A. Z.; Glassman, S. C.; Manasse, M. S.; and Zweig, G. 1997. Syntactic clustering of the web. 29(8):1157–1166.

Broder, A. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, 21–29.

Broder, A. Z. 1998. Filtering near-duplicate documents. In *Proc. FUN 98*.

Fetterly, D.; Manasse, M.; Najork, M.; and Wiener, J. 2003. A large-scale study of the evolution of web pages. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, 669–678. ACM.

Giles, C. L.; Bollacker, K. D.; and Lawrence, S. 1998. CiteSeer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, DL '98, 89–98. ACM.

Hajishirzi, H.; Yih, W.-t.; and Kolcz, A. 2010. Adaptive near-duplicate detection via similarity learning. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, 419–426. ACM.

Henzinger, M. 2006. Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, 284–291. ACM.

Herstein, I. N. 2006. *Topics in algebra*. John Wiley & Sons.

Manku, G. S.; Jain, A.; and Das Sarma, A. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, 141–150. ACM.

Sood, S., and Loguinov, D. 2011. Probabilistic near-duplicate detection using simhash. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, 1117–1126. ACM.

Williams, K., and Giles, C. L. 2013. Near duplicate detection in an academic digital library. In *Proceedings of the 2013 ACM Symposium on Document Engineering*, DocEng '13, 91–94. ACM.

Williams, K.; Wu, J.; and Giles, C. L. 2014. SimSeerX: A similar document search engine. In *Proceedings of the 2014 ACM Symposium on Document Engineering*, DocEng '14, 143–146. ACM.

Author Index