Combined Tree Kernel-based classifiers for Assessing Quality of Scientific Text *

Liliana Mamani Sanchez
Trinity College Dublin
mamanisl@tcd.ie

Hector Franco-Penya
Dublin Institute of Technology
hector.franco@dit.ie

Abstract

This document describes Tree Kernel-SVM based methods for identifying sentences that could be improved in scientific text. This has the goal of contributing to the body of knowledge that attempt to build assistive tools to aid scientist improve the quality of their writings. Our methods consist of a combination of the output from multiple support vector machines which use Tree Kernel computations. Therefore, features for individual sentences are trees that reflect their grammatical structure. For the AESW 2016 Shared Task we built systems that provide probabilistic and binary outputs by using these models for trees comparisons.

1 Introduction

The system described in this article was submitted to the Automated Evaluation of Scientific Writing (AESW) Shared Task (Daudaravicius et al., 2016).

English is the most common language used for scientific writing across the world. Other things being equal, many scientific writers are non-native English speakers, what leads to a demand for assisting tools that employ "grammar error correction technologies" to compose scientific articles (Daudaravicius, 2015).

Several competitions similar to the one this article addresses have been organized, namely: the Helping Our Own (HOO) Shared Task (Dale and Kilgarriff, 2011; Dale et al., 2012), The CoNLL-2014 shared task on Grammatical Error Correction (Ng et al., 2014). Those evaluations make use of human annotated examples of correct and incorrect grammar (Dahlmeier et al., 2013; Yannakoudakis et al., 2011).

Particularly, Leacock et al. (2010) provide a comprehensive overview of various aspects related to grammar error detection research.

This paper is organized as follows: Section 2 briefly describes the goals of the task our models attempt to address, Section 3 describes our experiments including the proposed Tree Kernel models, whose results are reported in Section 4. Section 5 further comments on the results, and Section 6 concludes with some summarizing remarks.

2 The task

The task consists in identifying sentences that need improvement or correction. This is done by classifying sentences into ones that do need amendments and those that do not. The training dataset provided by the organizers comprises sentences pairs (S_o, S_e), where S_o is a sentence presented as written by a non-native individual, these sentences contain at least one grammatical or lexical mishap that can be corrected. The corresponding S_e has been edited as to make S_o to look as a sentence with good academic style. For instance, example (2), shows a sentence that is an improved version of the one in example (1).

(1) This is called as sub-additivity property of von-Neumann entropy.

(2) This is called a sub-additivity property of von Neumann entropy.

* Both authors contributed equally to the contents and experiments described in this paper.

Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications, pages 223–228,
San Diego, California, June 16, 2016.

The intuition behind this task is that systems that automatically identify sentence candidates for improvement may be built. The core solution would be provided by keeping track of common pitfalls committed by non-native individuals in their writing as in (1). Subsequently, such a system would learn how these pitfalls have been addressed as to emulate text produced by native writers.

The training dataset also comprises a set of sentences written by native English academics; they constitute a body of text that is representative of good academic writing style.

3 Experiments

For our experiments we used constituent trees corresponding to the examples from the training dataset. This dataset provided by the shared task organizers comprises parse tree structures which were generated by using the Stanford parser (Klein and Manning, 2003).

3.1 Tree Kernels

The tree structures were used to train Support Vector Machines using the SVM-light implementation by Joachims (1999) and SubSet Tree kernel (SST) computation tool (Collins and Duffy, 2002a; Moschitti, 2004; Moschitti, 2006) built on top of the former.

In essence, a Tree Kernel classifier computes a kernel function between two trees by comparing subtrees extracted from them. Since the number of possible comparisons between subtrees is exponential, we restricted the choice of subtrees to SubSet Trees (SSTs) (Collins and Duffy, 2002b). A SST is a tree where the leaves may comprise non-terminal symbols and the rules that these trees reflect should be well formed according to the rules of the target language grammar, in this case English.

Tree Kernel methods over Support Vector Machine have been successfully used on many other natural language processing applications, such as Semantic Role Labelling (Moschitti et al., 2008), question answer classification (Moschitti et al., 2007) or relational text categorization (Moschitti, 2008).

