Context Aware Citation Recommendation System

Jun Hui Erh, M.Eng. CS '11
Sarah Nguyen, B.A. CS '13
Kyu-Young Kim, B.S. CS '12
Eric Gold, B.S. CS '14
{je96,smn64,kyk8,elg77}@cornell.edu

Abstract

This paper present a prototype machine learning method for generating citation recommendations within research papers, articles, and other technical documents. This is potentially useful for authors who include some statement of fact within their work and go back only later to provide appropriate citation for such content. To facilitate this, we phrase this problem as an instance of multi-class learning in which we map so-called 'local contexts' in a paper to possible documents that would best serve as a cited source. Within this paper, we look at the effectiveness of a Naive-Bayesian classifier that employs bag-of-words document descriptions. We encountered difficulties scaling this technique given the massive amounts of data available and necessary for proper classification. We closely analyze specific results over a subset of this data taken from $Citeseer^X$.

Keywords: Context, Extraction, Recommendation Systems

1 Introduction

While writing research papers, we often want to find good citations for various snippets of text in the paper. To do this, one must search relevant literature and find a small number of good citations. This is a very time-consuming task when the amount of literature present on the topic is very large or when one is new to the area of research. It would be very useful to have a citation recommendation system which could suggest a small number of good citations given a snippet of text which needs to be cited. Given the large number of research papers with examples of good citations, we seek to build a supervised learning algorithm which learns a model to suggest good citations given a context for citation.

2 Problem Definition

For a document to be cited (*target*), the text surrounding a citation is considered the *local context* of a citation. The text from the other parts of the document such as title, abstract

are considered the *global context* of the *source* paper. We used a Naive Bayes classifier to compute probabilities of cited papers based on the presence of certain words. The features for classification are words from the local context and the global context.

For each test document, we return a ranked list of k possible citations (class labels).

3 Related Work

Last semester, we studied a machine learning method based on Structural SVMs for generating citation recommendations. In order to deal with the large number of targets, we introduced Masks for each target class and showed the trade off between test accuracy and training time while choosing the mask sizes. This approach did not scale effectively.

Other approaches mostly deal with recommending a bibliography list for a manuscript or recommending papers to reviewers. They rely on a user profile or a partial list of citations. The most recent approach by He et al. [4] uses a context-aware approach for recommending citations. It uses the following methodology: A candidate set of papers are selected based on heuristics such as has an author in common, most similar abstract, most similar title, etc. These papers are then scored by measuring the similarity between the manuscript and the candidate. They propose a non-parametric probabilistic model to measure the context-based and overall relevance between the two documents. Tang and Zhang [2] uses a topic based recommendation approach for recommending citations. It does so by modelling the topic distributions of paper and their cited documents using a Restricted Boltzmann Machine (RBM) model. For each document to be cited it will model the topic representation of the paper and use a probability measure to compute how likely a paper be cited and return the top K papers with the largest probability. In recommending local context, they compute the KL-divergence of the local context with the recommended paper.

4 System Architecture and Implementation

We utilized the same training data as last semester with an additional 10,000 documents. Generating training data for our classifier is similar to the process from last semester. Raw data was taken from $Citeseer^X$ in the form of plain text representations of papers. While this had the effect of rendering unreadable most notation, charts, and diagrams, the actually body text of the papers remained more or less intact. Using the same processing tools employed by $Citeseer^X$ itself, we proceeded to extract citation information from our chosen data set. This consisted of local contexts surrounding the citation, as well as the titles, authors, and publication dates of the cited papers. This could be matched against $Citeseer^X$ metadata to determine unique identifiers for most cited papers in the dataset. Additionally, these tools also served useful in parsing apart the body text of input papers into its constituent sections, allowing us to use paper abstracts, section headers, authors, etc. as the basis of additional training features should it prove necessary.

In pre-processing our documents, we discard the in-text citations and reference sections for each paper. In-text citations and reference sections were removed to eliminate the possibility of a paper being recommended based on the name of an author and not on the actual content of the document. We tokenized words by space characters, only extracting words consisting of letters (after converting all uppercase letters to lowercase ones), dashes, underscores, and apostrophes. We also discarded the top 1000 words according to word count

from all documents, words with a length < 3, words with word count < 3. Then for each document, we constructed a word histogram of the global and local context.

