CMPUT 650 Project Report: Citation Function Classification Using Word Embeddings

Cody A. Rosevear **Erick E. Ochoa**
Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
{rosevear,eochoa}@ualberta.ca

1 Introduction

Citation function classification may be construed as the problem of assigning to a given citation context a label which best characterizes both the valence (positive, negative, or neutral) and rationale of that citation context, given a pre-specified set of labels from which to choose. For example, consider the task of classifying the following citation context:

> The processing described in this paper has been implemented in NTT Communication Science Laboratories' experimental machine translation system ALT-J / E.

In this citation context, the author mentions that a particular methodological approach to some task has been implemented in previous literature. This is a relatively clear example of a citation context which belongs to the class of positive use.

The goal of a complete citation classification system, therefore, should be to take in as an input citation contexts such as the above and classify them with the most appropriate label given the classification scheme currently in use.

Such a system could provide the basis for a number of useful knowledge based applications, including the construction of more accurate and informative knowlegde networks, the contextualization of common bibliometrics such as the h-index and impact factor, and the development of tools to assist in the assesment of academics' competence and familiarity with respect to a scholarly field, as revealed through their citation patterns.

2 Related Work

Given the utility of solving this problem it has attracted a fair amount of previous work, whose methodologies encompass both traditional rule-based approaches and newer machine learning algorithms. Concerning the former approach, Meyers (2013) use a lexical dictionary and hand-crafted rules in an attempt to solve a binary classification task where the classification scheme consists solely of two categories: contrastive and corrroborative, without any regard for the valence of the citation context.

The work of Di Iorio et al. (2013) attempts to incorporate both valence and function into a (non-binary) classification scheme through the use of a pipeline approach which determines the valence and function of a citation context separately, using standard sentiment analysis techniques to determine the valence, and a combination of ontology extraction and pattern matching to determine the citation function.

Teufel et al. (2006) eschew the pipeline approach and instead rely on a k-nearest neighbours algorithm trained on a set of manually annotated citations. In addition to using features defined in previous work (Teufel and Moens, 2002), they use a variety of hand-crafted features derived from keywords, cue-phrases, and syntactic patterns commonly observed to correlate with specific citation functions while annotating the corpus.

Of particular interest is the fact that all of these approaches either rely on hand-crafted rules for explicit pattern matching or hand crafted training features; none of them attempt to represent citation contexts explicitly as sentence vectors in the vein of (Le and Mikolov, 2014), which is itself an extension of word vector representations as described in (Mikolov et al., 2013) to entire sentences and/or paragraphs. As such, the goal of this paper is to explore the effectiveness of using word emdedding representations of citation contexts as opposed to sets of hand-crafted features.

```
<S AZ="OWN" ID="S-28"> The Narration relation is
     characterized by a series of events displaying forward
     movement of time , such as in passage <CREF/> ) . </S>
...
<S AZ="OTH" ID="S-1"> Several researchers <REF CFunc="Neut"
     ID="R-6" TYPE="P">Partee 1984</REF> , <REF CFunc="Neut"
      ID="R-7" TYPE="P">Hinrichs 1986</REF> , <REF CFunc="
     Neut" ID="R-8" TYPE="P">Nerbonne 1986</REF>
```

Figure 1: Raw data from the corpus

Category	**Description**
`Weak`	Weakness of cited approach
`CoCoGM`	Contrast/Comparison in Goals or Methods(neutral)
`CoCo-`	Author's work is stated to be superior to cited work
`CoCoR0`	Contrast/Comparison in Results (neutral)
`CoCoXY`	Contrast between 2 cited methods
`PBas`	Author uses cited work as basis or starting point
`PUse`	Author uses tools/algorithms/data/definitions
`PModi`	Author adapts or modifies tools/algorithms/data
`PMot`	This citation is positive about approach used or problem addressed (used to motivate work in current paper)
`PSim`	Author's work and cited work are similar
`PSup`	Author's work and cited work are compatible/provide support for each other
`Neut`	Neutral description of cited work, or not enough textual evidence for above categories, or unlisted citation function

Figure 2: Citation function classification scheme

2.1 Data

The corpus used for this experiment was obtained from Teufel (2010), and is composed of 161 scientific articles published by the Association for Computational Linguistics (ACL). The contents of the articles have been structured in a machine friendly XML format and manually annotated by experts to denote different sections, paragarphs, sentences, and metadata attributes. Examples of the raw corpus content and its format are exhibited in Figure 1. Note the XML node `REF`, which contains the metadata attribute `CFunc` which is short for Citation Function. This is the correct citation function to which the citation context should be assigned by a classifier in order for the classification to be deemed correct.

