RGCL at SemEval-2020 Task 6: Neural Approaches to Definition Extraction

Tharindu Ranasinghe[1], **Alistair Plum**[1], **Constantin Orăsan**[2], **Ruslan Mitkov**[1]
[1]Research Group in Computational Linguistics, University of Wolverhampton, UK
[2]Centre for Translation Studies, University of Surrey, UK
{tharindu.ranasinghe, a.j.plum, r.mitkov}@wlv.ac.uk
c.orasan@surrey.ac.uk

Abstract

This paper presents the RGCL team submission to SemEval 2020 Task 6: DeftEval, subtasks 1 and 2. The system classifies definitions at the sentence and token levels. It utilises state-of-the-art neural network architectures, which have some task-specific adaptations, including an automatically extended training set. Overall, the approach achieves acceptable evaluation scores, while maintaining flexibility in architecture selection.

1 Introduction

Definition Extraction refers to the task in Natural Language Processing (NLP) of detecting and extracting a *term* and its *definition* in different types of text. A common use of automatic definition extraction is to help building dictionaries (Kobyliński and Przepiórkowski, 2008), but it can be employed for many other applications. For example, ontology building can benefit from methods that extract definitions (Hearst, 1992; Malaisé et al., 2007), whilst the fields of definition extraction and information extraction can employ similar methodologies. It is therefore normal that there is growing interest in the task of definition extraction.

This paper describes our system that participated in two of the three subtasks of Task 6 at SemEval 2020 (DeftEval), a shared task focused on definition extraction from a specialised corpus. Our method employs state-of-the-art neural architectures in combination with automatic methods which extend and clean the provided dataset.

The remaining parts of this paper are structured as follows. First, we present related work in the area of definition extraction and the related field of relation extraction (Section 2). The three subtasks and the dataset provided by the task organisers are described in Section 3. Next, we describe our system (Section 4), followed by the results of the evaluation (Section 5) and a final conclusion (Section 6).

2 Related Work

The first efforts related to definition extraction happened in the field of hypernym extraction, where relations that usually indicate a definition were also dealt with. This includes the *X is a type of Y* relation, such as *salmon is a type of fish*, where *salmon* is a *hyponym* of *fish*. Notable work includes Hearst (1992), who automatically extracts hyponyms from large amounts of unstructured text using lexico-syntactic patterns. Inspired by this approach, Malaisé et al. (2004) describe a similar method to mine definitions in French, which are then classified in terms of their semantic relations, limited to the *hypernymy - synonymy* relation. The approach is also used for building ontologies (Malaisé et al., 2007).

The importance of the semantic relations between words for pattern-based approaches to definition extraction is highlighted in (Sierra et al., 2008). Here, the authors describe and explain definitional verbal patterns in Spanish, which they also propose to use for mining definitions. The proposed system is further presented in Alarcón et al. (2009) and is aimed at Spanish technical texts. The system uses the aforementioned verbal patterns, as well as corresponding tense and distance restrictions, in order to extract a set of candidate terms and their definitions. Once extracted, the system applies some filtering rules

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 717–723
Barcelona, Spain (Online), December 12, 2020.

and a decision tree to further analyse the candidates. Finally, the results are ranked using heuristic rules. All aspects of the system were developed by analysing the Institut Universitari de Lingüística Aplicada technical corpus in Spanish, which is also used for evaluation.

Machine learning algorithms have also been used for definition extraction. Gaudio and Branco (2009) describe an approach that is said to be language independent and test it with decision trees and Random Forest, as well as Naïve Bayes, k-Nearest Neighbour and Support Vector Machines using different sampling techniques to varying degrees of success. Kobyliński and Przepiórkowski (2008) process Polish texts and use Balanced Random Forests, which bootstrap equal sets of positive and negative training examples to the classifier, as opposed to a larger group of unequal sets of training examples. Overall, while the approach is said to increase run time, it does bring minor increases in performance with some fine-tuning.

