University of Houston at CL-SciSumm 2016: SVMs with tree kernels and Sentence Similarity*

Luis Moraes, Shahryar Baki, Rakesh Verma and Daniel Lee

Computer Science Department
University of Houston, TX 77204

Abstract. This paper describes the University of Houston team's efforts toward the problem of identifying reference spans in a reference document given sentences from other documents that cite the reference document. We investigated the following approaches: cosine similarity with multiple incremental modifications and SVMs with a tree kernel. Although the best performing approach in our experiments is quite simple, it is not the best under every metric used for comparison. We also present a brief analysis of the dataset which includes information on its sparsity and frequency of section titles.

1 Introduction

The CL-SciSumm 2016 shared task poses the problem of automatic summarization in the Computational Linguistics (CL) domain. Single text summarization is hardly new, however, in addition to the reference text to be summarized we are also given citances i.e. sentences that cite our reference text.

The shared task is broken into multiple tasks with the unifying theme of leveraging citances. Task 1a is, given a citance, to identify the span of reference text that best reflects what has been cited. Task 1b asks us to classify the cited aspect according to a predefined set of facets: hypothesis, aim, method, results, and implication. Finally, Task 2 is generating a structured summary.

We experimented with the following approaches: SVMs with a tree kernel and cosine similarity based on TF/IDF weights for sentences. The best results when measuring by sentence inclusion are obtained by cosine similarity. However, ROUGE-L scores are better for the tree kernel approach.

We also study two characteristics of the dataset: sparsity and section importance. We define section importance as the normalized frequency of the section, i.e., the number of correct reference sentences that belong to this section across all the citances. We found that the introduction is the most cited section in this year's dataset. We also find that citances are less sparse than the average sentence within the corpus.

* Research supported in part by NSF grants CNS 1319212, DGE 1433817 and DUE 1241772

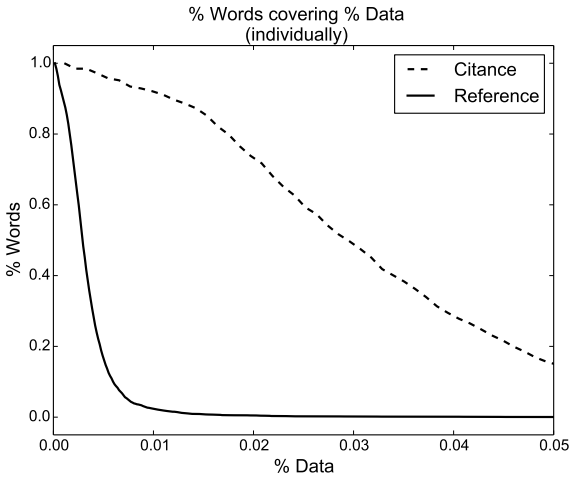

Fig. 1. The percentage of unique words in a set of sentences that appears in a percentage of all sentences. An indirect measure of sparsity.

2 Dataset

The dataset [3] consists of 30 total documents separated into three sets of 10 documents each: training, development, and test sets. For the following analysis no preprocessing has been done (for instance, stemming). There are 23784 unique words among the reference documents in the dataset. The citances contain 6415 unique words. The most frequent word among reference documents appears in 4125 sentences. The most frequent word among citances appears in 598 sentences. There are 6706 reference sentences and 913 citance sentences (a few annotations have more than one). The average reference sentence has approximately 22 words in this dataset whereas citances have an average of approximately 30 words.
In Figure 1 we can see how sparse the dataset is; the quicker the decay, the greater the sparsity. Noise in the dataset is one of the factors for the sparsity. We can see that citances, seen as a corpus, are in general less sparse than the reference texts. This can be an indication that citances have some common structure or semantics.

2.1 Citances

It should be noted that citances have a few peculiarities, such as an abundance of citation markers and proper names. Citation markers will sometimes include the names of authors, thus causing our sentences to have a greater frequency of proper names. Identifying and ignoring citation markers should reduce noise. This could justify the lesser sparsity (reoccuring authors), but could also justify greater sparsity (unique authors).

2.2 Frequency of Section Titles

For each cited reference sentence, we looked at the title of the section in which it appears. The titles that appeared with greatest frequency can be seen in Table 1. To extract these section titles we looked at the parent nodes of sentences within the XML document. The "title" and "abstract" sections are special since they refer to parent nodes of type other than SECTION. These results clearly show sentences that are cited are not uniformly distributed within a document.

Title	Frequency Relevant
introduction	25.55%
abstract	6.98%
title	5.58%
conclusion.	4.46%
the approach.	4.05%
potential for improvement.	3.35%
evaluation.	3.21%

Table 1. The frequency refers to how many times the section contained a relevant sentence. This was calculated from all the documents in the dataset.

