Final Project Report

Aditya Khandelwal 20171117
Vani Sancheti 20171179

May 1, 2020

Abstract

In recent years there has been an increasing interest in approaches to scientific summarization that take advantage of the citations a research paper has received in order to extract its main contributions. In this context, the CL-SciSumm 2019 Shared Task has been proposed to address citation-based information extraction and summarization. In this report we present several systems to address two of the CL-SciSumm tasks. Notably, supervised systems to match citing and cited sentences (Task 1A), and again a supervised approach to identify the type of information being cited (Task 1B).

1 Introduction

Although scientific summarization has always been an important research topic in the area of natural language processing (NLP) in recent years new summarization approaches have emerged which take advantage of the citations that a scientific article has received in order to extract and summarize its main contributions.

CL-SciSumm explores summarization of scientific research in the domain of computational linguistics research. The Shared Task dataset comprises the set of citation sentences (i.e., "citances") that reference a specific paper as a (community-created) summary of a topic or paper. Citances for a reference paper are considered a synopses of its key points and also its key contributions and importance within an academic community. The advantage of using citances is that they are embedded with meta-commentary and offer a contextual, interpretative layer to the cited text. Citances offer a view of the cited paper which could complement the reader's context, possibly as a scholar.

2 Literature Review

Previous work has shown the importance of the citation summaries in understanding what a paper says. The citation summary of an article A is the set of sentences in other articles which cite A. (Elkiss et al., 2008)(1) performed a large-scale study on citation summaries and their importance. They conducted several experiments on a set of 2, 497 articles from the free PubMed Central (PMC) repository . Results from this experiment confirmed that the "Self Cohesion" (Elkiss et al., 2008)(1) of a citation summary of an article is consistently higher than the that of its abstract.

Many system were registered for the both the subtask. Some system that we explored:

1. **System1** This system was submitted by University of Manchester. They explored supervised and semi-supervised approaches. They also did fine tuning of BERT to make it task specific. They formalised the problem as a ranking problem based on similarity and used Bilateral multi-perspective matching model for identifying the cited span. The model encodes the two input sentences using BiLSTM and then matches the encoded one in both directions (2)

2. **System2** This system was developed at LaSTUS-TALN+INCO. In this system for Task 1A they included a supervised approach based on recurrent neural networks and an unsupervised system based on sentence similarity, for Task 1B one supervised approach. (3)

3 Problem Statement

1. **Task1A :** For each citance (i.e. a citation sentence that references the RP), identify the spans of text (cited text spans) in the RP that most accurately reflect the citance.

2. **Task1B :** For each cited text span, identify what facet of the paper it belongs to, from a predefined set of facets that mainly are:

   (a) Method

   (b) Aim

   (c) Implication

   (d) Results

   (e) Hypothesis

3.1 Links to our code and the dataset

1. A dataset of around 1000 articles and their respective citing papers were provided by the organizers of the challenge.We separated the data of documents in 75% for training and 25% for the validation set for most of the models described below. Link to the dataset: `https://github.com/WING-NUS/scisumm-corpus`

2. Colab Link for Task1A

3. Colab Link for Task1B

4 Task 1A

For this task we extracted citing and text spans from the reference paper from the datasets provided by the organizers. We have used these for training our models to learn how to classify valid or invalid pairs. We pre-processed the XML as well as annotated files for extracting both positive as well as negative pairs. The positive pairs are the one that are

provided in the annotated file, while the negative pairs are the one created by randomly selecting some text span for a particular citing sentence. The data was pre-processed by tokenizing the text and then each token was lemmatised.

4.1 Model

We experimented different approaches for this task, taking into consideration that the main notion is to identify sentence similarity between the citing text and the text span from the reference paper.

Supervised Model In this we followed a supervised approach that consists of a neural network architecture for finding the best-matched sentences from the reference document. For this we considered building two identical sub-networks (Siamese networks) which is used for the two inputs (citing and cited text pair). Architecture for the model:

1. *Input layer*: Citance and Reference sentences.
2. *Embedding layer*: We experimented using glove, Bert and SciBert sentence embeddings.
3. *LSTM Layer*.
4. *Fully connected layer*.
5. *Prediction Layer*: Calculating manhattan distance in this layer. This unit indicating the similarity between the citance and the reference sentence.

