Reference Scope Identification for Citances Using Convolutional Neural Network

Saurav Jha
MNNIT Allahabad, India
mail@sauravjha.com.np

Aanchal Chaurasia
NIT Rourkela, India
aanchal21194@gmail.com

Akhilesh Sudhakar
IIT (BHU), Varanasi, India
akhileshs.s4@gmail.com

Anil Kumar Singh
IIT (BHU), Varanasi, India
nlprnd@gmail.com

Abstract

In the task of summarization of a scientific paper, a lot of information stands to be gained about a reference paper, from the papers that cite it. Automatically generating the reference scope (the span of cited text) in a reference paper, corresponding to citances (sentences in the citing papers that cite it) has great significance in preparing a structured summary of the reference paper. We treat this task as a binary classification problem, by extracting feature vectors from pairs of citances and reference sentences. These features are lexical, corpus-based, surface and knowledge-based. We extend the current feature set employed for reference-citance pair identification in the current state-of-the-art system. Using these features, we present a novel classification approach for this task, that employs a deep Convolutional Neural Network along with two boosting ensemble algorithms. We outperform the existing state-of-the-art for distinguishing between cited spans and non-cited spans of text in the reference paper.

1 Introduction

Citation sentences or 'citances' that cite a reference paper (RP) can give valuable information about the larger context in which the RP is written, key ideas behind the RP and a concise synopsis of it. All of this is important for a task like scientific paper summarization, which not only requires the content of a paper but also meta-information about it. This kind of information would otherwise have to be obtained from sources such as literature reviews and surveys about the paper, which in turn is time-consuming and labor-intensive. This goal has also been outlined in a recent shared task on scientific paper summarization, the 3rd Computational Linguistics Scientific Document Summarization Shared Task[1].

The first step towards building a system that can obtain information about an RP from a citing paper (CP) that cites it, is to find spans of text in the RP that are cited by a particular citance in the CP. In the context of the above-mentioned shared task, this first step is referred to as Task 1A. Task 1A, thus offers a good foundation for the goal mentioned above, by identifying the relevant reference sentences for a citance. We present a novel approach to Task 1A. While we build on previous work by Yeh et al. (2017), our major contributions can be described as:

- We model a new feature set to represent a citance-reference sentence pair along with building a classification system that uses a binary classification technique for classifying a <CP sentence, RP sentence> pair according to whether the CP sentence cites the RP sentence or not.
- We show performance gains over the results of Yeh et al. (2017)(which is the current state-of-the-art) by achieving better F1-scores, using a feature set that has lesser number of features than that used in the above work.
- We explore various measures for evaluating similarity between texts while build-

[1]http://wing.comp.nus.edu.sg/ cl-scisumm2017/

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 23–32

ing this feature set. Feature representations extracted (as described later), are used to train three binary classifiers - an Adaptive Boosting Classifier (ABC), a Gradient Boosting Classifier (GBC) and a CNN classifier.

The datasets provided for this year's as well as last year's shared task have been used.

2 Related Work

There has been a large amount of work on the task of summarizing scientific documents. However, as is clear from review surveys and papers such as Jones (2007), Teufel and Moens (2002) and Nenkova (2011), just using citances of a paper does not taken into account the context of a user or place the paper in a larger perspective of related work. Most of the related work on the task of identifying text spans in the RP that correspond to a particular citance, have been presented at the shared task mentioned in the previous section. We highlight some relevant work and various methods used for this task.

