ACNLP at SemEval-2020 Task 6: A Supervised Approach for Definition Extraction

Fabien Caspani Pirashanth Ratnamogan Mathis Linger Mhamed Hajaiej
BNP Paribas, France
{fabien.caspani, pirashanth.ratnamogan, mathis.linger, mhamed.hajaiej}@bnpparibas.com

Abstract

We describe our contribution to two of the subtasks of SemEval 2020 Task 6, *DeftEval: Extracting term-definition pairs in free text*. The system for *Subtask 1: Sentence Classification* is based on a transformer architecture where we use transfer learning to fine-tune a pretrained model on the downstream task, and the one for *Subtask 3: Relation Classification* uses a Random Forest classifier with handcrafted dedicated features. Our systems respectively achieve 0.830 and 0.994 F_1-scores on the official test set, and we believe that the insights derived from our study are potentially relevant to help advance the research on definition extraction.

1 Introduction

SemEval 2020 Task 6 (Spala et al., 2020) is a definition extraction problem divided into three different subtasks, in which the objective is to find the term-definition pairs in free text, possibly spanning across sentences boundaries. In *Subtask 1* the system classifies a sentence as containing a definition or not. *Subtask 2* consists of labeling each token with their corresponding BIO tags. Finally, in *Subtask 3* the system pairs tagged entities, and labels the relation existing between them. The Definition Extraction from Texts (DEFT) corpus (Spala et al., 2019) is the dataset used for this task, and consists of human-annotated data from textbooks on a variety of subjects.

Definition extraction methods in the research literature fall into one of the following categories. The earlier research on the subject applies rules-based approaches, such as Klavans and Muresan (2001) whose system extracts definitions from medical publications. Similar approaches can also be found in Cui et al. (2004; 2005) as well as in Westerhout and Monachesi (2007). These methods find definitions by locating words such as *is called*, *means* or *is*, and using grammatical rules on top of that. They generally suffer from low recall, but this issue has been addressed by features-based methods. Fahmi and Bouma (2006) have trained a sentence classifier with features based on bag-of-words and n-grams, position of the sentences in the text and syntactic information. Westerhout (2009) expands the set of features with additional linguistic and structural information, and uses a hybrid approach between a rules-based system and a machine learning classifier.

The above approaches do not generalize well to new domains, as rules and handcrafted features are often relevant to specific tasks and time-consuming to create, hence the development of Machine Learning and Deep Learning models. Li et al. (2016) use a Long Short-Term Memory neural (LSTM) network classifier, where features are automatically created from raw input sentences and part-of-speech sequences. Contextual words embeddings depend on the adjacent text, and are pretrained in an unsupervised manner to automatically learn semantic concepts before being used on different downstream tasks (Akbik et al., 2018). Lin and Lu (2018) apply transfer learning to their downstream token classification problem, which consists in fine-tuning the context string embeddings to a particular task without having to learn them from scratch. Transformer-based architectures have been recently used with transfer learning: Alt et al. (2019) solve a relation extraction problem with GPT (Radford et al., 2018) embeddings, and Pouran Ben Veyseh et al. (2019) use BERT (Devlin et al., 2018) embeddings for a definition extraction task.

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 479–486
Barcelona, Spain (Online), December 12, 2020.

2 Systems Description

In this section, we describe the transformers architectures used for *Subtask 1*, as well as the Random Forest classifier trained with features based on entities tags and positional information to solve *Subtask 3*.

2.1 Subtask 1: Sentence Classification

We model this subtask as a binary classification problem, where the system determines whether a sentence contains or not a definition. We use transformers architecture to solve this problem: BERT, and RoBERTA for a Robustly Optimized BERT Pretraining Approach (Liu et al., 2019). More particularly, we use the PyTorch-Transformers implementations of BERT and RoBERTa from Hugging Face (Wolf et al., 2019).

Sentence Classifier We use pretrained RoBERTa word embeddings of the input sentence, as well as fully-connected and softmax output layers to fine-tune the model on the downstream sentence classification task. Moreover, similarly to what is done in the original BERT paper (Devlin et al., 2018), we concatenate the last four hidden layers of the RoBERTa model before going through the fully connected layers. To better generalize the prediction of our model, we also consider different methods. Stochastic Weight Averaging that was first introduced in Izmailov et al. (2018), consists in combining weights of the same network at different stages of training. Similarly to what is done in Zhang et al. (2015), we use data augmentation by replacing words selected at random in the sentence with one of their possible synonyms. This list of possible choices for each word, called a Thesaurus, is based on WordNet synonyms. Finally, label smoothing (Szegedy et al., 2015) changes hard targets to soft ones in order to less penalize incorrect predictions, and to potentially improve robustness of the model.