Most recently, Spala et al. (2019) have created DEFT, a corpus for definition extraction from unstructured and semi-structured texts. Citing some of the pattern-based approaches also mentioned here, the authors argue that definitions have been well-defined and not necessarily representative of natural language. Therefore, a new corpus is presented that is said to more accurately represent natural language, and includes more messy examples of definitions. Parts of the DEFT corpus make up the dataset for this shared task, which is described in more detail in the following section.

3 Subtasks and Dataset

The DeftEval shared task is split into three subtasks. The first is Sentence Classification, where the task is to predict whether a given sentence contains a definition. Subtask 2 is a sequence labelling task, which includes requires participants to assign BIO tags to indicate which tokens in a sentence belong to terms and their definitions. Furthermore, the BIO tags are fine-grained, denoting whether terms and definitions are *primary*, *secondary* (the second time a term or definition has been seen in a text), *referential* or *ordered* (multiple terms that have inseparable definitions). The final subtask is Relation Classification which requires to classify the relation between terms and their definitions. Included are the tags *direct* and *indirect* definitions (links term or referential term to definition, respectively), *supplements* (links indirect to direct definition), *refers-to* (links referential term/definition to term/definition) and *AKA* (links alias term to term).

The corpus provided by the organisers is made up of parts of the DEFT corpus described in Spala et al. (2019). This corpus has been compiled specifically for definition extraction tasks and is made up of legal contracts (2.443 sentences) and textbook data (21.303 sentences). Citing a growing need for definition extraction corpora, the creators also developed an annotation scheme that is specific to the task of definition extraction.

4 Methodology

In this section we present the different approaches we employed for each subtask. The overall approach is based on a neural network architecture, but each subtask requires different methods of preprocessing the data, as well as task-specific tweaks to the data and architecture. Our implementation has been made available on Github.[1]

4.1 Sentence Classification

This section describes the methodology employed for *Subtask 1: Sentence Classification*, as well as the experiments carried out in order to boost performance. We first present the methods used to process and extend the data, followed by a description of the main neural network architecture employed.

4.1.1 Data Processing and Cleaning

We first used the data converting python script that the organisers provided to convert the deft corpus in to classification instances. After that we concatenated all the files in the training folder in to single file and used it for training purposes while the concatenated file from the dev folder is used for evaluation

[1] https://github.com/tharindudr/defteval

purposes. As the Sentence classification task required only to predict 1 (contains a definition) or 0 (does not contain a definition) it was feasible to perform some simple cleaning to increase the classification performance without causing any side effects. Upon analysis of the data we found that many sentences had some kind of numbering at the beginning, such as in the following example:

> *41. The evolution of various life forms on Earth can be summarized in a phylogenetic tree ([link])*

Using a simple regular expression to match numbers and a punctuation mark at the beginning of a sentence, we removed these character strings across all sets. We used the same approach for finding and deleting character strings such as *([link])*, which have been inserted by the task organisers to replace actual links to websites (see also the above example). In cases where the link replacement formed part of the sentence we did not perform a deletion:

> *Examples of some neutral atoms and their electron configurations are shown in [link].*

This decision was made as it would otherwise leave sentences incomplete. After comparing the performance of our algorithm on both cleaned and uncleaned text we observed a marginal increase of 0.01 across all evaluation metrics using on the best performing architecture. Other than this we did not carry out any additional cleaning. This was also due to the fact that we use BERT embeddings, making it unnecessary to remove any other characters, as it includes vectors for most characters.

4.1.2 Data Augmentation

In order to improve the performance of our classification we extend the training set automatically. To achieve this, the sequence labelling part of the system (described in Section 4.2) was used to detect terms in the training data. Where possible, we extracted the first sentence of the corresponding Wikipedia articles for these terms by scraping Wikipedia. This is due to the fact that the first sentence usually defines the term or item that the article is about. However, the approach had little impact on the performance of the system, trading increases in precision for decreases in recall and decreasing the F1-score by about 0.02. What we learned is since the data augmentation process is completely automated and not manually checked it introduces a certin level of noise to the dataset which result in decreasing the performance.