Yeh et al. (2017) also used a binary classification approach for Task 1A, as we do. They used five classification algorithms to learn the binary classification model, with L2-SVM performing the best. Moraes et al. (2016) used SVM with subset tree kernel, a type of convolution kernel. They computed similarities between three tree representations of the citance and reference text. Li et al. (2016) used an SVM classifier with a topical lexicon to identify the best matching reference spans for a citance, using IFD similarity, Jaccard similarity and context similarity. The PolyU system by Cao et al. (2016), for Task 1a, used SVM-rank with lexical and document structural features to rank reference text sentences for every citance. Klampfl et al. (2016) applied a modified version of an unsupervised summarization technique (TextSentenceRank) to the reference document. Nomoto (2016) treated the problem as a ranking problem, learning one component of the similarity through a neural network and using TF-IDF scores for the other component. Aggarwal and Sharma (2016a) employed lexical and syntactic dependency cues in writing rules to extract text spans in the RP for a given CP citance. Malenfant and Lapalme (2016) presented a novel approach to solve this task. They first performed another task of identifying the facet of each sentence of the RP. These facets belonged to a predefined set of facets, such as *introduction, abstract, results, etc.* They then used the facet information to match each sentence to a citance having the same facet in the CP.

3 Method

The structure of the dataset is described in Section 4.1. Citances and their actual reference texts are extracted from gold-standard annotations. Citances in CPs are paired with each sentence in the RPs, along with a binary label indicating their actual reference relations - 0 if the citance actually refers to the RP sentence and 1 if it doesn't. For each such pair, a feature vector is extracted that describes the relatedness between the given citance and the reference sentence. These feature vectors, along with their corresponding labels, are used to train the binary classifiers.

3.1 Feature Extraction

As mentioned in the section on related work, most approaches to this task have either been based on ranking of possible cited sentences in the RP for a given CP citance, or on binary classifying each RP sentence as relevant or irrelevant to a given CP citance. We use the latter approach. Our method is based on the assumption that a CP sentence and corresponding RP sentence should be semantically and lexically similar, representing similar meaning, idea or abstract concept. This is a natural assumption to make, since modeling the problem based on this assumption helps to separate relevant sentences (to the CP citance) in the RP from irrelevant ones. Inspired by the idea of Yeh et al. (2017), the feature set for each citance-reference pair is divided into three different *classes* of citation-dependent features (i.e., lexical, knowledge-based and corpus-based) and one *class* of citation-independent features (i.e., surface). However, we must mention here that our work is significantly different from Yeh et al. (2017), when it comes to the set of features used. Through control experiments (Section 5.1), we show the effect of using our set of feature over theirs. We incorporate several modified and newly added features.

The features marked by an asterisk (*) are the ones that are borrowed, but modified. The features marked by two asterisks (**) are the newly added features in this work. For features that have been borrowed from Yeh et al. (2017), more elaborate details about them can be seen in their work.

3.1.1 Lexical features

This class deals with the features representing similarity measure of words for each pair of citance and reference sentence. As suggested by the results of Kenter et al. (2016) for short text similarity tasks, the overall sentence similarity measure based on each feature is calculated by averaging the similarities over all the words in the sentences.

1. **Word overlap*:** Word overlap between the citance and the reference sentence based on five metrics: *Dice coefficient, Jaccard coefficient, Cosine similarity, Levenshtein distance based fuzzy string similarity and modified gestalt pattern-matching based sequence matcher score*, the last one as reported by Ratcliff and Metzener (1988).

2. **TF-IDF similarity:** The TF-IDF vector cosine similarity between the citance and the reference sentence as reported by Baeza-Yates and Ribeiro-Neto (2011).

3. **ROUGE measure:** The ROGUE (Lin and Hovy, 2003) metrics used are: ROGUE-1, ROGUE-2 and ROGUE-L.

4. **Named entity overlap*:** Measured using Dice coefficient, fuzzy string similarity, sequence matcher score and word2vec similarity.

5. **Number overlap*:** Number overlap between the citance and the reference sentence measured by fuzzy string similarity and sequence matcher score.

6. **Significance of citation-related word pairs:** The number of significant word pairs and the summation of significance scores of such word pairs extracted for each citance-reference pair based on Pointwise Mutual Information (PMI) score (Church and Hanks, 1989) collected from a dictionary containing significant words pairs appearing in the cited citance-reference pairs.