The classification scheme used in this corpus was obtained from (Teufel, 2010), and is composed of 12 classes, summarized in Figure 2.

Figure 3 shows the distribution for each of the categories in the whole dataset. The dataset consists of 2,646 citations. As one can see, the dataset is unbalanced. The majority of the citations belong to the class of `Neut` or `PUse`.

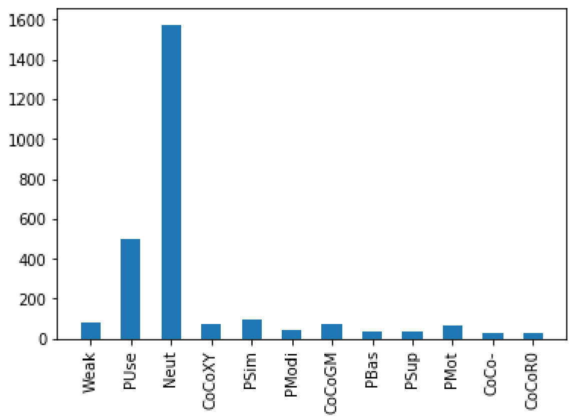

Figure 3: Category Frequency

3 Experiments

Our experiments consisted of using and evaluating various unsupervised models to obtain features for use in classification, as opposed to using handcrafted features. To this end, we experimented with three types of sentence representation:

1. Bag of Words
2. Word Vectors
3. Paragraph Vectors

The Bag of Words approach represents collections of words as a vector of fixed length. Each position in the vector corresponds to a different word. Each word is said to be one-hot encoded. The sentence is then represented by a vector containing the counts for each of the encoded words.

Word vector models are continous vector representations of words. To obtain a representation for words, neural network models are trained on a large corpus with a prediction task (e.g. predict a word given its context or predict the context given the word) (Le and Mikolov, 2014). Facebook recently released pre-trained word vector models that have been trained using the Wikipedia corpus (Bojanowski et al., 2016). Facebook's pre-trained word vectors were used to generate a representation for each of the citations in our corpus. In order to represent entire citation sentences instead of just words, we chose to average the vectors for each of the words in a sentence. This has the effect of representing each sentence as the average of all the words in that sentence.

Paragraph vectors are created via an unsupervised algorithm that learns "fixed-length feature representations from variable-length pieces of text." (Le and Mikolov, 2014). The paragraph vector model can later be used to infer paragraph vectors for unseen paragraphs. Stochastic gradient descent is used to infer these new paragraph vectors.

These three different features were used to train different classifiers. Each classifier had the task of correctly classifying citations according to the 12 categories as defined in Figure 2.

3.1 Tools And Implementation

To test the above methods of citation representation we used three classfiers: k-nearest neighbours, logistic regression, and a support vector machine (SVM). For each classifier we relied on the implementation as provided by the open source Python machine learning library sci-kit learn (Pedregosa et al., 2011). For our vectorizers we again used sci-kit learn in the case of the Bag of Words model, while for the case of paragraph vectors we used the Doc2Vec component of the Gensim library developed by RaRe Technologies (Řehůřek and Sojka, 2010). Facebook's FastText comes pretrained, and so there was no need to create our own vectorizer. We include in the appendix a link to the github repository containing all of the project data and source code.

