SemDeep-6

Proceedings of the 6th Workshop on Semantic Deep Learning (SemDeep-6)

8 January, 2021
Online

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-950737-19-2

SemDeep-6

Welcome to the 6th Workshop on Semantic Deep Learning (SemDeep-6), held in conjunction with IJCAI 2020. As a series of workshops and a special issue, SemDeep has been aiming to bring together Semantic Web and Deep Learning research as well as industrial communities. It seeks to offer a platform for joining computational linguistics and formal approaches to represent information and knowledge, and thereby opens the discussion for new and innovative application scenarios for neural and symbolic approaches to NLP, such as neural-symbolic reasoning.

SemDeep-6 features a shared task on evaluating meaning representations, the WiC-TSV (Target Sense Verification For Words In Context) challenge, based on a new multi-domain evaluation benchmark. The main difference between WiC-TSV and common WSD task statement is that in WiC-TSV there is no standard sense inventory that systems need to model in full. Each instance in the dataset is associated with a target word and single sense, and therefore systems are not required to model all senses of the target word, but rather only a single sense. The task is to decide if the target word is used in the target sense or not, a binary classification task. Therefore, the task statement of WiC-TSV resembles the usage of automatic tagging in enterprise settings.

In total, this workshop accepted four papers, two for SemDeep and two for WiC-TSV. The main trend, as it has been the norm for previous SemDeep editions, has been the combination of neural-symbolic approaches to language and knowlege management tasks. For instance, Chiarcos et al. (2021) discuss methods for combining embeddings and lexicons, whereas Moreno et al. (2021) present a method for relation extraction and Vandenbussche et al. explore the application of transformer models to Word Sense Disambiguation. Finally, Moreno et al. (2021) also take advantage of language models for their participation to WiC-TSV. In addition, this SemDeep edition had the pleasure to have Michael Spranger as keynote speaker, who gave an invited talk on Logic Tensor Network, as well as an invited talk by Mohammadshahi and Henderson (2020) on their Findings of EMNLP paper on Graph-to-Graph Transformers.

We would like to thank the Program Committee members for their support of this event in form of reviewing and feedback, without whom we would not be able to ensure the overall quality of the workshop.

Dagmar Gromann, Luis Espinosa-Anke, Thierry Declerck, Anna Breit, Jose Camacho-Collados, Mohammad Taher Pilehvar and Artem Revenko.

Co-organizers of SemDeep-6. January 2021

Organisers:

Luis Espinosa-Anke, Cardiff University, UK
Dagmar Gromann, Technical University Dresden, Germany
Thierry Declerck, German Research Centre for Artificial Intelligence (DFKI GmbH), Germany
Anna Breit, Semantic Web Company, Austria
Jose Camacho-Collados, Cardiff University, UK
Mohammad Taher Pilehvar, Iran University of Science and Technology, Iran
Artem Revenko, Semantic Web Company, Austria

Program Committee:

Michael Cochez, RWTH University Aachen, Germany
Artur d'Avila Garcez, City, University of London, London UK
Pascal Hitzler, Kansas State University, Kansas, USA
Wei Hu, Nanjing University, China
John McCrae, Insight Centre for Data Analytics, Galway, Ireland
Rezaul Karim, Fraunhofer Institute for Applied Information Technology FIT, Sankt Augustin, Germany
Jose Moreno, Universite Paul Sabatier, IRIT, Toulouse, France
Tommaso Pasini, Sapienza University of Rome, Rome, Italy
Alessandro Raganato, University of Helsinki, Helsinki, Finland
Simon Razniewksi, Max-Planck-Institute, Germany
Steven Schockaert, Cardiff University, United Kingdom
Michael Spranger, Sony Computer Science Laboratories Inc., Tokyo, Japan
Veronika Thost, IBM, Boston, USA
Arkaitz Zubiaga, University of Warwick, United Kingdom

Invited Speaker:

Michael Spranger, Sony Computer Science Laboratories

Invited Talk

Michael Spranger: Logic Tensor Network - A next generation framework for Neural-Symbolic Computing

Artificial Intelligence has long been characterized by various approaches to intelligence. Some researchers have focussed on symbolic reasoning, while others have had important successes using learning from data. While state-of-the-art learning from data typically use sub-symbolic distributed representations, reasoning is normally useful at a higher level of abstraction with the use of a first-order logic language for knowledge representation. However, this dichotomy may actually be detrimental to progress in the field. Consequently, there has been a growing interest in neural-symbolic integration. In the talk I will present Logic Tensor Networks (LTN) a recently revised neural-symbolic formalism that supports learning and reasoning through the introduction of a many-valued, end-to-end differentiable first-order logic, called Real Logic. The talk will introduce LTN using examples that combine learning and reasoning in areas as diverse as: data clustering, multi-label classification, relational learning, logical reasoning, query answering, semi-supervised learning, regression and learning of embeddings.