3.1.2 Knowledge-based features

1. **WordNet-based semantic similarity*:** The best semantic similarity score between words in the citance and the reference sentence out of all the sets of cognitive synonyms *(synsets)* present in the WordNet, following Miller (1992) and Pedersen et al. (2004).

3.1.3 Corpus-based feature

1. **Word2Vec-based semantic similarity**:** The word-to-word semantic similarity between the citance and the reference sentence is obtained based on the pre-trained embedding vectors of the GoogleNews corpus, following Mikolov et al. (2013). Campr and Jezek (2015) show several advantages that such embeddings offer, compared to those generated by traditional algorithms, such as LSA.

3.1.4 Surface features

This class includes features dealing primarily with the morphology of the reference sentences. These include:

1. **Count of words:** The count of words in the reference sentence.

2. **Count of characters**:** The total count of all characters in the reference sentence.

3. **Count of numbers:** The count of numbers in the reference sentence.

4. **Count of special characters**:** The number of special characters in the reference sentence : *“@”, “#”, “$”, “%”, “&”, “*”, “-”, “=”, “+”, “>”, “<”, “[”, “]”, “{”, “}”, “/”*.

5. **Normalized count of punctuation markers**:** The ratio of count of punctuation characters to the total count of characters in the reference sentence.

6. **Count of long words**:** The count of words in the reference sentence exceeding six letters in length.

7. **Average word Length**:** The ratio of count of total characters in a word to the count of words in the reference sentence.

8. **Count of named entities:** The number of named entities in the reference sentence.

9. **Average sentiment score**:** The overall positive and negative sentiment score of the reference sentence averaged over all the words, based on the SentiWordNet 3.0 lexical resource as described by Baccianella et al. (2010).

10. **Lexical richness**:** The lexical richness of the reference sentence based on Yule's K index.

3.2 Classification Algorithms

As our approach treats the training data as pairs of citances and reference sentences, the number of reference sentences that a citance refers to is much smaller for a reference paper, leading to a highly imbalanced data set with the ratio of non-cited to cited pairs being 383.83 : 1 in the combined corpus of development and training set and 355.76 : 1 in the test set corpus. This is not surprising since CPs usually cite only a small text span of an entire RP. Hence, our dataset is hugely imbalanced with negative examples being the majority. Following the work of Bowyer et al. (2002), we experimented with combinations of three different degrees of Random under-sampling (20%, 30% and 35%) on the majority class (negative samples). On each undersampled dataset, we apply the SMOTE (Synthetic Minority Over-sampling Technique) method (Bowyer et al. (2002)) to generate synthetic cited pairs until the ratio of the cited to non-cited pairs is 1:1. The best results were obtained with 30% random undersampling rate. To take care of correlated features, if any, Principal Component Analysis (PCA), following Tipping and Bishop (1997) is applied on both training and testing feature sets. Experiments were done by varying the number of principal components from 30-40 and the best performance was obtained by retaining the top 35 principal components.

For the classification task, we use two boosting ensemble algorithms: Adaptive Boosting Classifier (ABC) as described by Abe et al. (1999), Gradient Boosting Classifier (GBC) as described by Friedman (1999) and a deep Convolutional Neural Network (CNN) as described by Schmidhuber (2015).

The implementation of ABC and GBC rely on sci-kit learn[2], while the CNN is implemented using Keras[3] (Chollet et al. (2015)). Gensim (Rehurek and Sojka (2010)) is used to carry out word2vec related operations.

3.2.1 Boosting Ensemble Algorithms

Boosting ensemble algorithms work by creating a sequence of models that attempt to correct the mistakes of the models used before them in the sequence. Therefore, these offer the added benefit of combining outputs from weak learners (those whose performance is at least better than random chance) to create a strong learner with improved prediction performance, by paying higher focus on instances that have been misclassified or have higher errors by preceding weak rules. This is assisted by a majority vote of the weak learner's predictions weighted by their individual accuracy. Figure 1 shows the illustration of such a boosting framework, as described by Bishop and Nasrabadi (2007).