Token Classifier We train a token classifier with the labels provided for *Subtask 2* in order to complement the predictions of the sentence classification model. During inference, we say that a sentence contains a definition if it contains tokens classified as *B-Definition* or *I-Definition*. When training the model, we group sentences into paragraphs and use an architecture similar to the one described for the above classifier, except that it is fine-tuned on a downstream token classification task.

Models Combination Every sentence has one prediction from the sentence classifier, and another one from the token classification model. If both models agree that a sentence contains or does not contain a definition, we keep the prediction as such. Sentences for which the models classifications differ are ranked in decreasing order of predictions scores, and for each model we classify the top half of the sentences as having a definition. The intuition behind the addition of a token classifier is that the use of more granular information during training (label at the token level instead of the sentence level only) may lead to better predictions for *Subtask 1*.

2.2 Subtask 3: Relation Classification

We approach this subtask as a multi-label classification problem, where the system determines which *entity B* is linked to *entity A*, and the relation that exists between them. We use a Random Forest classifier (Breiman, 2001) to solve this problem. We now describe the features used and the post-processing steps.

Features and Labels Each entity (let us call it A) can be linked to another entity (let us call it B) in the same paragraph. For this pair of entities, either they are linked and the target label in the classification problem corresponds to the relation between them, or they are not linked and we create a dummy target label corresponding to the fact that there is no relation between them. We have the following features for this pair:

- *Labels from Subtask 2*: Encodes the tags of entities A and B, as well as the tags of other entities in the same paragraph.
- *Relative positions*: Specifies whether entity A precedes B with or without other entities between them, and conversely.
- *Sentence related features*: Specifies whether entities A and B are in the same sentence or not, in addition to the number of words and entities in their respective sentences.

- *Paragraph related features*: Counts the number of words, sentences and entities in the paragraph.
- *Subject related feature*: Encodes the subject of the textbook the paragraph is extracted from.

Post-Processing Steps Once our classifier gives its predictions, we make sure that a *Definition* is at most linked to one *Term* in a paragraph. If it is not the case, we apply a rule specifying that a *Definition* is linked to the closest *Term* in the paragraph. Also, because our model only allows entity *A* to be linked to at most one entity *B*, we add rules to deal with duplicate sentences in the same paragraph. For instance, the same *Referential-Definition* can be linked to the previous *Definition* as well as the *Term* in the same sentence.

3 Experimental Setup

3.1 Subtask 1: Sentence Classification

Data We use the training, development and test datasets provided for SemEval 2020 Task 6, that consist of annotated text from textbooks on 7 different subjects. Each text data file is divided into several paragraphs, that are further divided into sentences. In total, there are 17,819 sentences (5,782 of them containing a definition) in the train dataset, and 872 sentences (284 of them containing a definition) in the development dataset. In the test dataset on which are evaluated our submissions, there is a total of 859 sentences (279 of them having a definition). We use the BIO (standing for *Begin*, *Interior* and *Out*) tags provided to train our token classification model. The publicly available *BERT-large-cased* and *RoBERTa-large* model have been pretrained on some of the following datasets: *BookCorpus*, *CC-NEWS*, *OpenWebText* and *Stories*.

Parameter Settings Using 5-fold cross-validation on the train dataset, we try different combinations of hyperparameters to find the ones giving the best mean F_1 score across the folds. For each iteration, we use 95% of the train dataset as our train data, and the remaining 5% for validation. Cross-validations are not performed across the 20 possible folds because fine-tuning the models on our downstream tasks is very time-consuming, and folds are created by sampling (without replacement) paragraphs from the train dataset.

The BERT and RoBERTa models fine-tuned on the downstream sentence classification task have the following configuration. It is trained over 6 epochs and we use the Adam optimizer (Kingma and Ba, 2014) with learning rate of $1e^{-5}$. The maximum length of a sequence after tokenization is 128 and the batch size is 48. Two fully-connected layers with a dropout rate of 0.1 are used to output the predictions. When data augmentation (DA) is used, the size of the training data is doubled. During this process, a sentence is sampled (with replacement) from the dataset, and at least 8 of its words picked at random are replaced with synonyms. When Stochastic Weight Averaging (SWA) is used, the weights of the models saved during the last 2 epochs of training are averaged.

