Klick Labs at CL-SciSumm 2018

Gaurav Baruah and Maheedhar Kolla

Klick Inc., 175 Bloor Street East, Toronto, Ontario M4W 3R8, Canada
gbaruah@klick.com, mkolla@klick.com

Abstract. The CL-SciSumm 2018 shared task is designed to further the state-of-the-art in constructing summaries of research papers. Participating systems extract focused information from reference papers given their citing papers in order to construct a summary. To that end, given a citing sentence, we would need to know what concept (a result, method, or inference) is the citing sentence referring to in the reference paper. We explore the efficacy of word embedding based similarity functions to find relevant sentences from reference papers that are likely being referred to by citing sentences (citances). We also classify the sentences into appropriate discourse facets. Our preliminary findings indicate rank optimization for returning top sentences is useful for this task, and using term-IDF weights for averaging term embeddings improves performance.

Keywords: word embeddings · sentence similarity · sentence classification

1 Introduction

An appropriate (and adequate) summary of a specific line of research, could greatly benefit researchers: salient findings can be quickly collated, information overload can be avoided, contrasting viewpoints can be examined, all of which help in a better understanding of the state-of-the-art of the research topic, while getting an overview of its historical development.

An obvious first step in such an endeavor is to generate an adequate summary of *one* research publication. The CL-SciSumm shared tasks [5, 6, 4] enable researchers to explore solutions for the constructing a structured summary of a research paper given its citation contexts. This is interesting because, a set of citations (from various citing papers) provides varying viewpoints about the knowledge in the referred research paper. To better understand the relationship between the reference and citing papers, it is necessary to find sentences in the reference paper, that describe concepts which are the subject of the citances (the sentences that refer to the reference paper) in the citing papers.

In this paper, we explore various baseline algorithms (with some variations), for ranking the sentences of the reference paper, in order of their similarity to a given citance. We experiment with BM25 over sentences, cosine distance between sentence embeddings (constructed by averaging the word embeddings of the sentence), weighted averages for embeddings, as well as, rank optimization,

and variations in word embeddings construction. Our preliminary findings show that:

- BM25 is a strong baseline for this task.
- Inverse Document Frequency weighted averaging of word embeddings improves performance.
- Optimizing for number of top sentences to return from ranked reference paper sentences improves performance.

1.1 Challenge Data and Evaluation Metrics

The task organizers shared "topics", each consisting of a reference paper, and at least 10 citing papers. Within each citing paper, sentences that referred to the reference paper (i.e. *citances*) were manually annotated. Furthermore, sentence(s) in the reference paper that are most likely the subject of discourse of each citance were also manually annotated. The type of discourse (aim, hypothesis, method, implication, or result) was also noted. Gold standard summaries of the reference papers were also shared as part of the training set.

There are 3 subtasks in CL-SciSumm shared task:

- *Task1a*: find sentences in the reference paper that are referred to by a citance,
- *Task1b*: classify the found sentences into appropriate discourse facets.
- *Task2*: construct a summary of the reference paper using the found sentences (and possibly other related content in the topic's files)

The evaluation metrics for Task1 are Precision, Recall, and F1 (the harmonic mean of Precision and Recall). Task2 is evaluated using the ROUGE [10] family of metrics. For this iteration of the CL-SciSumm shared task, we only participate in Task1.

2 Data Preprocessing

We downloaded the `Parcit Structured XML`[1] [3] version of the ACL Anthology Reference Corpus [2]. We extracted the all the text from the XML format (except for `figure, equation, table` tags). Using the corpus text we constructed:

- a term to IDF (inverse document frequency) lookup table for all terms in the corpus; we utilized the scikit-learn's [12] "`TfidfVectorizer`" (with default parameters) for computing term-IDF values.
- word2vec embeddings [11] for terms; (with parameters: context window 8, number of iterations 20, negative samples 10, number of dimensions 200)
- word2vec embeddings with normalized text (no punctuation, all lowercase); (with parameters: context window 5, number of iterations 25, negative samples 15, number of dimensions 200)

[1] https://acl-arc.comp.nus.edu.sg/archives/acl-arc-160301-parscit

3 Task1a Methods and Experiments

We approach Task1a as a sentence similarity problem: if the citance text is the "query" sentence, we need to find the sentences from the reference paper that are *most similar* to the query sentence. The similarity function could potentially incorporate various lexical or semantic features, and the goal then is to find a sentence similarity function that performs best for this subtask. As the similarity function imposes an ordering over the sentences in the reference paper, we also try to optimize the number of most similar sentences to be returned.