The **Adaptive Boosting Classifier (ABC)** algorithm works in a similar way discussed above. The base classifier (or weak learner) used in this case is a decision tree.

Gradient Boosting Classifier (GBC), on the other hand, begins by training several models sequentially on the original data set while allowing each model to gradually minimize the loss function of the whole system using the Gradient Descent method, as described by Collobert et al. (2004). The

[2]http://scikit-learn.org
[3]https://keras.io

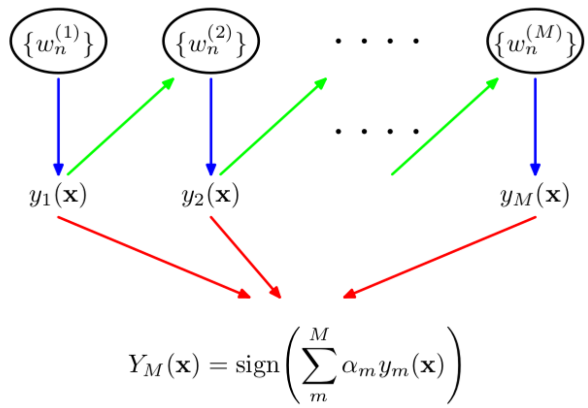

Figure 1: Schematic illustration of the boosting framework. Adapted from Bishop and Nasrabadi (2007): each base classifier $y_m(x)$ is trained on a weighted form of the training set (blue arrows) in which the weights $w_n(m)$ depend on the performance of the previous base classifier $y_m{-}1(x)$ (green arrows). Once all base classifiers have been trained, they are combined to give the final classifier $Y_M(x)$ (red arrows)

base classifiers in a GBC are regression trees. Since our task is a binary classification, only one regression tree is used as a special case.

3.2.2 Convolutional Neural Network

Convolutional Neural Networks, as described in Schmidhuber (2015), have the ability to extract features of high-level abstraction with minimum pre-processing of data. They have been widely used for sentence classification problems, such as by Kim (2014). Recently, Ngoc Giang et al. (2016) also used CNNs for a sequence classification problem involving classification of DNA sequences by considering these sequences as text data. Given the success of CNNs on these, we explore their use in our task.

However, in our case, a class-imbalance problem occurs due to the number of positive reference-citance instances being far too low *(495)*. These are too few examples for any deep learning model to extract meaningful features from the original text. Using the original sentences, and modeling it directly as a sequence classification on pairs of sentences would introduce too much sparsity owing to this imbalance. Not surprisingly, our experiments on using original sentences directly in an attention-based RNN model (2015) resulted in a precision score of 0.002 for positive and 0.24 for negative samples. Thus, we choose to train the CNN on the feature sets as inputs instead of the sentences directly. Figure 2 describes the CNN architecture chosen by us after repeated experiments and tuning on the development data.

A 1D Convolutional layer accepts inputs of the form *(Height * Width * Channels)*. In our case, we can visualize each feature vector as an image with a unit channel, unit height and a width equal to the number of features in the reduced feature vector obtained after applying PCA. Therefore, the input shape for the vector to be fed into the input layer of the CNN, becomes *(No. of features * 1)*.

3.3 Post Filtering

The binary classifier may classify multiple sentences in the RP as positive, i.e., being relevant to a particular citance. However, the existence of inherent errors in the model means that all of these sentences may not be in the ideal text span of the RP corresponding to the citance. In order to reduce our false positive error rate, we post-process by filtering out some of these false positives. We use the approach of Yeh et al. (2017) for the post-filtering task. In this approach, the final output denotes the top-k sentences from the ordered sequence of classified reference sentences based on the TF-IDF vector cosine similarity score to measure the relevance between the citance and the reference sentences. All sentences other than the top-k are not included in the final output text span, even though the model might have labelled them as positive.