Document frequency and word frequency (total number of times a given word appeared in the data set) were calculated with a script that analyzed the entire data set calculated frequencies for each word in the dataset.

All data generation was performed using a mix of Python and Java.

5 Methodology

Every in-text citation in a research paper can be considered as an example for the learning algorithm. For a given citation, the words surrounding a citation can be considered as the *local context* and words from other parts of the paper, for example: title, abstract, introduction, etc., can be considered as the *global context*. We consider each cited paper as a class label. Recommendations were generated by calculating the probability $P(y|d)$ that a local context in document d is in class y (in other words, local context in document d cites document y), and then choosing the k highest ones. This is done using Bayes Theorem with a Naive Bayes Classifer which assumes that words in a class appear independently of each other:

$$h_{text}(\vec{x}) = argmax_y\{P(Y = y)\prod_{i=1}^{l} P(W = w_i|Y = y)\}$$

where $\vec{x}$ is the vector of all the words and their frequencies in the document d.

The probability, with Laplacian smoothing, that a word w is used to cite a certain document y is found from the training data:

$$P(w|y) = \frac{1+TF(w,y)}{|F|+\sum_{w\in F} TF(w,y)}$$

where TF is the term freqency of a word in a particular document (overall number of times a word w appears in document y) and F is the size of the dictionary (number of different values w can take).

Training the classifier involves computing $P(w|y)$ for both global and local context. These are stored in an inverted index to save space. We also computed the IDF value for each word and $P(y)$ (probability of document y being cited).

To classify the local contexts of a document, we first obtain a candidate set of possible classes using the global context. We obtain this candidate set by considering only the classes that has global context training examples that contain at least one word with the top m words by tf-idf in the global context of the document being classified. This reduces computation time since a class without any training examples with words that overlaps with the global context of the document will most likely have a low $P(y|d)$. From there, we select the top l classes by $P(y|d)$ as possible citations for the local context. After obtaining a candidate, we run the Naive Bayes classifier using the local context of the test document and pick the top k classes amongst the l candidates.

6 Experimental Evaluation

6.1 Data

Similar to last semester, we used research papers from $CiteSeer^X$ to generate the training and test data sets. The data from $CiteSeer^x$ was resolved to an unique identifier by matching records from metadata obtained using OAI harvesters. Unfortunately, the OAI harvesters and associated identifier document-mappings have been compromised and we were not able use the the metadata to do a sanity check that the classes returned by our classifier is sensible.

The distribution of the number of citations each paper receives follows Zipf's law. In order to evaluate the performance of our approach on documents belonging to various regions of the distributions, we sub-sampled all the citations into three data sets. The most frequent citations consist of the top 1000 most frequently cited documents. For the mid-frequency cited documents, we sampled documents which were cited between 14-15 times. Rare documents consist of documents cited at least twice. A large number of our data comprises of documents which were only cited once, but we were unable to use them because there is no way to seperate them into test and training sets. Thus, the rarely cited documents are documents that are cited twice. The data sets are summarized in Table 1.

Data Set	# Classes	Train Data Size	Test Data Size
Rarely Cited	1000	1528	328
Medium Cited	1000	7623	1525
Most Cited	1000	25596	5119

Table 1: Data characteristics

6.2 Results

Our Naive Bayes classifier is trained using 5-fold cross validation on the data sets. We evaluate the performance of the classifier by changing the number of words used to obtain the candidate set, the size of the candidate set and ranking evaluation. We also did a brief analysis on the running time of our algorithm.

6.2.1 Effects of number of words to obtain candidate set

In evaluating the test set, we varied the number of words used from the global context to construct a candidate set. Figure 1 and Figure 2 shows the test error for the 3 different data sets and selecting top k classes using candidate set of size 1000 and 20 respectively.