The BERT and RoBERTa models fine-tuned on the downstream token classification task have the same configuration as the models described above, except from the following: the learning rate is $3e^{-5}$, the model is trained over 5 epochs, data augmentation is not used and the weights of the models saved during every epoch of training are averaged. Also, the maximum length of a sequence after tokenization is 256 (paragraphs inputs contain more words than sentences) and the batch size is 32.

Experiments for the token classifier We are interested in a model that correctly classifies definitions related tokens to improve the predictions of the sentence classifier. In table 1 we present the results of our experiments on the development dataset: the RoBERTa model with Stochastic Weight Averaging performs the best for the *B-Definition* and *I-Definition* classes, and this is the one being selected.

Experiments for the sentence classifier Table 2 shows the results of different architectures on the development dataset. They are compared against a baseline model consisting of a Linear SVC model trained on data pre-processed with a count vectorizer, and we see that it is largely outperformed by transformers architectures.

In the experiments with the sentence classifier only, the RoBERTa model with Stochastic Weight Averaging outperforms the others on the development dataset, followed by the RoBERTa model with label smoothing (LS). To combine the predictions of the sentence and token classifiers, we follow the

Model	Additions	B-Def Precision	B-Def Recall	B-Def F_1	I-Def Precision	I-Def Recall	I-Def F_1
BERT	-	0.617	0.688	0.651	0.772	0.782	0.777
RoBERTa	-	0.685	0.664	0.674	0.804	0.772	0.788
	SWA	0.669	0.685	**0.677**	0.800	0.778	**0.789**

Table 1: Results of experiments on the development dataset for the token classifier

procedure detailed in section 2.1, but we see that it does not necessarily lead to better predictions. Again, the RoBERTa model with Stochastic Weight Averaging gives the best results, followed closely by the RoBERTa model without additional methods. However, the results of our cross-validations on the training dataset indicate that the RoBERTa model with data augmentation, label smoothing and Stochastic Weight Averaging performs better than the results shown on the development dataset only. Given the limited number of submissions allowed, we decide to use the latter model during the competition.

Experiment	Model	Additions	Precision	Recall	F_1
Sentence classifier only	Baseline	-	0.722	0.571	0.638
	BERT	-	0.802	0.817	0.809
	RoBERTa	-	0.822	0.813	0.817
		DA	0.799	0.813	0.806
		LS	0.799	0.845	0.822
		SWA	0.829	0.835	**0.832**
		DA + LS	0.825	0.813	0.819
		DA + SWA	0.808	0.817	0.812
		LS + SWA	0.805	0.817	0.811
		DA + LS + SWA	0.796	0.827	0.811
Sentence classifier with token classifier	BERT	-	0.803	0.849	0.825
	RoBERTa	-	0.801	0.853	0.826
		DA	0.799	0.845	0.822
		LS	0.789	0.849	0.818
		SWA	0.812	0.853	**0.832**
		DA + LS	0.799	0.831	0.815
		DA + SWA	0.802	0.845	0.823
		LS + SWA	0.795	0.838	0.816
		DA + LS + SWA	0.785	0.838	0.810

Table 2: Results of experiments on the development dataset for *Subtask 1*

3.2 Subtask 3: Relation Classification

Data We use the training, development and test datasets provided for SemEval 2020 Task 6. There is a total of 7,110 entities in the training dataset, 335 entities in the development dataset, and 358 entities in the test dataset. To train our classifier we use labels from *Subtask 2* (BIO tags), in addition to the IDs of paired entities and the relation label. For illustration, in the sentence *"Straightening a limb after flexion is an example of extension."*, entity T90 corresponds to *"Straightening a limb after flexion"* whose tokens are tagged as *Definitions*, entity T89 to the *Term "extension"*, and their relation is labeled as *Direct-Defines*.

Parameter Settings Using 20-fold cross-validation on the train dataset, we perform hyperparameter search with Optuna (Akiba et al., 2019) to find the values giving the best mean macro-F_1 score across the folds. For each iteration, we use 95% of the train dataset as our train data, and the remaining 5% for validation.