Figure 1b illustrates some SubSet trees[1] extracted by the SST kernel from the full syntactic tree structure for (2), which is shown in Figure 1a. Such subtrees will be used as features for the Support Vector Machine model.

Our intuition behind using parse trees comparisons to identify candidate text for correction is that non-native writers will regularly use spurious lexical grammatical structures across their writings. Therefore, given a new sentence to classify, its underlying grammatical structure will be compared to a collection of tree structures built out of a training dataset.

3.2 Models for sentence quality assessment

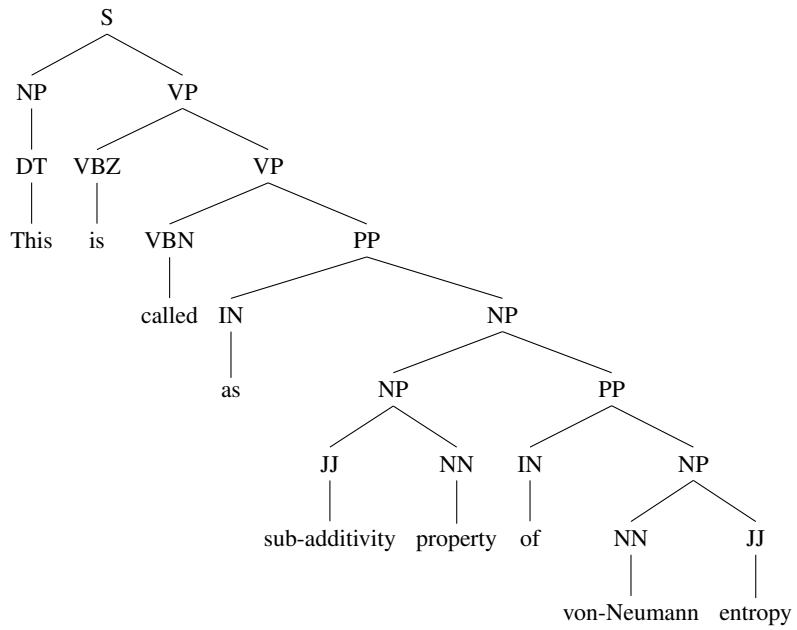
Our overall strategy was to build models that use tree representations of sentences to be used in Support Vector Machines-based systems for sentence quality assessment. SVMs work with training examples labelled as either positive (1) or negative (-1). Therefore, sentences in need of edition are labelled as positive examples while sentences which do not need edition are labelled as negative examples. A machine learning system trained on these examples aims to predict a positive one.

Labelling examples as either positive or negative was translated into labelling tree structures corresponding to these examples with the respective label; for instance, the tree in Figure 1a is deemed a positive example. Internally, the SST tree tool assigns these labels to the corresponding subtrees: following from this, features such as the subtrees in 1b, are labelled as positive examples for the kernel computation.

To prepare the datasets for the training stage, we divided the dataset provided by the organizers in tree groups: D_o: sentences as written by non-native speakers and need edition; D_e are sentences that have a counterpart in D_o which were edited to have an improvement in good academic writing style; and D_n: sentences written by native English academic writers which do not have counterparts in D_o and do not need edition.

We wanted to experiment with different conditions on how the identification of sentences in need of improvement may occur. Therefore, considering these three subdatasets D_o, D_e and D_n, three models were built.

[1] For reasons of brevity, the whole set of SubSet Trees that is extracted from the full parse tree is not shown here.



(a) Constituent tree from the sentence in example (1).

(b) SubSet Trees (SST) extracted from (a).

Figure 1: Example of how SubSet Trees (SST) extract sub trees as features from constituent tree structures.

- $\mathcal{MO}$: uses uniquely sentences in D_o as positive examples, and sentences from D_e as negative examples.
- $\mathcal{ZM}$: uses sentences from D_o as positive examples, and sentences from D_n as negative examples.
- $\mathcal{ZMO}$: uses sentences from D_o as positive samples and sentences from both D_e and D_n as negative samples.