From Figure 1, we observe that as the number of words used to construct the candidate set increases, the error rate increases eventually hitting a plateau. We hypothesize that choosing more words generates more possible classes and thus introduces more noise and the probability of choosing the correct citation decreases. This tells us that a small number of words is able to correctly classify documents while using more words doesn't increase our accuracy, possibly because of overfitting. The error rate remains relatively flat for all different data sets and different top k classes at about 50 words. This may be due to the fact that for each data set there are only 1000 possible classes, thus after 50 words, all possible classes are already included in the candidate set. Also, for each data set the

error rate decreases as we increases the the number of target documents returned by the algorithm. This is to be expected because the algorithm has more chances to choose the original citation.

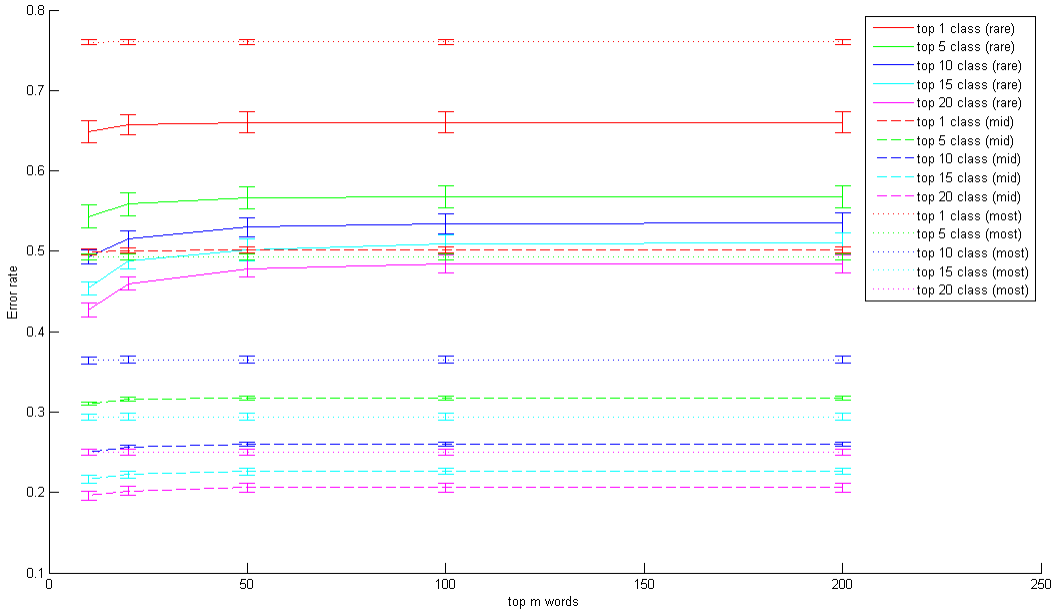

Figure 1: Test error for different data sets with varying number of words with candidate set of size 1000

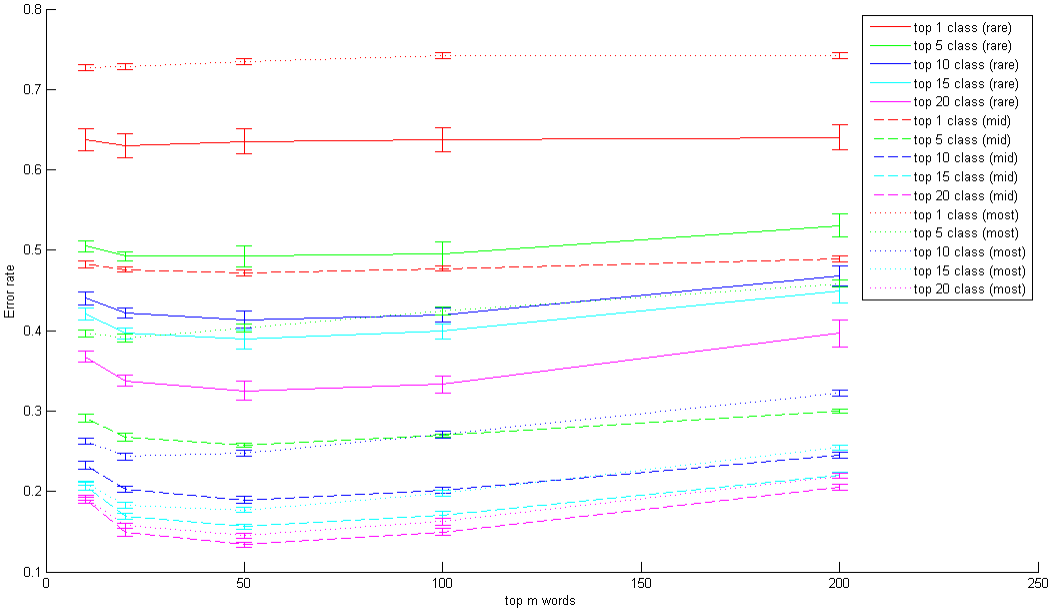

Figure 2: Test error for different data sets with varying number of words with candidate set of size 20