4 Experiments

4.1 Dataset

We use the development corpus, the training corpora and the test corpus provided for the CL-SciSumm Shared Tasks 2016[4] and 2017[5]. As reported in Jaidka et al. (2016), each corpus comprises 10 reference articles, their citing papers and annotation files for each reference article. The citation annotations specify

[4]http://wing.comp.nus.edu.sg/cl-scisumm2016/
[5]http://wing.comp.nus.edu.sg/ cl-scisumm2017/

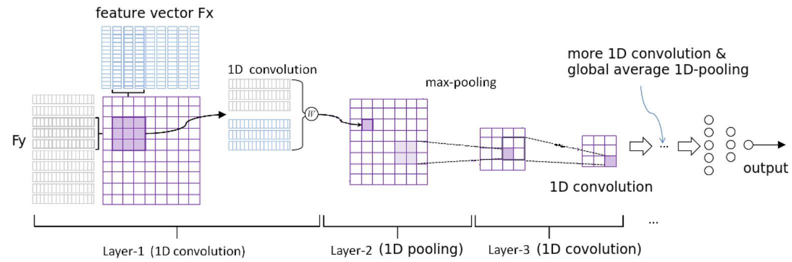

Figure 2: Our CNN architecture: stack of two 1-D convolutional layers with 64 hidden units each (ReLu activations) + 1-D MaxPooling + stack of two 1-D convolutional layers with 128 hidden units each (ReLu activations) + 1-D Global Average Pooling + 50% Dropout + a single unit output dense layer (sigmoid activation)

citances, their associated reference text and the discourse facet that it represents.

4.2 Experimental Settings

Precision, Recall and F1-Score are used as evaluation metrics. The average score on all topics in the test corpus is reported. We run experiments on two separate training sets.

In the **first run**, we use data only from the 2016 shared task, and not from the 2017 shared task. This is because we need a common ground for comparison with the existing state-of-the-art (Yeh et al. (2017)), which used this dataset. We first train our data on the training set, and tune the CNN's hyperparameters on the development set. We then augment the training data and the development data to train the final models. We test our model on the test provided as part of this dataset. Table 2 shows the performance of the CNN model on this test set, and compares it with the existing state-of-art and another well-performing model. We have reported only the CNN's performance in this table as (as will be seen in the results of the second run), this is a better performing model than ABC and GBC, in our experimental setup.

In the **second run**, we make use of the datasets from both 2016 and 2017. Both the training datasets are augmented to form the initial training set. After tuning the CNN's hyperparameters on the development set (which is the same for both 2016 and 2017), the initial training and development sets are augmented to form the final training set. Grid search algorithm, as given by Bergstra and Bengio (2012), over 10-fold cross validation is used to find the best model parameters for ABC and GBC listed in Table 1. Since the gold-standard annotations for the 2017 test set were not yet available at the time of conducting our experiments, we use only the test set of 2016. We report performance of ABC, GBC as well as the CNN classifier on this test set. Table 3 shows these results.

Table 1: **Model parameter settings**

Classifier	**Architecture and Parameter settings**
ABC	Learning rate for shrinking the contribution of each decision tree = **1.3**, Boosting algorithm = **SAMME.R** for faster convergence
GBC	Learning rate for shrinking the contribution of regression tree = **0.15**, Loss = **Deviance** for probabilistic outputs, No. of Boosting stages = **100**

5 Results and Analysis

Precision, recall and F1-score obtained by the models on the test set with respect to the positive classes, evaluated by 10-fold cross validation are shown in Table 3. The CNN-based classifier was trained for 30 epochs. The best scores for each metric have been shown in bold.