$\mathcal{MO}$ depicts a scenario where the identification would be better for sentence that keep close similarity to the sentences in the training dataset and high precision is paramount. $\mathcal{ZM}$ favours situations where sentences to be assessed do not keep high similarity to the set of positive training examples but the model is able to generalize by contrasting with sentences with good academic style. Ultimately, $\mathcal{ZMO}$ should enclose advantages of the two previous models.

The training dataset of parse trees provided by the organizers contained 475,473 trees for D_o, 476,142 trees for D_e and 725,374 for D_n. Due to their length some sentences were unsuitable for processing by the TK-SVMLight tool, therefore examples whose length was above 1,400 characters were dropped out from the training set. This meant dropping 1,489 examples from D_o, 1,433 from D_e, and 283 from D_n. Thus, 948,693 training examples were used for $\mathcal{MO}$, 1,199,075 examples for $\mathcal{ZM}$, and 1,673,784 examples for $\mathcal{ZMO}$.

Another reason why we chose Tree Kernels computation as a basis for our systems is that we think this sort of classification task should rely uniquely on sentence-level features. Other non-linguistic candidate features could have been taken into account such as the relative position of an individual sentence within a paragraph or document. However, a system built comprising such a feature might not work properly on individual sentences that need classification.

3.3 Training and evaluation procedures

Unfortunately, the computation of kernels for all SubSet Trees is highly demanding in terms of computation time. Due to hardware limitations, for all three models the training data set was split into 100 sub-datasets, and a Support Vector Machine model was trained for each sub-dataset. Then, predictions were computed by using each of those hundred Support Vector Machine models over the unlabelled test dataset provided by the task organizers. The overall numerical categorization value for a system was calculated by averaging over these predictions. This categorization value was normalized to generate the probability of a sentence needing improvement $OutputProb$ using formula(1), where $OutputSVMmodel_i \in [-1, 1]$.

$$OutputProb = \frac{\sum_{i=1}^{100} OutputSVMmodel_i}{200} + 0.5 \quad (1)$$

Because the formula aims to estimate probabilities, if $OutputProb > 1$ then it is floored down to one, if $OutputProb < 0$, then it is rounded up to zero.

Some issues related to the use Stanford parser emerged during the evaluation stage. The parsing procedure for examples from the testing set was expected to produce a parse tree per example, which thereafter would be formatted properly for testing our systems. However, the parser failed to identify the boundaries of some sentences, particularly if there was an abbreviation with a period or a colon symbol occurring in them. For instance, abbreviations such as "etc." or "i e." caused the parser produce two parse trees for a single sentence.

This issue was overcame by running a script that matches the tree structure with the original sentence if the tree structure contained at least 50% of the words in the original sentence, for which no more than 10 consecutive sentences or consecutive trees can be unmatched.

This procedure left 188 sentences without a match (which is negligible amount of the total testing set: 0.13%). For these sentences a probability of 0.5 or a label 'false' was assigned.

4 Results

Table 1 shows the results for the systems submitted to the task organizers.

The system was evaluated according to the predicted label (bin) and according to the predicted probability (prob).

Table 1: Results in terms of Precision, Recall and F-measure for systems that produce either a probabilistic (prob) or binary (bin) prediction. The number in parenthesis points to the relative ranking compared to other systems.

System	Precision	Recall	F_1
$\mathcal{ZM}$ bin	0.4482 (4)	0.7279 (6)	0.5548 (3)
$\mathcal{ZM}$ prob	0.7062 (6)	0.8182 (2)	0.7581 (3)
$\mathcal{MO}$ bin	0.3960 (8)	0.6970 (7)	0.5051 (7)
$\mathcal{MO}$ prob	0.6576 (8)	0.8014 (3)	0.7224 (3)

$\mathcal{ZMO}$ results are not reported as the system did not produce any positive prediction ($F_1 = 0$).

For the competition of binary output systems, our $\mathcal{ZM}$ system performs better than $\mathcal{MO}$ system for $F-$score, resulting ranked in third place. This is 0.073 points behind the best system score (HU), but 0.1041 points better than the baseline system from the task organizers.