We use the Scikit-learn (Pedregosa et al., 2011) implementation of Random Forest with the following

parameter values: 546 trees, the maximum depth of trees is 59, the minimum number of samples required to be at a leaf node is 3 and the minimum number of samples required to split an internal node is 26. We assign weights to the different classes in the loss function so that their contribution is inversely proportional to class frequencies.

Experiments Table 3 shows the results of experiments for *Subtask 3* on the development dataset. We see that the use of the post-processing steps described in section 2.2 greatly improves the predictions of the Random Forest classifier alone.

Model	Post-Processing	Precision	Recall	F_1
Random Forest	-	0.913	0.834	0.856
	Yes	0.946	0.936	**0.941**

Table 3: Results of experiments on the development dataset for *Subtask 3*

4 Results & Discussion

Subtask 1 is evaluated with the F_1-score on the positive class. Table 4 summarizes our results on the test dataset with three different approaches. In a first approach, we train the model described in section 2.1 on the training plus development datasets for the downstream sentence classification task. The F_1-score of 0.783 on the positive class is less than the score of 0.811 obtained during the experiments with these model parameters. To improve this result, we use a 5-fold cross-validation on the training plus development datasets, and create an ensemble model by averaging the predictions of the models trained on the different folds. This second approach results in an improved F_1-score of 0.809. In a third approach, we adopt the same strategy to train an ensemble model on the downstream token classification task, and combine its predictions with the ones of the above classifier to substantially increase the F_1-score to 0.830. Our best submission helps us achieve 6th place in this subtask of the SemEval competition.

Submission	Model	Additions	Precision	Recall	F_1
Sentence classifier only	RoBERTa	DA + LS + SWA	0.809	0.760	0.783
	RoBERTa with ensembling	DA + LS + SWA	0.798	0.821	0.809
Sentence classifier with token classifier	RoBERTa with ensembling	DA + LS + SWA	0.819	0.842	**0.830**

Table 4: *Subtask 1* results on the SemEval test dataset

After analysis of the results on the development dataset, we see that our model performs differently across subjects: *Sociology* has the best F_1-score of 0.921 and *Physics* the worst score of 0.667. One of the reasons that might explain the poor performance of our model on the *Physics* related data is that definitions are often used to describe figures (i.e. *"The slender arrow represents a ray of unpolarized light."*), which may not be the case in other subjects. We also see that our model struggles when the term and definition are not in the same sentence: the sentence *"These rules are special cases of the laws of conservation of charge and conservation of energy."* is labeled has containing a definition and its related term *"Kirchhoff's rules"* is in the next sentence, but our model does not classify it as having a definition. However, it classifies the very similar sentence *"Kirchhoff's rules, special applications of the laws of conservation of charge and energy, can be used to analyze it."* has containing a definition, even though it is not labeled as having one. *"Comte named the scientific study of social patterns positivism."* is another example of sentence not labeled has having a definition, but that the model classifies as having one. These last two examples show that the annotation process may be inconsistent, making it difficult for our model to generalize well.

The evaluation metrics chosen by the organizers for *Subtask 3* is the macro-average F_1-score for the five following relations: *AKA*, *Direct-Defines*, *Indirect-Defines*, *Refers-To* and *Supplements*. The classes in the

data are very imbalanced, and the distribution of examples between them in the test set is the following: 37 *AKA*, 294 *Direct-Defines*, 13 *Indirect-Defines*, 1 *Supplements* and 13 *Refers-To* relations. Table 5 summarizes our results on the test dataset with the Random Forest classifier only, and with the addition of the post-processing steps. The classifier achieves a macro-average F_1-score of 0.902, increased to 0.994 with the post-processing steps, a substantial improvement explained in parts by the large impact the F_1-score of the *Refers-To* class has on the overall score. Our last submission helps us achieve the 2nd best score (tied) on the leaderboard of this subtask.

Model	Post-Processing	Precision	Recall	F_1
Random Forest	-	0.983	0.869	0.902
	Yes	0.993	0.996	**0.994**

Table 5: *Subtask 3* results on the SemEval test dataset

The difference in scores between our results on the development and test datasets can be mainly explained by two reasons. First, *Qualifies* relations are easier to classify in the test set because they only involve *Terms* directly surrounding *Qualifiers*, and not *Definitions* that may be further apart. Then, *Refers-To* relations involved in duplicate sentences of a paragraph are more straightforward to handle with rules than other classes such as *AKA* present in the development dataset.