Table of Contents

CTLR@WiC-TSV: Target Sense Verification using Marked Inputs and Pre-trained Models

Jose G. Moreno
University of Toulouse
IRIT, UMR 5505 CNRS
F-31000, Toulouse, France
jose.moreno@irit.fr

Elvys Linhares Pontes
University of La Rochelle
L3i
F-17000, La Rochelle, France
elvys.linhares_pontes@univ-lr.fr

Gaël Dias
University of Caen
GREYC, UMR 6072 CNRS
F-14000, Caen, France
gael.dias@unicaen.fr

Abstract

This paper describes the CTRL participation in the Target Sense Verification of the Words in Context challenge (WiC-TSV) at SemDeep-6. Our strategy is based on a simplistic annotation scheme of the target words to later be classified by well-known pre-trained neural models. In particular, the marker allows to include position information to help models to correctly identify the word to disambiguate. Results on the challenge show that our strategy outperforms other participants ($+11,4$ Accuracy points) and strong baselines ($+1,7$ Accuracy points).

1 Introduction

This paper describes the CTLR[1] participation at the Word in Context challenge on the Target Sense Verification (WiC-TSV) task at SemDeep-6. In this challenge, given a target word w within its context participants are asked to solve a binary task organised in three sub-tasks:

- *Sub-task 1* consists in predicting if the target word matches with a given *definition*,
- *Sub-task 2* consists in predicting if the target word matches with a given *set of hypernyms*, and
- *Sub-task 3* consists in predicting if the target word matches with a given couple *definition* and *set of hypernyms*.

Our system is based on a masked neural language model with position information for Word Sense Disambiguation (WSD). Neural language models are recent and powerful resources useful for multiple Natural Language Processing (NLP) tasks (Devlin et al., 2018). However, little effort has been made to perform tasks, where positions represent meaningful information. Regarding this line of research, Baldini Soares et al. (2019) include markers into the learning inputs for the task of relation classification and Boualili et al. (2020) into an information retrieval model. In both cases, the tokens allow the model to carefully identify the targets and to make an informed prediction. Besides these works, we are not aware of any other text-based tasks that have been tackled with this kind of information included into the models. To cover this gap, we propose to use markers to deal with target sense verification task.

The remainder of this paper presents a brief background knowledge in Section 2. Details of our strategy, including input modification and prediction mixing is presented in Section 3. Then, unofficial and official results are presented in Section 4. Finally, conclusions are drawn in Section 5.

2 Background

NLP research has recently been boosted by new ways to use neural networks. Two main groups of neural networks can be distinguished[2] on NLP based on the training model and feature modification.

- First, *classical neural networks* usually use pre-trained embeddings as input and models learn their own weights during training time. Those weights are calculated directly on the target task and integration of new features or resources is intuitive. As an example, please refer to the Figure 1(a) which depicts the model from Zeng et al. (2014) for relation classification. Note that this model uses

[1]University of **C**aen Normandie, University of **T**oulouse, and University of **L**a **R**ochelle team.

[2]We are aware that our classification is arguable. Although this is not an established classification in the field, it seems important for us to make a difference between them as this work tries to introduce well-established concepts from the first group into the second one.