Regarding the probabilistic output systems competition, our $\mathcal{MO}$-based system was ranked third. Similarly, this system is 0.073 points behind the best system (HITS) and 0.1073 points above the baseline. Also, this system's results seem to keep correlation with the results from the systems provided by the team NTNU-YZU.

5 Discussion

It is unfortunate the dataset size prompted us to split the training set into a hundred sub-datasets to train their corresponding Support Vector Machine models. This split was done according to the order in which sentences appear on the training set, expecting that indirectly each modified sentence will provide the negative and the positive examples (this is the sentence before and after being edited, S_o and S_e respectively) is likely to fall within same sub-set. While this is a simplified strategy to split the training set, the effectiveness of other methods is an open research question to explore. Clustering algorithms could provide a better split.

The distribution of positive and negative examples for $\mathcal{MO}$ was perfectly balanced as 50% of samples were positive while the other 50% were negative. The $\mathcal{ZM}$ model was fairly balanced, the $\mathcal{ZMO}$ was not balanced (475,473 positive examples in contrast to 952,284 negative examples) and this lead to the corresponding systems produce only negative predictions.

It seems that the F1 measure is proportional to how well balanced each data set is, this could be due to not using the development set to tune the threshold for binary predictions, or to re-arrange probabilities.

Systems trained using the $\mathcal{ZM}$ perform better than the ones using $\mathcal{MO}$. We think a reason for this is that in $\mathcal{MO}$ a positive example shares various subtrees with its corresponding negative example. Therefore, this may affect the classifier ability to calculate predictions for some examples in the test set. It would be ideal to make use of the test set gold standard to have more conclusive insights in this respect.

6 Conclusions

We described in this paper machine learning-based systems for identifying sentences that need amendments to improve their academic style. Our four systems have a Support Vector Machines computations as a core, they were built having tree representations of target sentences as features. The best systems of these four are the ones that use a model where sentences needing improvement are deemed as positive examples, and as negative examples sentences that were not edited and do not correspond to the positive examples are taken into account. This may be due to various reasons such as the distribution of positive and negative examples, or entropy of the datasets. We intend to address these and other issues in our systems in future work.

We think these systems' performance can be improved by modifying: the combination of models, and the software implementation, the representation of features. So far, we have implemented an empirical combination of sub-models output to have a global prediction output.

Making changes in the implementation of tree kernels computation may help to create models that meet scalability requirements. The final prediction would benefit of having less sub-models while keeping computation time reasonable.

Finally, we expect these attempts and future work to contribute to the state of the art of assistive writing technologies.

References

Michael Collins and Nigel Duffy. 2002a. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 263–270. Association for Computational Linguistics.

Michael Collins and Nigel Duffy. 2002b. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 263–270, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31.

Robert Dale and Adam Kilgarriff. 2011. Helping Our Own: The HOO 2011 Pilot Shared Task. *the Generation Challenges Session at the 13th European Workshop on Natural Language Generation,*, pages 242–249.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the Preposition and Determiner Error Correction Shared Task. *the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62.

Vidas Daudaravicius, Rafael E. Banchs, Elena Volodina, and Courtney Napoles. 2016. A report on the automatic evaluation of scientific writing shared task. In *Proceedings of the Eleventh Workshop on Innovative Use of NLP for Building Educational Applications*, San Diego, CA, USA, June. Association for Computational Linguistics.

Vidas Daudaravicius. 2015. Automated Evaluation of Scientific Writing: AESW Shared Task Proposal. *Silver Sponsor*, pages 56–63.

Thorsten Joachims. 1999. Making large scale svm learning practical. Technical report, Universität Dortmund.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - ACL '03*, 1:423–430.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. Automated grammatical error detection for language learners. *Synthesis lectures on human language technologies*, 3(1):1–134.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question answer classification. In *Annual meeting-association for computational linguistics*, volume 45, page 776.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic eing. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04*, pages 335–es.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *EACL*, volume 113, page 24.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 253–262. ACM.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2014. *Shared Task on Grammatical Error Correction*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 180–189. Association for Computational Linguistics.