5 Conclusion

Our system used for *Subtask 1* is the combination of a sentence and a token classifier, both based on a transformer architecture. In *Subtask 3* we use a simple Random Forest model built with categorical features carrying the information of the paired entities, supplemented with post-processing rules. Both approaches are supervised and perform relatively well for the subtasks at hands.

Including the knowledge about individual BIO tags may substantially improve the performance of the model for *Subtask 1*, showing that a good token classifier is key to solve a definition extraction problem. With the DEFT corpus, it is sometimes difficult to make good predictions for rare classes, especially when building a token classifier. An idea to improve the models could be to use a data augmentation approach similar to the one used in *Subtask 1* on the examples with elements from the rare classes. One could build models specific to the different rare classes instead of one unique token classifier, or include additional information such as Part-of-Speech tags. Also, pretraining our transformers models on domain-specific text corpus crawled from textbooks is an idea that has been considered but not implemented due to time constraints.

Ideally, in a definition extraction task the sequence labeling and relation extraction steps should be handled by a single model. Our results on the separate subtasks suggest that improvements can be made in this direction, as classification errors of the token classifier has an impact on the relation extraction. Also, the performance of our supervised models heavily relies on the important investments made to tag the entities and relations in the text corpus. Research towards semi-supervised or unsupervised approaches could be fruitful as they do not depend as much on data labeling.

Acknowledgements

We would like to thank the Analytics Consulting team and more particularly Thomas Gilles for giving the resources to work on this task, as well as Bruno Taille for the very insightful discussions. We are also grateful to the DeftEval task organizers for providing the dataset and guidance during the competition.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. *arXiv e-prints*, page arXiv:1907.10902.

Christoph Alt, Marc Hübner, and Leonhard Hennig. 2019. Improving Relation Extraction by Pre-trained Language Representations. *arXiv e-prints*, page arXiv:1906.03088.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Hang Cui, Min-Yen Kan, and Tat-Seng Chua. 2004. Unsupervised learning of soft patterns for generating definitions from online news. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, page 90–99, New York, NY, USA. Association for Computing Machinery.

Hang Cui, Min-Yen Kan, and Tat-Seng Chua. 2005. Generic soft pattern models for definitional question answering. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, page 384–391, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, page arXiv:1810.04805.

Ismail Fahmi and Gosse Bouma. 2006. Learning to identify definitions using syntactic features. In *Proceedings of the Workshop on Learning Structured Information in Natural Language Applications*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging Weights Leads to Wider Optima and Better Generalization. *arXiv e-prints*, page arXiv:1803.05407.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980.

JL Klavans and S Muresan. 2001. Evaluation of the definder system for fully automatic glossary construction. *Proceedings. AMIA Symposium*, page 324—328.

Siliang Li, Bin Xu, and Tong Lee Chung. 2016. Definition extraction with LSTM recurrent neural networks. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data - 15th China National Conference, CCL 2016, and 4th International Symposium, NLP-NABD 2016, Yantai, China, October 15-16, 2016, Proceedings*, volume 10035 of *Lecture Notes in Computer Science*, pages 177–189.

Bill Yuchen Lin and Wei Lu. 2018. Neural adaptation layers for cross-domain named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2012–2022, Brussels, Belgium, October-November. Association for Computational Linguistics.

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 e-prints*, page arXiv:1907.11692.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Amir Pouran Ben Veyseh, Franck Dernoncourt, Dejing Dou, and Thien Huu Nguyen. 2019. A Joint Model for Definition Extraction with Syntactic Connection and Semantic Consistency. *arXiv e-prints*, page arXiv:1911.01678, November.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. `https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/languageunderstandingpaper.pdf`.

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, Florence, Italy. Association for Computational Linguistics.

Sasha Spala, Nicholas Miller, Franck Dernoncourt, and Carl Dockhorn. 2020. SemEval-2020 Task 6: Definition extraction from free text with the DEFT corpus. In *Proceedings of the 14th International Workshop on Semantic Evaluation*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *arXiv e-prints*, page arXiv:1512.00567.

Eline Westerhout and Paola Monachesi. 2007. Extraction of dutch definitory contexts for elearning purposes. *LOT Occasional Series*, 7:219–234.

Eline Westerhout. 2009. Definition extraction using linguistic and structural features. In *Proceedings of the 1st Workshop on Definition Extraction*, WDE '09, page 61–67, USA. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv e-prints*, page arXiv:1910.03771.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. *arXiv e-prints*, page arXiv:1509.01626.