UnSupervised Model For this approach we performed semantic search. We used Bert Based embedding for this approach and took cosine similarity between the citing sentence and each sentence of the reference paper. The top 3 sentences are then considered as the most similar match and thus these sentence form the span of text that is relevant to the citing sentence.

4.2 Sentence Embeddings

We experimented with different sentence embeddings:

1. **Glove Word Embedding**: We have used Glove word embeddings for computing word embeddings and then sentence embedding were computed by doing an average of all the word embeddings. We have used Glove 300d pre-trained embedding for this task.
2. **BERT pre-trained Embeddings**. We next explored two Bert based models. We used BERT as it produces word embedding that are dynamically controlled by the surrounding words.
   (a) **bert-base-nli-mean-tokens**: BERT-base model with mean-tokens pooling.

(b) **SciBERT** Since the above bert model were trained on data from general domains, therefore we explored this model which is closer to the CL-SciSumm domain.

5 Task 1B

In this section we present experiments aimed at identifying the facets the cited text spans belong to. For this task we did not use any information from the citing sentences. The features were generated exclusively based on the identified cited text of the RP. In the next section we describe the set of features generated.

5.1 Features

5.1.1 Positional Features

The sentence position in a paper can inform about the facet the sentence belongs to. For instance, sentences at the end of the document would probably belong to the Result facet. We use two features based on the location of the sentence in the reference document:
– Sentence position: the position of the sentence in the reference paper;
– Section sentence position: the position of the sentence in the section;

5.1.2 BERT based sentence embedding

We use BERT to extract features, namely sentence embedding vectors, from the reference sentences. BERT offers an advantage over models like Word2Vec, because while each word has a fixed representation under Word2Vec regardless of the context within which the word appears, BERT produces word representations that are dynamically informed by the words around them. We encode each reference sentence (identified cited text of the RP) into a vector of size of 768. Finally we get a vector of size of 770 by concatenating these two features.

5.2 Models

From Figure 1, we can clearly look how imbalanced the given dataset is. To tackle this problem we go with 2-layered classification pipeline. First we classify whether the given instance belong to the method facet or not. Then a multi-class classifier to identify one of the other facets in case it was previously classified as not-Method.

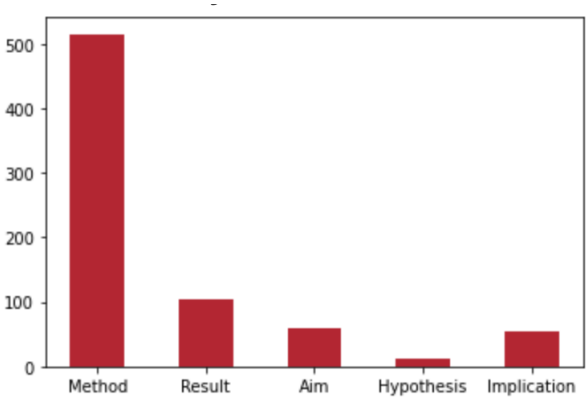

Figure 1: Histogram of dataset

5.3 Experiments

The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor performance on, the minority class, although typically it is performance on the minority class that is most important. So we experimented with different techniques to tackle this problem :

1. **Oversampling :** We used Synthetic Minority Oversampling Technique(SMOTE) to synthesize new examples from the existing examples. SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line.

2. **Undersampling :** Undersampling techniques remove examples from the training dataset that belong to the majority class in order to better balance the class distribution. For this we tried CNN(Condensed Nearest Neighbour) and OSS(One-Sided Selection).

3. **Ensemble Techniques :** We tried boosting based techniques. These models fits sequentially multiple weak learners in a very adaptative way: each model in the sequence is fitted giving more importance to observations in the dataset that were badly handled by the previous models in the sequence.

6 Results

6.1 Task 1A

Performance Results for various embeddings on Validation data			
Embedding	Recall	Precision	F1-Score
Glove	0.446	0.727	0.553
Bert	0.448	**0.730**	0.555
SciBert	**1.000**	0.500	**0.667**
BertEmbedding + Cosine Similarity(Unsupervised)	0.172	0.292	0.216

6.1.1 Observations

When looking at BERT-base and SciBERT models performance, SciBert tends to perform better reason being that is is more focused on the domain of the dataset. Glove and Bert tends to give the same performance.