In Figure 2, we can see that for all cases, the optimal number of words between underfitting and overfitting is about 50. This contrasts with Figure 1 where we cannot observe this pattern. We believe this may be due to the increased noise introduced by a large candidate set size.

6.2.2 Effect of candidate set size

For measuring the effect of the size of the candidate set, we varied the size of the candidate set used to evaluate the local context. Figure 3 shows the test error for the 3 different data sets and selecting the top 20 scored classes. We see that the optimal candidate set size is between 30 - 100 before underfitting or overfitting occurs.

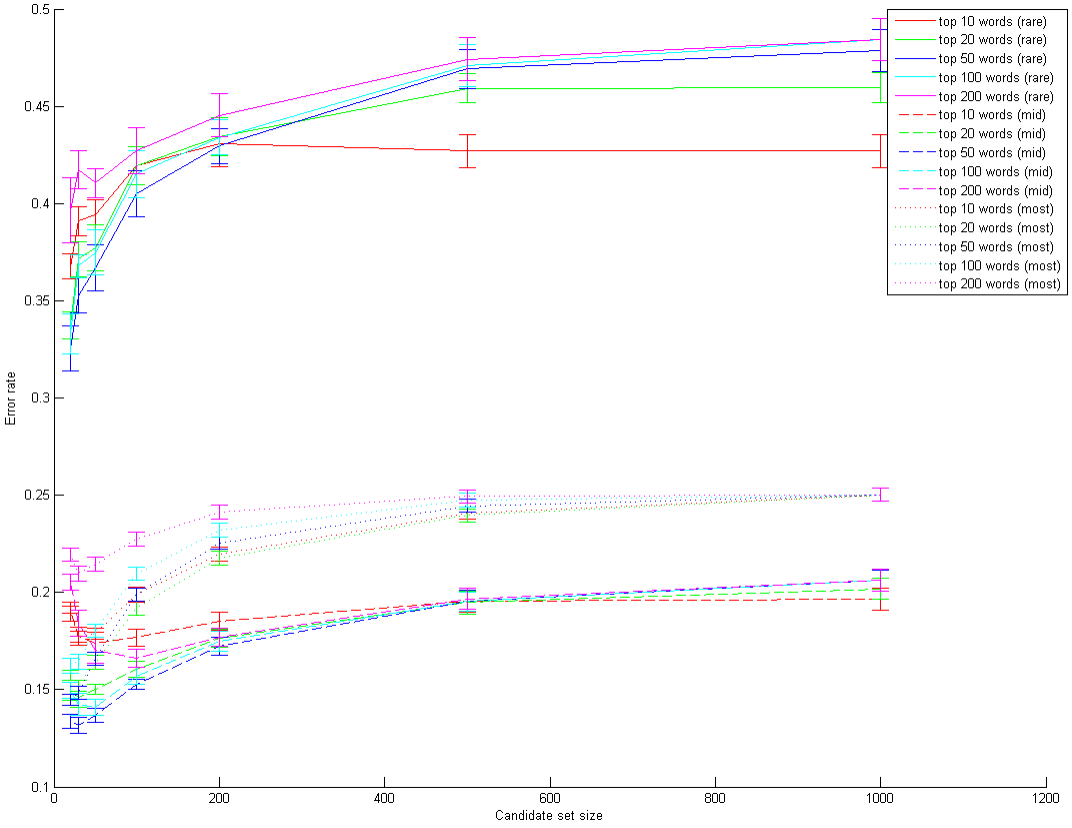

Figure 3: Test error when varying candidate set size

6.2.3 Ranking Evaluation

A recommendation system does not necessarily have to return the best recommendation since a user could select from a list of possible recommendations. Thus we return the top k scored classes. A document is considered correctly classified if the correct class is one of the top k scored classes. We summarize the test error in Table 2 below with a candidate set of size 1000 using 50 words with the corresponding plot in Figure 4.