3.2 System Pipeline

In order to extract features from citations, first the citations were pre-processed in the following way:

1. Lowercase all words.
2. Lemmatize.
3. Filter non-alphabetic characters.
4. Remove English stop words.

These pre-processed citations were then used to train vectorizers for both the Bag of Words and Doc2Vec (paragraph vector) models. Subsequently, these vectorizers were then used to transform the citation training data into vector form and used to train each of the classifiers. Each classifier was then tested on a held out test set to get the final score. The overall pipeline is shown in Figure 4

3.3 Parameter Tuning

For each of the classifiers used in the experiments we identified a subset of their parameters for tun-

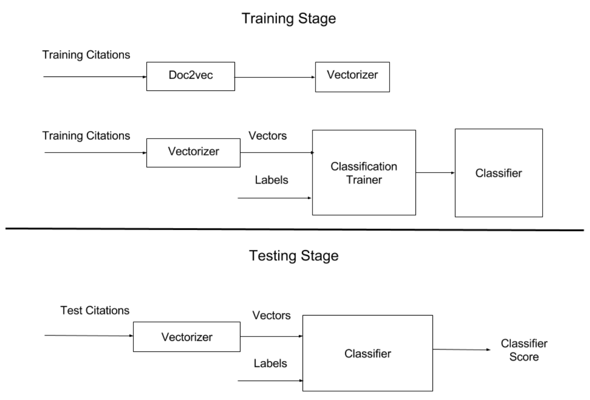

Figure 4: Pipeline

ing. These parameters were chosen based on recommendations found in the official documentation of sci-kit learn (Pedregosa et al., 2011) in conjunction with our own considerations specific to the current task. A finite set of representative values for each parameter were chosen, and an exhaustive search across these parameters and chosen values was performed to obtain the best-scoring parameters for each classifier. We include the final parameters used for each classifier and method of citation representation in the appendix.

Due to the small size of our data set we used 3-fold cross validation on 80% of the data to tune the parameters. Throughout tuning our optimization objective was to maximize the $F1$-Micro average. In order to test the models generalizability we tested all of the models on a held out test set comprising 20% of the original dataset.

4 Results

We now report the $F1$-Micro average obtained by the best scoring models, and, for completeness, the corresponding $F1$-Macro average of these models. The $F1$-Macro average and $F1$-Micro average are given by Equation 1 and Equation 2, respectively, where B is the $F1$ score "calculated based on the number of true positives (tp), true negatives (tn), false positives (fp), and false negatives (fn). Let tp_λ, fp_λ, tn_λ, and fn_λ be the number of true positives, false positives, true negatives and false negatives after binary evaluation for a label λ."(Tsoumakas et al., 2009)

$$B_{macro} = \frac{1}{q}\Sigma_{\lambda=1}^{q} B(tp_\lambda, tn_\lambda, fp_\lambda, fn_\lambda) \qquad (1)$$

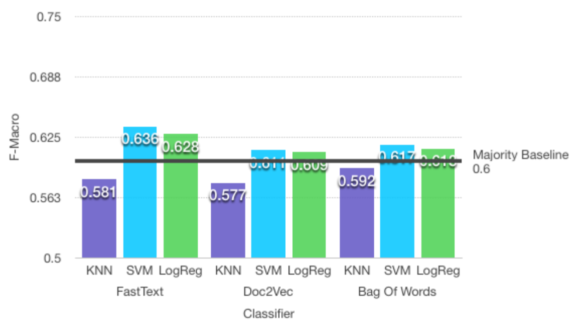

Figure 5: Score using $F1$-Micro

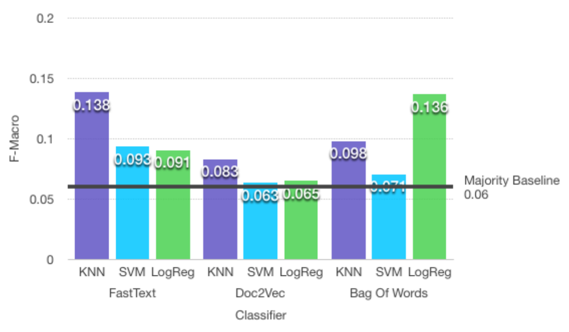

Figure 6: Score using $F1$-Macro

$$B_{micro} = B(\Sigma^q_{\lambda=1} tp_\lambda, \Sigma^q_{\lambda=1} tn_\lambda, \Sigma^q_{\lambda=1} fp_\lambda, \Sigma^q_{\lambda=1} fn_\lambda) \tag{2}$$

Figure 5 and Figure 6 show the $F1$-Micro and $F1$-Macro averages respectively.