Table 2 shows a comparison of the F1-score achieved by our model with that of the pre-

Table 2: **F1 scores of previous models**

System	F1-scores
Yeh et al. (2017)	0.1443
Aggarwal and Sharma (2016b)	0.11
Our Method	**0.2462**

vious models used for the task. The L2-SVM system by Yeh et al. (2017) produced an F1-score of 0.1443, which is the highest reported yet to the best of our knowledge. Our model outperforms it in terms of F1-score. It must be mentioned here that Klampfl et al. (2016) reported an F1-score of 0.346 on the development set corpus and 0.432 on the training set corpus of 2016. However, we have not considered their system in Table 2 because of the unavailability of their performance results on the test set corpus. Figure 3 compares the performance of our CNN classifier with their TextSentenceRank assisted sentence classifier on the development and training set corpus *(80:20 train:test split)* of 2016. Although the CNN classifier performs better on both the corpora, the improvement on the development corpus is much more significant than that on the training.

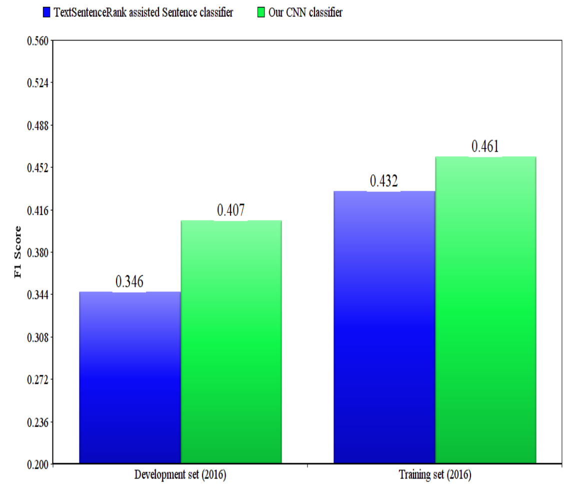

Figure 3: A comparison of the performance of our CNN-based classifier on the development and training set of 2016 with that of Klampfl et al. (2016)

5.1 Control Studies

We run control studies to analyze the contribution of each class of features to our final performance. We also control for the different techniques used, such as SMOTE and PCA, to see their effect. These control studies also help us to understand why our model outperformed the previous state-of-art. Since the CNN is our best performing classifier, we make use of it to perform these control studies.

5.1.1 Effect of feature classes

Figure 4 shows the effect of each class of features. We obtain these graphs by removing one class of features each time from the feature set and calculating the performance using all the other classes. From the bar plot in Figure 4, it is apparent that the class of lexical features contributes the most in distinguishing between a positive and a negative example. Not using this class of features gives 0.3371 lesser F1-score than when using all the features. This means that an information retrieval based component to this problem using lexical features such as TF-IDF, ROUGE etc. as mentioned in section 3.1.1 is the most important for this task. It is also possible that this class shows the maximum effect because of the good number of features in this class, i.e., 6. The second most significant class of features is the class of corpus-based features. Not using this class of features gives 0.3002 lesser F1-score than when using all the features. Our class of corpus-based features has just the word2vec feature. It is not surprising that this feature shows a high impact because word2vec representations capture a good level of semantic and syntactic similarity, which was one of the assumptions we built the model on. Not using the class of surface features gives 0.2429 lesser F1-score than when using all the features. One reason for the impact of surface features could be that it is perhaps the only class of features that takes numbers and special characters into account, and these are significantly high in number in scientific pa-