3 Task 1a

In this task we are asked to identify the reference sentences referred to by a given citance. We approach the problem from two different perspectives. One of our systems follows the intuitive idea that the citance and the sentences to which it refers must share some similarity. Thus, we modify this system with the intent to capture more forms of similarity. Our second system abstracts further, instead of looking at the similarity between citance and reference sentences we look at the similarity between citance and reference sentence pairs, (c, r). We attempt to learn how to distinguish relevant and irrelevant pairs. We modify this system with the intent to improve the learned classifier.

3.1 TF/IDF Approach

This approach compares the TF/IDF vectors of the citance and the sentences in the reference document. Each reference sentence is assigned a score according to the cosine similarity between itself and the citance. Stopwords were removed for all configurations. A few of the modifications to this general approach follow:

Stemming. (ST) To remove the effect of using words in their different forms we used stemming to reduce words to their root form. For this purpose, we use the Snowball Stemmer, provided by the NLTK package [1].

Context Expansion. (CE) Citances by themselves only have a limited amount of semantic information. We expand the semantic content of a citance by including the sentences that occur directly before and after the citance when constructing its word vector.

WordNet Expansion. (WN) Another method for semantic expansion is to include the lemmas obtained from the synsets of each word in a sentence. This applies to both citances and sentences in the reference document. We use the Lesk algorithm, provided through the NLTK package [1], to perform wordsense disambiguation. This is a necessary step before obtaining the synset of a word from WordNet [7]. The lemmas for each synset are added to the word vector of the sentence; this augmented vector is used when calculating the cosine similarity instead.

Sentence Limiting. (SL) In order to reduce noise (due to OCR), we eliminate from consideration all sentences outside a certain range of number of words. For the TF/IDF approach we only consider reference sentences with more than 10 and less than 70 words. This process eliminates 1494 sentences out of 6706 total, roughly 22%. The test set has a total of 390 sentences that should be retrieved. Only 340 can still be retrieved once we restrict the number of words.

3.2 Tree Kernel Approach

We classify the (c, r) pairs of citance and reference sentence as relevant or irrelevant. We then rank the sentences according to how confident we are it is relevant.

SVM. The Support Vector Machine (SVM) is a classifier that learns a hyperplane to separate two classes [2]. The hyperplane with the largest margin is considered the best choice. Thus, learning becomes an optimization problem.
After training, an SVM will have a set of support vectors that define its hyperplane. The hyperplane can be thought of as a description of the boundary between classes. The output is the signed distance from the instance to this hyperplane: a positive distance implies the item belongs to the positive class; a negative distance implies the item is in the negative class.
Learning can be carried out with just the Gram Matrix, a matrix of the "distances" between pairs of data items. This is known as the kernel trick. By choosing a different kernel we implicitly transform the space where items reside. This broadens the utility of SVMs since the new space may allow for simpler separation between classes.

Subset Tree Kernel. Our kernel of choice is a convolutional kernel on trees. Although there are a few different formulations, we use the subset tree kernel which compares the number of common subset trees (subtrees that can have nonterminals as leaves) [9].

Parse trees may be ideally suited for the kernel, but we can easily produce "flat trees" for vector-like data. Flat trees can be understood as trees where all leaves are directly connected to the root. Therefore, in addition to the parse tree, we use a flat tree for the bag-of-words representation and the POS tags. For each sentence we have 3 different trees (similar to [9]), so for each (c, r) pair we have 6 trees total. The similarity between items is calculated by first finding the similarity between respective trees (using the subset tree kernel), then adding up each tree's contribution.

Class Imbalance. The way we modeled the problem makes it heavily imbalanced. The number of (c, r) pairs that are irrelevant is much larger than the number that are relevant. Due to this imbalance the classifier was trained with subsets of all possible pairs to reduce the influence of the majority class. We train SVMs with a random selection of negative items. We experimented with three different positive-to-negative ratios: $1:1$, $1:4$, $1:8$. We performed 5 runs for each configuration since the selection of negative items could help or hurt our performance.

Context Expansion. (CE) For the Tree kernel approach, context expansion consists in adding the sentence above and below a citance as part of the citance. This is only used in training; preliminary results showed little difference in performance when classifying more (c, r) pairs (those of the extra sentences).

Sentence Limiting. (SL) Similar to how we eliminated sentences from consideration in the TF/IDF approach, we do the same here. Since we randomly select negative items, it becomes of greater concern that we perform multiple runs. To get a sufficient number of runs we use a narrower range to reduce classification time; sentences must have more than 15 and less than 35 words. This process eliminates 3462 sentences out of 6706 total, roughly 51%. Out of the 390 relevant sentences in the test set, only 215 meet the criteria.