3.1 Baseline Systems

BM25 (Vanilla) baseline (`BM25`) We first build a baseline based on the BM25 similarity [14] of two sentences. We utilized corpus wide IDF-weights for terms, and restricted sentence term frequency to 1.

BM25 (No Author) baseline (`BM25_noauth`) We noticed that references of the form "(author, year)" caused those sentences to be returned (from the reference papers) that had the "author's" names present. Such sentences are typically references to an even earlier paper by the same author(s). We therefore generated a variation on the baseline wherein we removed "(author, year)" tokens from the query citance text.

3.2 Word Embedding based similarity

We wanted to experiment with word embedding based approaches for sentence similarity. Typically a sentence or phrase embeddings [9] could be first constructed and then the sentence-embeddings of citances and reference paper sentences could be compared. Alternatively, we can compute an average vector of the word embeddings for a sentence which could help approximate the concept represented by the sentence. For this work, we follow the latter process, and we try 3 different variations in computing an average vector.

Average Embedding based similarity (`w2v_avg`) Our basic word embedding based similarity function:

- computes the average vector of the query citance's term embeddings giving us a query vector,
- computes the average vector for each sentence in the reference paper giving us sentence vectors,
- computes the cosine distance between the query vector and the sentence vectors, and
- returns a ranked ordering over sentence vectors based on their cosine distance from the query vector.

IDF-weighted Average Embedding based similarity (`w2v_idf_avg`) In this method, instead of using the regular average, we compute a weighted mean of the term embeddings. The weights are the IDF values for the terms in the sentence/citance; IDF of terms is computed using the ACL Anthology Reference Corpus (Section 2).

Smooth Inverse Frequency based similarity (`w2v_sif`) We also investigate the efficacy of the Smooth Inverse Frequency [1] based method for textual similarity, wherein:

- We first compute a weighted average for a sentence/query term embeddings, with the weight $a/(a+p(w))$ where $a = 0.001$, and $p(w)$ is the probability of occurrence of term w in the ACL Anthology Reference corpus (Section 2).
- We consider the set of vectors constituting of the query vector and the reference paper's sentence vectors; we compute the first principal component[2] [7] of the set of vectors; and then, we remove (subtract) the principal component, from each of the individual query-citance/sentence vectors in the set. This transformation aims to remove commonalities between the averaged embeddings [1].
- We return a ranked list of sentence vectors based on the cosine distance between the transformed query vector and the transformed sentence vectors.

3.3 Variations on the core methods

We also explore 3 different variations that build upon some of the methods described above.

Rank Optimization (`optRank`) Given that sometimes more than one sentence can be referred to from the citing paper, and that sometimes the similarity function returns the referred sentence lower in the ranked list, we can try to optimize the number of top documents to return for maximizing the F1 metric. We perform a 5-fold cross validation over the training data to determine how many top ranked sentences need to be returned on average, in order to maximize F1. We computed the mean number of top ranked sentences to be returned for each of the similarity method described in Section 3.

"General" vs. "Normalized" Embeddings We observed that `word2vec` program[3] returns a different vocabulary set and embeddings with differently processed inputs. For instance, given raw document text, word2vec produces different embeddings for "methodically", "Methodically", "Methodically," and "methodically,". On the other hand, given lower-cased document text with all punctuations removed (`normEmb`), produces only one embedding for "methodically". We tried to see how (if at all) this difference in embeddings affects system performance .