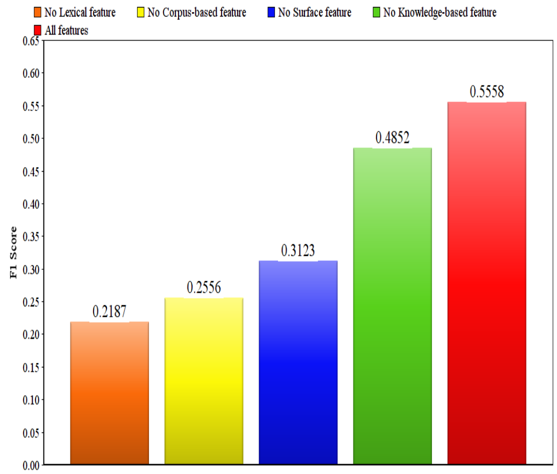

Figure 4: F1 Scores of CNN Model with different feature selection settings

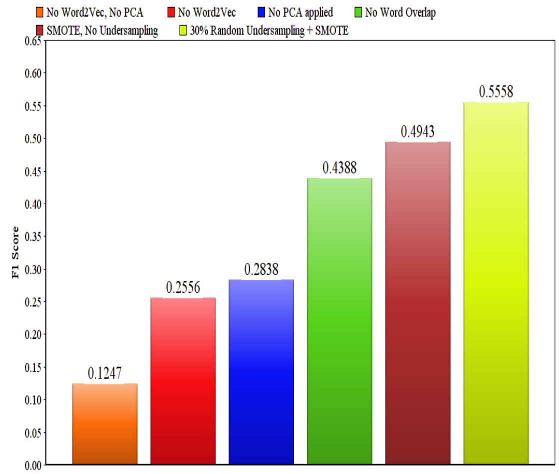

Figure 5: F1 Scores of CNN Model with different feature selection settings

pers. This class also has a high number of features, i.e., 10. Not using the class of surface features gives 0.0706 lesser F1-score than when using all the features. The plot also shows that the impact of WordNet-based features contribute the least to distinguishing between positive and negative examples.

It might therefore be concluded that part of the reason why our model outperforms the state-of-the-art is that their model does not make use of word2vec, while we do so. It also appears that the modified lexical features that we have used, namely, named entity overlap, number overlap and word overlap provide an added advantage to our model, over the state-of-the-art.

5.1.2 Effect of data-handling techniques

Figure 5 shows the contribution of different pre-processing and processing techniques used such as SMOTE (oversampling), undersampling and dimensionality reduction using PCA. There are a few observations that can be drawn from the bar plot in Figure 5. Firstly, dimensionality reduction is a crucial important step in this task. When PCA was not used on the feature set, the performance dropped from 0.5558 to 0.2838, which is a reduction in F1-score of 0.2720. The existing state-of-the-art has a higher number of features than our work does, and does not perform dimensionality reduction on these features, which might be one of the reasons behind the better performance achieved in our work. Further, we see that oversampling using SMOTE gives an improvement of 0.0555 and using undersampling over and above this further improves the performance by 0.0615.

Table 3: **Results obtained by different models**

Model	Precision	Recall	F1-score
ABC	0.7141	0.3579	0.4925
GBC	**0.7439**	0.3237	0.4512
CNN	0.6556	**0.5973**	**0.5558**

5.1.3 Classifier-wise performance

Table 3 shows the performance on the combined dataset of the 2016 and 2017 versions of the shared task, as described in more detail in section 4.2 as the ‘second run’. The CNN model gives the best performance on recall and F1-score, while the GBC model gives the best precision. Precision for each classifier is considerably higher than that of recall, indicating that there are relatively few false positives while a significant number of true positives have been missed out.

6 Future Work

To our knowledge, this is the first attempt which has used a deep learning model for addressing the task. However, for training our model, we had to be entirely dependent on the feature sets extracted. The number of positive instances in the corpus provided is still too low to train the model using the conventional CNN sequence-to-sequence approach, which, given more data, might be able to learn more interesting patterns in the citance-reference pairs. Also, recent extensions to word2vec such as the **Paragraph Vector** (Le and Mikolo (2014)) can be used to further enhance the semantic similarity measures between the reference-citance pairs.

Furthermore, the binary labels assigned to each <CP sentence, RP sentence> pair can be used to establish some partial order in between the training instances, which in turn, can help in modeling the task as a Learning to rank problem. This ordering can then be incorporated to predict the relevance-based ranking of referenced sentences for a citance.

7 Conclusions

We describe our work on reference scope identification for citances using an extended feature set applied to three different classifiers. Among the classifiers trained to distinguish cited and non-cited pairs, the CNN-based model gave the overall best results with an F1 score of 0.5558 on the combined corpus of CL-SciSumm 2016 and 2017. We also achieved an F1 score of 0.2462 on the 2016 dataset, which surpasses the previous state-of-the-art accuracy on the dataset. In addition to this, we carry out control studies reporting the contribution of various feature classes as well as feature selection methods that have been used by us.