4.1.3 System Architecture

In order to determine the most suitable system architecture for the sentence classification task, we experimented with three different neural architectures: Convolutional Neural Network (CNN) (Kim, 2014), Recurrent Neural Network (RNN) (Cui et al., 2018) and Transformer (Devlin et al., 2018). After running various configurations, we found the Transformer architecture to perform best.

With the introduction of BERT (Devlin et al., 2018) transformer architectures have shown a massive success in a wide range of NLP tasks. Transformer architectures have been trained on general tasks like language modelling and then fine-tuned for classification tasks (Sun et al., 2019; Ranasinghe et al., 2019b).

Transformer models take an input of a sequence and output the representation of the sequence. The sequence has one or two segments that the first token of the sequence is always [CLS] which contains the special classification embedding and another special token [SEP] is used for separating segments.

For text classification tasks, transformer models take the final hidden state $\mathbf{h}$ of the first token [CLS] as the representation of the whole sequence (Sun et al., 2019). The [CLS] token was then fed in to a simple softmax classifier to predict the label of the whole sentence: whether it contains a definition or not.

We fine-tuned all the parameters from transformer models as well as the softmax classifier jointly by maximising the log-probability of the correct label. For training the model, we used a batch-size of eight, Adam optimiser (Kingma and Ba, 2014) with learning rate $2\mathrm{e}{-5}$, and a linear learning rate warm-up over 10% of the training data. The models were trained using only training data. Furthermore, they were evaluated while training using an evaluation set that had one fifth of the rows in training data. We performed early stopping if the evaluation loss did not improve over ten evaluation rounds. All the models were trained for three epochs. We experimented with several transformer architectures like BERT (Devlin

et al., 2018), XLNet (Yang et al., 2019), XLM (Conneau et al., 2019), RoBERTa (Liu et al., 2019) and DistilBERT (Sanh et al., 2019). We used the HuggingFace's implementation of the transformer models (Wolf et al., 2019) and the pre-trained models available in the HuggingFace's model repository[2].

4.2 Sequence Labelling

This section describes the experiments we conducted for Subtask 2: *Sequence Labelling*. We first present the data processing methods used, followed by the neural network architecture employed. Due to the structure of the data and the way the annotations had to be made (CoNLL-like format) and evaluated, no cleaning was performed for this task.

4.2.1 Data Processing and Augmentation

As a preliminary step, we concatenated all the files from the train folder in Deft corpus to a single file and used it as the training file. Similarly we concatenated all the files from the dev folder in Deft corpus to a single file and used it for evaluation purposes.

For this subtask also we experimented with data augmentation techniques. We tried a similar approach as before, but with a bootstrapping focus: We used the classifier trained for this task to predict terms and extracted the first sentence from each corresponding Wikipedia article. Exploiting the structure of Wikipedia again, we simply automatically labelled the term in the corresponding sentence, therefore providing extra examples of the terms being used in a sentence. However, like in the previous case this step did not improve our results due to the noise it introduces. We also assume that the added terms were always mentioned at the beginning of a sentence, therefore adding positional bias to the classifier.

4.2.2 System Architecture

We experimented with three different neural network architectures for the sequence labelling task: LSTM-CRF (Lample et al., 2016), Stack-LSTM (Lample et al., 2016) and Transformer (Devlin et al., 2018). In this task we also found that the Transformer architecture performs best.

Transformer architectures have proved effective in NER tasks (Devlin et al., 2018), which are also sequence labelling tasks. In light of this, in this subtask, we implemented the approach suggested in the first transformers paper - BERT (Devlin et al., 2018): transformer model combined with a token-level classifier. After processing the sentence through the transformer model each word gets a vector representation. We used this vector representation as the input to the token-level classifier over the label set available for subtask 2. The token-level classifier consists of a dropout (Srivastava et al., 2014) and a linear classifier. We fine-tuned all the parameters from transformer models as well as the token-level classifier jointly by maximising the log-probability of the correct label.