[2] https://en.wikipedia.org/wiki/Principal_component_analysis
[3] https://github.com/tmikolov/word2vec

Averaging Embeddings over a Window (`_wndw_avg`) Given that the average length of sentences in the challenge data is 23 words, computing an average over 23 term embeddings may result in an inadequate sentence vector representation. We could try to capture phrase level concepts and use them for similarity computation. It is possible, that some phrase-level average vectors that occur early in one sentence, are most similar to phrases that occur later in another sentence. In this method:

- We compute average vectors for an overlapping window of 5 terms, e.g. a sentence of 6 words would return a set of 2 averaged vectors. Thus we get a set of average vectors A_q for the query citance, and another set A_s for a sentence in the reference paper.
- We generate a matrix D_{qs} of size $|A_q| \times |A_s|$ with each element being the pairwise cosine distance between the elements of A_q and A_s.
- We find the maximum similarity along each row of D_{qs} and compute a mean of the row maximums (R_{max_mean}). Similarly, we compute the mean of the column maximums (C_{max_mean}).
- The similarity function returns (R_{max_mean}+C_{max_mean})/2 for each sentence in the reference paper.

The idea here is to capture the maximum possible similarity between the phrases (windows) of two sentences.

4 Task1b Methods and Experiments

Task 1b involves identifying the *Discourse Facet*, of the cited text from each reference paper, from a pre-defined set of labels: *Aim*, *Hypothesis*, *Implication*, *Method*, *Result*. We approached this task as a multi-class classification problem where a trained classifier is used to predict the discourse facet label. For each cited text sentence, we construct its corresponding feature vector , which is an average of word2vec embedding vectors for each term in that particular sentence. We used our word2vec embeddings obtained from ACL Anthology Reference Corpus (as explained in Section 2) to construct our sentence feature vectors. We then experimentally compared three classifiers: KNN classifier, RandomForestClassifier, SVM classifier as implemented in scikit-learn [12], using average sentence word2vec embedding vectors as input to predict the discourse facet label.

5 Results

5.1 Task1a: Performance over the training set 2018

Table 1 lists the performance of methods (and respective variations) described in the previous sections. From the table, we can see that:

1. The BM25 baseline is hard to beat, when comparing against basic word embedding average based similarity functions.

Table 1. Task1a Precision, Recall, (F1) of methods/variations over Training Set 2018. **Bold** and *italics* values denote best performance in columns and rows respectively.

Method	Performance	with rank optimization (optRank)	with normalized embeddings (normEmb)	optRank and normEmb
BM25	0.132, 0.086, (***0.104***)	0.132, 0.086, (0.104)	NA	NA
BM25_noauth	0.118, 0.076, (0.092)	0.097, 0.125, (***0.109***)	NA	NA
w2v_avg	0.061, 0.039, (0.048)	0.049, 0.064, (0.056)	0.081, 0.052, (0.064)	0.061, 0.118, (*0.080*)
w2v_idf_avg	0.093, 0.060, (0.073)	0.093, 0.060, (0.073)	0.096, 0.062, (0.075)	0.058, 0.149, (*0.083*)
w2v_sif	0.095, 0.061, (0.074)	0.066, 0.128, (*0.087*)	0.084, 0.054, (0.066)	0.064, 0.124, (0.084)
w2v_wndw_avg	0.078, 0.051, (0.062)	0.056, 0.108, (0.078)	0.105, 0.068, (0.083)	0.082, 0.107, (***0.093***)
w2v_idf_wndw_avg	0.097, 0.063, (0.076)	0.085, 0.109, (*0.095*)	0.108, 0.070, (**0.085**)	0.070, 0.135, (0.092)

2. Optimizing number of top sentences to return almost always results in improvement in performance as measured by F1.
3. Using normalized embeddings improved performance over not using them, except in the case of smooth inverse frequency based similarity.
4. Using IDF weighted average vectors worked better than simple averaging of sentence word embeddings.
5. Window based averaging worked better than entire sentence averaging.

5.2 Task1b

Table 2. Task 1b training data: Discourse Facet breakdown

Discourse facet	number of sentences
Aim	28
Hypothesis	12
Implication	51
Method	365
Result	61

For our official submission, we experimented with the following classifiers:

- K-Nearest Neighbors Classifier
- RandomForest Classifier
- SVM Classifier

Using 2018 training data, we ran 5-fold stratified K-fold validation to evaluate and select the classifier that gives us better performance. As observed in Table 2, majority of discourse facet instances in current training data are labeled *method citation*. Evaluation results for our stratified K-fold validation are listed in Table 3.