References

Naoki Abe, Yoav Freund, and Robert E. Schapire. 1999. A short introduction to boosting.

Peeyush Aggarwal and Richa Sharma. 2016a. Lexical and syntactic cues to identify reference scope of citance. In *BIRNDL@ JCDL*, pages 103–112.

Peeyush Aggarwal and Richa Sharma. 2016b. Lexical and syntactic cues to identify reference scope of citance. In *BIRNDL@JCDL*.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*.

Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. 2011. Modern information retrieval - the concepts and technology behind search, second edition.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Christopher M. Bishop and Nasser M. Nasrabadi. 2007. Pattern recognition and machine learning. *J. Electronic Imaging*, 16:049901.

Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357.

Michal Campr and Karel Jezek. 2015. Comparing semantic models for evaluating automatic document summarization. In *TSD*.

Ziqiang Cao, Wenjie Li, and Dapeng Wu. 2016. Polyu at cl-scisumm 2016. In *BIRNDL@JCDL*.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Ronan Collobert, Patrick Gallinari, Léon Bottou, Hélène Paugam-Moisy, Samy Bengio, and Yves Grandvalet. 2004. Large scale machine learning.

Jerome H. Friedman. 1999. Greedy function approximation: A gradient boosting machine.

Kokil Jaidka, Muthu Kumar Chandrasekaran, Sajal Rustagi, and Min-Yen Kan. 2016. Overview of the cl-scisumm 2016 shared task. In *BIRNDL@JCDL*.

Karen Spärck Jones. 2007. Automatic summarising: The state of the art. *Information Processing & Management*, 43(6):1449–1481.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *CoRR*, abs/1606.04640.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Stefan Klampfl, Andi Rexha, and Roman Kern. 2016. Identifying referenced text in scientific publications by summarisation and classification techniques. In *BIRNDL@JCDL*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Lei Li, Liyuan Mao, Yazhao Zhang, Junqi Chi, Taiwen Huang, Xiaoyue Cong, and Heng Peng. 2016. Cist system for cl-scisumm 2016 shared task. In *BIRNDL@JCDL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Bruno Malenfant and Guy Lapalme. 2016. Rali system description for cl-scisumm 2016 shared task. In *BIRNDL@ JCDL*, pages 146–155.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A. Miller. 1992. Wordnet: A lexical database for english. *Commun. ACM*, 38:39–41.

Luis Moraes, Shahryar Baki, Rakesh M. Verma, and Daniel Lee. 2016. University of houston at cl-scisumm 2016: Svms with tree kernels and sentence similarity. In *BIRNDL@JCDL*.

Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Nguyen Ngoc Giang, Vu Anh Tran, Duc Luu Ngo, Dau Phan, Favorisen Lumbanraja, M Reza Faisal, Bahriddin Abapihi, Mamoru Kubo, and Kenji Satou. 2016. Dna sequence classification by convolutional neural network. 09:280–286, 01.

Tadashi Nomoto. 2016. Neal: A neurally enhanced approach to linking citation and reference. In *BIRNDL@ JCDL*, pages 168–174.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet: : Similarity - measuring the relatedness of concepts. In *AAAI*.

John W. Ratcliff and David Metzener. 1988. *Pattern Matching: The Gestalt Approach*.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks : the official journal of the International Neural Network Society*, 61:85–117.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Computational linguistics*, 28(4):409–445.

Michael E. Tipping and Christopher M. Bishop. 1997. Probabilistic principal component analysis.

Jen-Yuan Yeh, Tien-Yu Hsu, Cheng-Jung Tsai, and Pei-Cheng Cheng. 2017. Reference scope identification for citances by classification with text similarity measures. In *ICSCA '17*.

32