For training the model, we used a batch-size of eight, Adam optimiser (Kingma and Ba, 2014) with learning rate $1e{-}5$, and a linear learning rate warm-up over 6% of the training data. The models were trained using only training data. Furthermore, they were evaluated while training using an evaluation set that had one fifth of the rows in training data. Similar to the subtask 1, we performed early stopping if the evaluation loss did not improve over ten evaluation rounds. All the models were trained for three epochs. We experimented with several transformer architectures: BERT (Devlin et al., 2018), XLNet (Yang et al., 2019), XLM (Conneau et al., 2019), RoBERTa (Liu et al., 2019) and DistilBERT (Sanh et al., 2019). We used the HuggingFace *TokenClassification* interface (Wolf et al., 2019) and the pre-trained models available in the HuggingFace model repository[3].

We also experimented with adding a Conditional Random Field (CRF) layer (Zheng et al., 2015) after the output of the Transformer. However evaluation of several configurations showed that adding the CRF layer does not improve the results. Therefore, we did not pursue these experiments any further.

5 Evaluation

In this section we present the evaluation results that were obtained during testing. We also provide a brief look at the final submission results of the shared task.

[2] https://huggingface.co/models
[3] https://huggingface.co/models

5.1 Sentence Classification Results

Table 1 shows the evaluation of the different architectures we developed for the sentence classification task using the development set. We have also included baseline results which was performed using a Naive Bayes bag of words approach. It is clear that, while marginal, XLNet performs best overall. Interestingly, we compared BERT-Large against XLNet-Base, meaning that our best architecture was much less resource intensive to run.

For the final task evaluation using the test set, we achieved an F1-Macro score of 0.7885, placing us 25th out of 56 participants. Compared to our evaluation results, this is a relatively high loss. We assume that our model has been largely overfitted in to the training set we used.

	Not Definition			**Definition**			**Weighted Average**			
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1 Macro**
CNN	0.78	0.73	0.72	0.76	0.71	0.75	0.74	0.77	0.74	0.76
RNN-BILSTM	0.76	0.71	0.74	0.68	0.74	0.72	0.75	0.72	0.73	0.75
BERT	0.90	0.88	0.89	0.81	0.79	0.80	0.86	0.86	0.86	0.84
XLNet	0.91	0.90	0.90	0.82	0.80	0.81	0.87	0.88	0.87	**0.86**
Baseline	0.89	0.54	0.68	0.49	0.87	0.63	0.66	0.68	0.66	0.67

Table 1: Results for Subtask 1 For each model, Precision (P), Recall (R), and F1 are reported on all classes, and weighted averages. Macro-F1 is also listed (best in bold).

5.2 Sequence Labelling Results

Table 2 shows the evaluation results for the different architectures we tested for the Sequence Labelling task. As before, we see XLNet with the best results, and again see that the less resource intense base version is almost on par with the large version. It should also be noted that the best results were achieved with shortened maximum sequence lengths, down from 128 to 64.

In the official evaluation on the test set we ranked 28th of 51 with an F1-score of 0.4872. This shows a significant drop in performance, possibly due to overfitting.

Model	**P**	**R**	**F1**
BERT	0.71	0.74	0.73
ROBERTa	0.67	0.70	0.69
XLNet - Base	0.71	0.75	0.73
XLNet - Large	**0.72**	**0.76**	**0.74**

Table 2: Results for Subtask 2 For each model, Precision (P), Recall (R), and F1 are reported overall (best in bold).

6 Conclusion

We have presented the system the RGCL team has prepared for the SemEval-2020 Task 12. The design of the system allows for easy switching of different architectures to accommodate the needs of the task at hand. For this task, we have shown the Transformer architecture using XLNet is the most successful when working with limited resources. It has also been shown that data augmentation techniques we experimented, while not detrimental to overall performance, do not necessarily improve performance. In a shared task setting, the effect of the extended data from Wikipedia was not useful, however, for a wider approach with higher recall, this could be more helpful.