Table 3. Task 1b Training cross validation results

Classifier	Mean Accuracy	Mean Precision	Mean Recall	Mean F1
KNN (Num Neighbors = 5)	0.66	0.64	0.66	0.62
RandomForestClassifier (n_estimators=50)	0.725	0.64	0.715	0.63
SVM classifier (Kernel='RBF')	0.719	0.6	0.719	0.61

5.3 Submitted Runs for Task1

Based on the performance of our methods on the Training Set for 2018, we submit as runs the system outputs of: (i) `BM25` + `optRank`, (ii) `BM25_noauth` + `optRank`, (iii) `w2v_idf_avg` + `optRank` + `normEmb`, (iv) `w2v_sif` + `optRank`, (v) `w2v_wndw_avg` + `optRank` + `normEmb`, (vi) `w2v_idf_wndw_avg` + `optRank` + `normEmb`. For each of these methods we predicted the discourse facet using the Random Forest Classifier.

6 Conclusions and Future Work

We submitted 6 runs for Task1a of the CL-SciSumm 2018 shared task. We found that BM25 forms a good baseline for this task, and IDF weights of terms can boost effectiveness of word embedding averaging based methods for sentence similarity. Optimizing number of sentences to return for Task1a also improves performance over the F1 metric.

In the future, for discourse facet prediction, we wish to combine word embedding features along with word or character overlap segments between the reference text and title text to generalize and improve prediction. In particular for Task1a, as more data becomes available, it could be promising to use learning to rank methods [8, 15] or answer selection based methods [16, 13], for finding reference paper sentences similar to citances.

References

1. Arora, S., Liang, Y., Ma, T.: A simple but tough-to-beat baseline for sentence embeddings (2017), https://openreview.net/pdf?id=SyK00v5xx
2. Bird, S., Dale, R., Dorr, B.J., Gibson, B., Joseph, M.T., Kan, M.Y., Lee, D., Powley, B., Radev, D.R., Tan, Y.F.: The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics (2008)
3. Councill, I.G., Giles, C.L., Kan, M.Y.: Parscit: an open-source CRF reference string parsing package. In: LREC. vol. 8, pp. 661–667 (2008)
4. Jaidka, K., Chandrasekaran, M.K., Jain, D., Kan, M.Y.: The CL-SciSumm shared task 2017: results and key insights. In: Proceedings of the Computational Linguistics Scientific Summarization Shared Task (CL-SciSumm 2017), organized as a part of the 2nd Joint Workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL 2017) (2017)
5. Jaidka, K., Chandrasekaran, M.K., Rustagi, S., Kan, M.Y.: Overview of the CL-SciSumm 2016 shared task. In: Proceedings of Joint Workshop on Bibliometric-enhanced Information Retrieval and NLP for Digital Libraries (BIRNDL 2016) (2016)

6. Jaidka, K., Chandrasekaran, M.K., Rustagi, S., Kan, M.Y.: Insights from cl-scisumm 2016: the faceted scientific document summarization shared task. International Journal on Digital Libraries pp. 1–9 (2017)
7. Jolliffe, I.: Principal component analysis. In: International encyclopedia of statistical science, pp. 1094–1096. Springer (2011)
8. Lauscher, A., Glavaš, G., Eckert, K.: University of mannheim@ CLSciSumm-17: Citation-based summarization of scientific articles using semantic textual similarity (2017)
9. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning. pp. 1188–1196 (2014)
10. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out (2004)
11. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
12. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
13. Rao, J., He, H., Lin, J.: Noise-contrastive estimation for answer selection with deep neural networks. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. pp. 1913–1916. ACM (2016)
14. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: Bm25 and beyond. Found. Trends Inf. Retr. **3**(4), 333–389 (Apr 2009). https://doi.org/10.1561/1500000019, http://dx.doi.org/10.1561/1500000019
15. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 373–382. ACM (2015)
16. Yu, L., Hermann, K.M., Blunsom, P., Pulman, S.: Deep learning for answer sentence selection. arXiv preprint arXiv:1412.1632 (2014)