Figure 5 shows that most of the classifiers are on par with the baseline. However, the K-nearest neighbor classifier has the worst performance when using $F1$-Micro average as an evaluation metric. More complex classifiers tend to classify better using these features. It seems that word vectors from FastText helped to improve the models, but not significantly.

Figure 6 shows some classifiers beating the baseline by a factor of two. The K-nearest neighbor classifier did better than other more complex classifiers when evaluating using the $F1$-Macro average. It is also interesting to note that simpler features (i.e. Bag of Words) were adequate for this task.

5 Analysis

All of the models perform significantly better when scored using $F1$-Micro average rather than $F1$-Macro average, relative to the majority baseline. This superior performance is due to the imbalance in the data set. Since approximately 75% of the data falls within the categories of Neut or PUse the majority baseline for $F1$-Macro average is artifically low, and thus relatively easy to beat, while the $F1$-Micro average is artificially high, and thus harder to beat.

The fact that $F1$-Micro averages are significantly higher than the corresponding $F1$-Macro averages suggests that our classifiers are performing better on the more populous labels in constrast to the rarer ones. This is again likely due to the imbalanced nature of the dataset: both the classifiers and bag of words and Doc2Vec models are being trained on biased data, and so their capacity to pick out examples of citations not part of the two majority classes is likely leading to underfitting with respect to those classes due to the lack of training instances.

However, data imbalance alone is not the only factor to consider. The usage of Facebook's pretrained FastText model should have ameliorated some of the issues due to data imbalance, since it was trained on a much larger corpus; and, in fact, we do see that for both logisitic regression and the SVM FastText does marginally better than other approaches. One possible explanation for the fact that the improvements are so minor despite the large training corpus is that the Wikipedia corpus does not give the same semantics to words as an entriely academic domain would.

6 Conclusion

In conclusion, the classifiers have little and unbalanced data to train on. As a result, even complex classifiers like SVM perform poorly in the classification task. Classifiers which are designed to better classify unbalanced datasets should be explored in future work. For example, ensemble methods are known to perform better in unbalanced datasets by aggregating the predicting power of many simple classifiers. We believe that proper tuning of parameters in ensamble methods might yield better performance.

Classifiers trained on FastText's word vectors performed best. However, the performance is barely above the majority baseline. We believe that this poor performance is due to the size of the dataset and its unbalanced distribution.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Angelo Di Iorio, Andrea Giovanni Nuzzolese, and Silvio Peroni. 2013. Towards the automatic identification of the nature of citations. In *SePublica*. pages 63–74.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR* abs/1405.4053. http://arxiv.org/abs/1405.4053.

Adam Meyers. 2013. Contrasting and corroborating citations in journal articles. In *RANLP*. pages 460–466.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50. `http://is.muni.cz/publication/884893/en`.

Simone Teufel. 2010. The structure of scientific articles: Applications to citation indexing and summarization (center for the study of language and information-lecture notes) .

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

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '06, pages 103–110. http://dl.acm.org/citation.cfm?id=1610075.1610091.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2009. Mining multi-label data. In *Data mining and knowledge discovery handbook*, Springer, pages 667–685.

A Parameters of reported models using FastText

	KNN	LogReg	SVM

BagOfWord	n neighbors = 5
	weights = uniform
		C = 1
		penalty = l1
		fit intercept = True
			C = 100
			kernel = poly
			degree = 2

FastText	n neighbors = 5
	weights = distance
		C = 1
		penalty = l2
		solver = newton-cg
			C = 1
			gamma = 1
			kernel = rbf

Doc2Vec	n neighbors = 5
	weights = uniform
		C = 1
		penalty = l1
		solver = liblinear
			C = 1
			kernel = linear

Table 1: Parameters used for different classifiers.

B Resources

https://github.com/efferifick/nlp