6.2 Task 1B

We first tried running classification models using only positional features i.e. sid and ssid. But the classifiers we got were biased towards method facet. As shown in the figure.2, in the binary classifier itself the data is skewed. So we tried undersampling of majority class(Method) to match the instances of method and non-method but it gave even worse results. The reason undersampling might not work here is that we have very less training data as a whole and further reducing it, makes it worse.

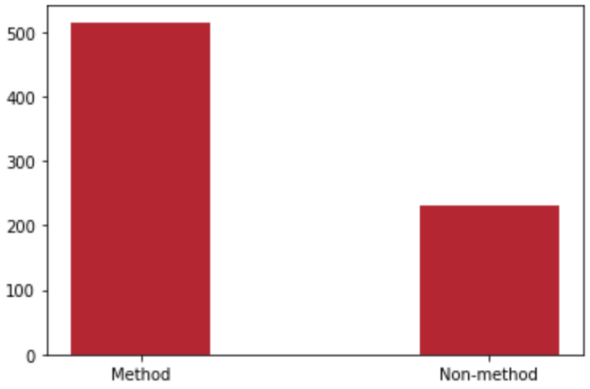

Figure 2: Histogram of Method vs Non-method

Performance Results with only positional features	
Classifier (Binary + Multi-class)	F1-score
Logistic Regression(B) + Random Forest(M)	0.37
SVM(B) + Random Forest(M)	0.33
Random Forest(B) + LR(M)	0.37
Random Forest(B) + Random Forest(M)	0.38

After concatenating sentence embedding vector of length 768, trained using BERT we got much better results.

Performance Results using both positional features and BERT embeddings	
Classifier (B + M)	F1-Score
Logistic Regression(B) + Random Forest(M)	0.49
Random Forest(B) + Random Forest(M)	0.51

6.3 Resampling techniques

As listed in section 5.3, we tried different resampling techniques. Undersampling gave worse results because of very limited training data. Ensemble techniques were also unable to deliver satisfying results. We got the following results using different techniques:

Performance Results of different resampling techniques	
Technique	F1-Score
Random Undersampling	0.49
RUSBoostClassifier	0.41
EasyEnsembleClassifier	0.31
Oversampling using SMOTE	0.71

The EasyEnsembleClassifier is an ensemble of AdaBoost learners trained on different balanced boostrap samples. The balancing is achieved by random under-sampling. RUSBoostClassifier is random under-sampling integrating in the learning of an AdaBoost classifier. Oversampling using SMOTE gave the best results with a **F1 score of 0.71**.

7 Limitations

One of the major challenge that we faced was that the data was biased. For Task1B the proportion of data belonging to the "Method" facet was very high due to which the models we trained were not robust. Resampling techniques like oversampling also do not add any new information to the already available data. Insufficient size of training data was a limitation to our models. Along with this, all of our approaches uses sentence embeddings computed using the pre-trained models, which increases the time for training. For glove we used the embedding of size 300 and for BERT we used embeddings of size 768.

8 Future Scope

In this report we have described various supervised and unsupervised approaches for both the task provided. We have explored various sentence embedding including pretrained BERT models. Further we can train these BERT models by fine tuning it to get embeddings more task specific. For task1A after computing these embeddings we can build a classifier stating the pair of sentences as valid or not.

For task 1B, Feature extraction is very important to get good results from classification models. Better sentence embeddings could result in better classification results. Also more advanced deep learning techniques could extract more meaningful features specific to a task but would also require very high amount of data.

References

Elkiss, Aaron, Siwei Shen, Anthony Fader, Gunes Erkan, David States, and Dragomir R. Radev. 2008. Blind men and elephants: What do citation summaries tell us about a research article? Journal of the American Society for Information Science and Technology

Zerva, C., Nghiem, M.Q., Nguyen, N.T., Ananiadou, S.: UoM@CL-SciSumm 2019. In: BIRNDL2019 (2019)

Luis Chiruzzo1,Ahmed AbuRaed, Alex Bravo, and Horacio Saggion:LaSTUS-TALN+INCO @ CL-SciSumm 2019

Kupiec, Julian, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In SIGIR '95, pages 68–73, New York, NY, USA. ACM.