Implementation. We utilize SVM-LIGHT-TK [8, 4] for our experiments. We use the Stanford CoreNLP [6] for obtaining POS tags and parse trees from our data. First we train our SVM on all the relevant (c, r) pairs with a few select irrelevant pairs. Then, we classify all possible pairs in the reference text. The top 3 are selected as the output of our system.

4 Task 2

Task 2 consists in generating a summary for the reference text. For Task 2 we average the sentence scores given by each citance and extract sentences until we reach the 250 word limit. This was only performed for the Tree kernel approach and only against human summaries. Furthermore, we only evaluated the best performing run of each configuration (according to its performance in Task 1a).

5 Evaluation

The evaluation of our systems is done by comparison of several metrics which are detailed below. For the Tree kernel method, these values are averaged over 5 runs.

ROUGE-L. The ROUGE metrics are useful for evaluating summaries. In particular we look at the ROUGE-L metric [5], which has the fewest parameters. The ROUGE-L metric is based on the Longest Common Subsequence (LCS). Consequently, it is more lenient since sentences that share words will be considered as somewhat correct.

Top-3 Metrics. Our systems output the top 3 sentences, thus we compute recall, precision, and F_1 score for these sentences. If a relevant sentence appears in the top-3, then it factors into recall, precision, and F_1 score. Note that due to sentence limiting (SL) we impose a limit on the F_1 score attainable. Since we always return the top 3 sentences, for the 279 citances of the test set we return 837 total sentences. We can calculate the maximum attainable F_1 score. For the Tree kernel approach it is 35.04%. For the TF/IDF aproach it varies between 44.98% and 58.02%.

Mean Average Rank. We compute the average rank by obtaining the rank of all the relevant reference sentences of each citance. These are normalized according to the total number of sentences being considered. The normalized rank is a value within the interval $[0, 1]$. We average these ranks among the citances for a document. Finally we find the mean of these averages for all documents. For example, for a single reference text with two citances, each referring to a single sentence, we would average the normalized rank of these two sentences according to their appropriate citance. We then find the mean among the documents processed. Lower is better.

Method	ROUGE-L
TKern(1:1)+SL	58.78%
TKern(1:4)+SL	57.90%
TKern(1:8)+SL	57.76%
TKern(1:1)+SL+CE	**58.84%**
TKern(1:4)+SL+CE	58.12%
TKern(1:8)+SL+CE	57.87%

Table 2. F_1-score of ROUGE-L metric for Tree kernel approach.

Method	ROUGE-L	Method	ROUGE-L
TFIDF	**50.63%**	TFIDF+CE	48.05%
TFIDF+ST	50.35%	TFIDF+ST+CE	48.37%
TFIDF+SL	49.45%	TFIDF+SL+CE	47.48%
TFIDF+WN	46.61%	TFIDF+WN+CE	45.87%
TFIDF+ST+SL	49.10%	TFIDF+ST+SL+CE	47.47%
TFIDF+ST+WN	37.93%	TFIDF+ST+WN+CE	36.75%
TFIDF+SL+WN	45.85%	TFIDF+SL+WN+CE	45.40%
TFIDF+ST+SL+WN	38.29%	TFIDF+ST+SL+WN+CE	37.25%

Table 3. F_1-score of ROUGE-L metric for TF/IDF approach.

Method	P@3	R@3	F_1	Mean Avg.Rank
TKern(1:1)+SL	5.63%	12.10%	7.69%	0.262 (215/390)
TKern(1:4)+SL	4.85%	10.41%	6.61%	0.263 (215/390)
TKern(1:8)+SL	5.13%	11.02%	7.00%	0.245 (215/390)
TKern(1:1)+SL+CE	5.71%	12.25%	7.79%	0.261 (215/390)
TKern(1:4)+SL+CE	4.70%	10.10%	6.42%	0.262 (215/390)
TKern(1:8)+SL+CE	5.11%	10.97%	6.97%	0.252 (215/390)
TFIDF	7.88%	16.92%	10.75%	0.106 (298/390)
TFIDF+ST	8.60%	18.46%	11.73%	0.124 (329/390)
TFIDF+SL	8.72%	18.71%	11.89%	0.093 (276/390)
TFIDF+WN	4.77%	10.25%	6.51%	0.154 (331/390)
TFIDF+ST+SL	**8.96%**	**19.23%**	**12.22%**	0.112 (309/390)
TFIDF+ST+WN	5.61%	12.05%	7.66%	0.164 (336/390)
TFIDF+SL+WN	6.09%	13.07%	8.31%	0.137 (307/390)
TFIDF+ST+SL+WN	5.61%	12.05%	7.66%	0.144 (286/390)
TFIDF+CE	7.04%	15.12%	9.61%	0.159 (330/390)
TFIDF+ST+CE	7.04%	15.12%	9.61%	0.167 (349/390)
TFIDF+SL+CE	7.88%	16.92%	10.75%	0.137 (303/390)
TFIDF+WN+CE	4.65%	10.00%	6.35%	0.216 (354/390)
TFIDF+ST+SL+CE	7.76%	16.66%	10.59%	0.149 (325/390)
TFIDF+ST+WN+CE	4.89%	10.51%	6.68%	0.220 (**356**/390)
TFIDF+SL+WN+CE	5.25%	11.28%	7.17%	0.183 (326/390)
TFIDF+ST+SL+WN+CE	4.77%	10.25%	6.51%	0.189 (304/390)