k	Rarely Cited	Medium Cited	Most Cited
1	0.6604	0.5019	0.7601
5	0.5661	0.3177	0.4930
10	0.5301	0.2598	0.3648
15	0.5019	0.2260	0.2943
20	0.4786	0.2061	0.2502

Table 2: Test error rate of top k classes

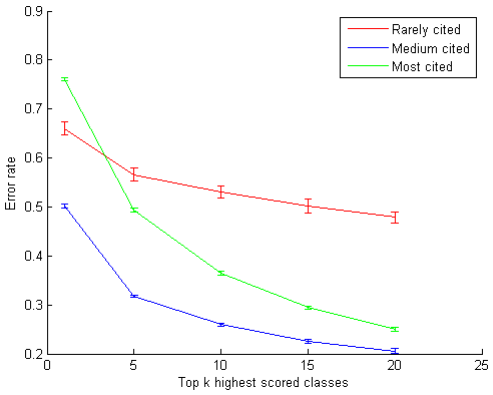

Figure 4: Graph of error rate for top k classes

As expected we see that the error drops as we increase the number of classes returned. An interesting observation to note is that the drop in error rate as we increase the number of classes diminishes as we increase k. This is more evident for rarely cited documents and medium cited documents. It will be interesting to see if the error rate eventually flattens out at a value $k < 1000$ (since each data set only has 1000 classes). However we did not run this experiment since in a practical system, the chances that a user selecting a possible citation diminishes with the rank of the classes. This is analogous to a search engine where the click through rate of a link returned by a query diminishes with the rank as returned by the search engine.

Another interesting observation to note is that increasing k improves the error rate for most-cited documents significantly, while inceasing k doesn't improves the error rate for rarely-cited documents as much except when k is small.

6.2.4 Effect of citation frequency

From Figure 4, we can clearly see that the error rate of the most cited data set is always higher than the error rate of the medium cited documents. Using a 1-tail z-test, we found with 95% confidence that the most frequently cited data set has a higher error rate than the medium cited data set. This is against our intuition since a more frequently cited class will have more training examples which by conventional wisdom typically leads to lower error rates. We collected data on the test error for all class labels in the medium and most frequently cited data sets. However, we were not able to observe any pattern between the error rates of a specific class with its citation frequency. This should be investigated further in the future.

6.3 Runtime Analysis

The program to classify the local context of a document can be break down into 3 sections:

- Global words: This is the portion that computes the top m words by tf-idf of the document. To compute the top m words we have to first compute the tf-idf score of all the words in the global context of the document. Thus, the run time is proportional to the number of words in the document.

- Candidate set: This is the section that uses the m words from the previous segment and obtain top l possible classes. We have to compute the $P(y|d)$ of all the classes that have at least one word in the m words. Since the more words we use generally increases the number of classes with overlapping words, the run time is proportional to m.

- Local context: This segment returns the top k classes using Naive Bayes Classifier on the local context of the document. To find the top k classes, we have to compute $P(y|d)$ using the local context of all possible classes. The run time is proportional to the number of words in the local context and the number of possible classes l.

We collected the running time of the 3 sections on the rarely-cited data set. Using this data set, we found that the time to compute tf-idf of all words in global context is 12.6ms per document with an average of 620 words. In Figure 5 below we summarize the running time of finding the candidate set when we increase the number of words. Figure 6 shows the running time to find the top k classes for a local context when the size of the candidate set varied.