We also tried to participate in the final subtask, *Relation Classification*. However, due to time constraints, we were not able to achieve a valid submission for the this subtask. We approached it as a sequence pair classification task and employed a Siamese Neural Network which was shown to perform well in sequence pair classification tasks (Mueller and Thyagarajan, 2016; Ranasinghe et al., 2019a). The

architecture we employed is similar to the architecture presented in (Reimers and Gurevych, 2019). When two sequences have a relation, we extracted the sequences and provided them as the input for the Siamese transformer architecture. Then we used the objective function suggested as classification objective function in (Reimers and Gurevych, 2019) and optimised the cross-entropy loss. Due to the complexity of this task, we managed to run only a baseline of the proposed architecture which achieved very low evaluation scores on the development data. Therefore, we did not have a submission for this task and do not present any results here. In future, we hope to carry out further experiments with Siamese transformer architectures for relation classification tasks.

Going forth, we also wish to use this system for further tasks across further languages. While we may not achieve the best performance, the system utilises realistic system resources and is therefore very versatile. This is particularly with regard to the first subtask, where the difference to the best team is around 0.09, whereas for subtask two the best team is 0.36 ahead of us, indicating that our system is not competitive. It is possible to extend these experiments to a different domain easily using a pretrained transformer model in that domain given that a corpus similar to deft corpus is available in that domain. For an example, our system should be easily adoptable to biology domain using the BioBERT pretrained transformer model (Lee et al., 2019) and a deft corpus like corpus on biology domain.

References

Rodrigo Alarcón, Gerardo Sierra, and Carme Bach. 2009. Description and evaluation of a definition extraction system for spanish language. In *WDE '09 Proceedings of the 1st Workshop on Definition Extraction*, pages 7–13.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Jianjing Cui, Jun Long, Erxue Min, Qiang Liu, and Qian Li. 2018. Comparative study of cnn and rnn for deep learning based intrusion detection system. In Xingming Sun, Zhaoqing Pan, and Elisa Bertino, editors, *Cloud Computing and Security*, pages 159–170, Cham. Springer International Publishing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Rosa Del Gaudio and António Branco. 2009. Language independent system for definition extraction: first results using learning algorithms. In *WDE '09 Proceedings of the 1st Workshop on Definition Extraction*, pages 33–39.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Łukasz Kobyliński and Adam Przepiórkowski. 2008. Definition extraction with balanced random forests. In *International Conference on Natural Language Processing*, pages 237–247.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 09.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Véronique Malaisé, Pierre Zweigenbaum, and Bruno Bachimont. 2004. Detecting semantic relations between terms in definitions. *CompuTerm 2004 - 3rd International Workshop on Computational Terminology*, pages 55–62.

Véronique Malaisé, Pierre Zweigenbaum, and Bruno Bachimont. 2007. Mining defining contexts to help structuring differential ontologies. *Application-driven terminology engineering*, 2:19.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2786–2792. AAAI Press.

Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. 2019a. Semantic textual similarity with Siamese neural networks. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1004–1011, Varna, Bulgaria, September. INCOMA Ltd.

Tharindu Ranasinghe, Marcos Zampieri, and Hansi Hettiarachchi. 2019b. Brums at hasoc 2019: Deep learning models for multilingual hate speech and offensive language identification. *In Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation (December 2019)*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Gerardo Sierra, Rodrigo Alarcón, César Aguilar, and Carme Bach. 2008. Definitional verbal patterns for semantic relation extraction. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, 14(1):74–98.

Sasha Spala, Nicholas A. Miller, Yiming Yang, Franck Dernoncourt, and Carl Dockhorn. 2019. DEFT: A corpus for definition extraction in free- and semi-structured text. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 124–131.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In Maosong Sun, Xuanjing Huang, Heng Ji, Zhiyuan Liu, and Yang Liu, editors, *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. 2015. Conditional random fields as recurrent neural networks. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, page 1529–1537, USA. IEEE Computer Society.