Table 4. Recall, precision, and F_1 score at Top-3. Average rank of relevant sentences. In parentheses we have the number of relevant sentences with non-zero similarity.

Method	ROUGE-L
TKern(1:1)+SL	27.56%
TKern(1:4)+SL	26.05%
TKern(1:8)+SL	**27.68%**
TKern(1:1)+SL+CE	27.09%
TKern(1:4)+SL+CE	26.52%
TKern(1:8)+SL+CE	25.64%

Table 5. F_1-score of ROUGE-L metric for Task 2.

6 Discussion

6.1 TF/IDF Results

The TF/IDF approach is unexpectedly our best performing approach. Although some modifications hurt performance, upon closer inspection we see how they might improve our results in other ways. It is, however, surprising that the minimally modified TFIDF+SL+ST approach has the highest F_1 score. Furthermore, among the TF/IDF configurations, the unmodified TF/IDF approach has the highest ROUGE-L F_1 score.
One of the considerations when using cosine similarity is whether or not we get any value at all due to sparsity. As we can see from Table 4, a portion of relevant sentences are indistinguishable since they have zero similarity. WordNet expansion, stemming, and context expansion provide a significant increase in non-zero similarities among relevant sentences. However, recall and precision decrease in turn. Sentence limits have the opposite effect: they increase recall and precision but also decrease the number of relevant sentences we can distinguish.

6.2 Tree Kernel Results

For the Tree kernel approach, we expected the inclusion of more negative items to increase our recall and precision. However, the configuration with 1 : 1 ratio had the best performance. It is significant that this dominance occurs not only with regards to the ROUGE-L scores in Table 2 but also in terms of top-3 metrics in Table 4.
Average rank did improve with a greater number of negative items. We conjecture these negative items in training had the effect of lowering the rank of negative items in testing, thus improving the rank of positive items. This overall improvement came at the expense of recall and precision.
The performance variation between runs was calculated for each system as the maximum F_1 at top-3 score attained minus the minimum F_1 at top-3 score attained in these 5 runs. The system with the largest variation was Tkern(1:1)+SL, with 2.7% difference, whereas the system with the least variation was TKern(1:1)+SL+CE, with 0.9% difference.
The output had a curious behavior. For each reference text, the sentences chosen by the system were the same regardless of citance. Even after context expansion, the behavior persisted. Unfortunately, this did not translate into high ROUGE-L scores for Task 2. It is possible that tuning the method so this behavior does not occur would increase its performance.
Finally, it is interesting to note that the ROUGE-L scores for the Tree kernel approach were consistently higher than the scores for the TF/IDF approach.

7 Future Work

We investigated the effects of various modifications on the performance of a simple TF/IDF approach. In addition, an SVM with a tree kernel was also employed with mixed results. Although the traditional F_1

score points to one of the simplest approaches as the most effective, it is important to remember the two approaches tackled the same problem from different perspectives. Whether these two perspectives – similarity between citance and reference sentence and the similarity between (c, r) pairs – complement each other is worth exploring.
The characterization of citances also warrants further research. The peculiarities apparent (and those less apparent) could lead to improvements in these tasks.

References

1. Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.
2. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
3. Kokil Jaidka, Muthu Kumar Chandrasekaran, Sajal Rustagi, and Min-Yen Kan (2016). Overview of the 2nd Computational Linguistics Scientific Document Summarization Shared Task (CL-SciSumm 2016). In *Proceedings of the Joint Workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL 2016)*, Newark, New Jersey, USA.
4. Thorsten Joachims. *Advances in Kernel Methods*, chapter Making Large-scale Support Vector Machine Learning Practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.
5. Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 605. Association for Computational Linguistics, 2004.
6. Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.
7. George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.
8. Alessandro Moschitti. Making tree kernels practical for natural language learning. In *EACL 2006, 11st Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, April 3-7, 2006, Trento, Italy*, pages 113–120. The Association for Computer Linguistics, 2006.
9. Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. Exploiting syntactic and shallow semantic kernels for question answer classification. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. The Association for Computational Linguistics, 2007.