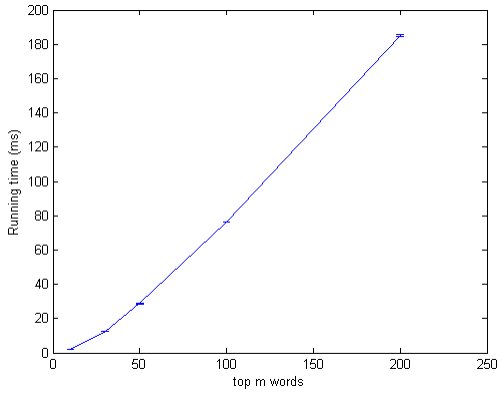

Figure 5: Running time to find candidate set of size 30 with varying number of words

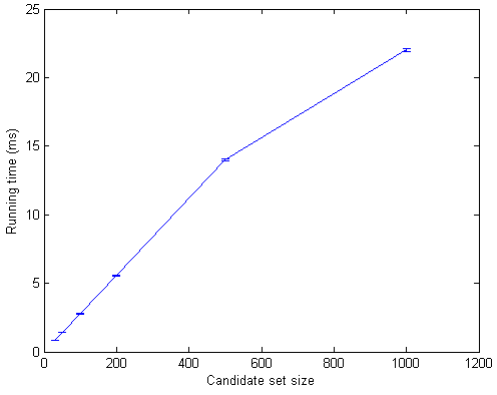

Figure 6: Running time to find top k classes of local context with varying candidate set size (candidate set found using top 30 words by tf-idf)

From Figure 5 and Figure 6 we see that the running time for the corresponding section increases as we increase number of global context words and candidate set size respectively. However we can see that the running time to find the candidate set is much larger and increases more rapidly than the running time to find top k classes of a local context. The running time to find the candidate set is also much larger than the time needed to compute tf-idf of all words in the global context. Thus, to classify the local context of a document, most of the time is spent on finding the candidate set.

7 Future Work

Ideally, the eventual result of this work would be a freely available online service. However, a great deal of preliminary work still needs to be done. Our current system relies on a simple bag-of-word approach and assumes independence between the words in a document. We could extend this work by using more complex models such as the Rocchio algorithm or an n-gram approach.

There might also be some utility found in looking at different ways of generating and evaluating our results. Our current assumption is that best set of results will be ones similar to those that would be found by a human, but this may not necessarily be the case. Humans are influenced heavily by social trends, and there is an argument to be made that the more recent and more frequently cited papers will continue to be cited more often (rich-get-richer phenomenon) because people are used to seeing them rather than for any inherent merit over the alternatives. Our results should not necessarily be held sway to these same trends just because those are what appear in the training data. Consequently, a beneficial line of future work might be to look at this as a diverse ranking problem, offering users a set of good matches that are also somewhat different from each other, rather than perhaps a number of very similar documents all from the same author or institution.

8 Conclusion

In this work we have presented a Naive Bayes classifier for creating citation recommendations. We have evaluated our approach on three different datasets representing three categories of citations, namely popular citations, regular citations and rare citations based on how frequently they are cited. We evaluated how our system performs when varying the word set size and the candidate set size.

Any such system based on this or other work will have to take great care in accounting for the scale of the data involved and the computation time needed to retrain as new data is added. It is possible that the approach used here could scale well to include more documents because this method could efficiently add new training documents since the majority of the data remains static. In generating possible citations for a document, a large portion of the running time is spent in generating the candidate set. However from our experiments, the optimal number of words needed to generate the candidate set is relatively small and thus the time needed to generate possible citations will not be too long.

9 Acknowledgments

- Thorsten Joachims, Cornell University: Advisor
- Hema Koppula, Cornell University: Consulting on past semester's project
- Katharina Morik, Technische Universitt Dortmund: Consulting and suggestions for future work
- Martin Berggren, Cornell University: Systems support

10 References

[1] Crammer, K. and Singer, Y. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res. 2* (Mar. 2002), 265-292.

[2] J. Tang and J. Zhang. A Discriminative Approach to Topic-Based Citation Recommendation. *PAKDD'09*.

[3] Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First international Conference on Machine Learning*(Banff, Alberta, Canada, July 04 - 08, 2004). ICML '04, vol. 69. ACM, New York, NY, 104.

[4] He, Q., Pei, J., Kifer, D., Mitra, P., and Giles, L. 2010. Context-aware citation recommendation. In *Proceedings of the 19th international Conference on World Wide Web* (Raleigh, North Carolina, USA, April 26 - 30, 2010). WWW '10. ACM, New York, NY, 421-430.