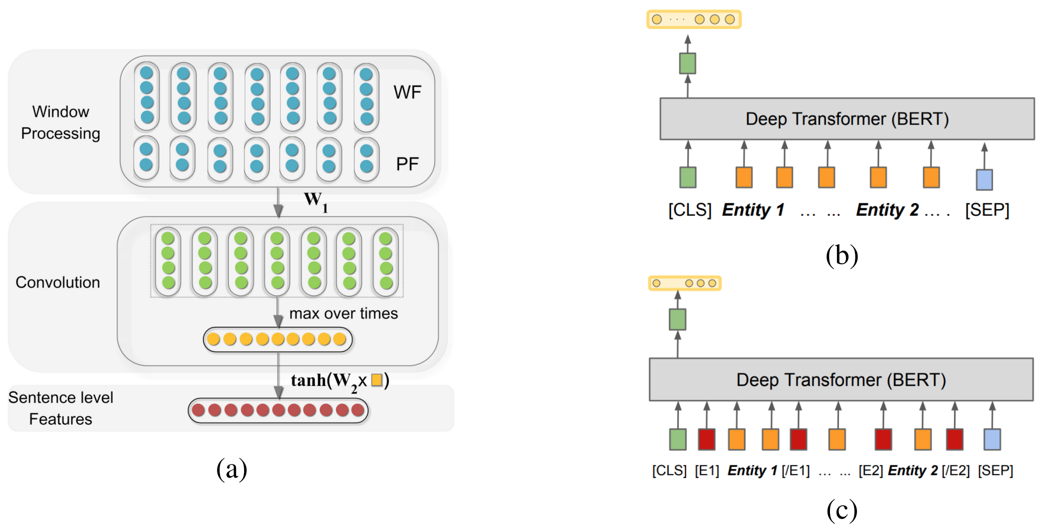

Figure 1: Representation examples for the relation classification problem proposed by Zeng et al. (2014) (a) and Baldini Soares et al. (2019) (b and c).

the positional features (PF in the figure) that enrich the word embeddings (WF in the figure) to better represent the target words in the sentence. In this first group, models tend to use few parameters because embeddings are not fine-tuned. This characteristic does not dramatically impact the model performances.

- The second group of models deals with *neural language models*[3] such as BERT (Devlin et al., 2018). The main difference, w.r.t. the first group, is that the weights of the models are not calculated during the training step of the target task. Instead, they are pre-calculated in an elegant but expensive fashion by using generic tasks that deal with strong initialised models. Then, these models are fine-tuned to adapt their weights to the target task.[4] Figure 1(b) depicts the model from Devlin et al. (2018) for the sentence classification based on BERT. Within the context of neural language models, adding extra features like PF demands re-train of the full model, which is highly expensive and eventually prohibitive. Similarly, re-train is needed if one opt for adding external information as made for recent works such as KNOW+E+E (Peters et al., 2019) or SenseBERT (Levine et al., 2019).

We propose an alternative to mix the best of both worlds by including extra tokens into the input in order to improve prediction without re-training it. To do so, we base our strategy on the introduction of signals to the neural language models as depicted in Figure 1(c) and done by Baldini Soares et al. (2019). Note that in this case the input is modified by introducing extra tokens ([E1], [/E1], [E2], and [/E2] are added based on target words (Baldini Soares et al., 2019)) that help the system to point out the target words. In this work, we mark the target word by modifying the sentence in order to improve performance of BERT for the task of target sense verification.

3 Target Sense Verification

3.1 Problem definition

Given a first sentence with a known target word, a second sentence with a definition, and a set of hypernyms, the target sense verification task consists in defining whether or not the target word in the first sentence corresponds to the definition or/and the set of hypernyms. Note that two sub-problems may be set if only the second sentence or the hypernyms are used. These sub-problems are presented as sub-tasks in the WiC-TSV challenge.

3.2 CTLR method

We implemented a target sense verification system as a simplified version[5] of the architecture proposed by Baldini Soares et al. (2019), namely $BERT_{EM}$. It is based on BERT (Devlin et al., 2018), where an extra layer is added to make the

[3]Some subcategories may exist.

[4]We can image a combination of both, but models that use BERT as embeddings and do not fine-tune BERT weights may be classified in the first group.

[5]We used the *EntityMarkers[CLS]* version.

classification of the sentence representation, i.e. classification is performed using as input the [CLS] token. As reported by Baldini Soares et al. (2019), an important component is the use of mark symbols to identify the entities to classify. In our case, we mark the target word in its context to let the system know where to focus on.

3.3 Pointing-out the target words

Learning the similarities between a couple of sentences (sub-task 1) can easily be addressed with BERT-based models by concatenating the two inputs one after the other one as presented in Equation 1, where S_1 and S_2 are two sentences given as inputs, $t^1_i(i = 1..n)$ are the tokens in S_1, and $t^2_j(j = 1..m)$ are the tokens in S_2. In this case, the model must learn to discriminate the correct definition and also to which of the words in S_1 the definition relates to.

$$\begin{gathered}\mathrm{input}(\mathrm{S}_1, \mathrm{S}_2) = \\ [\mathrm{CLS}] \quad t^1_1 \quad t^1_2 \quad ... \quad t^1_n \\ [\mathrm{SEP}] \quad t^2_1 \quad t^2_2 \quad ... \quad t^2_m\end{gathered} \tag{1}$$

To avoid the extra effort by the model to evidence the target word, we propose to introduce this information into the learning input. Thus, we mark the target word in S_t by using a special token before and after the target word[6]. The input used when two sentences are compared is presented in Equation 2. S_t is the first sentence with the target word t_i, S_d is the definition sentence, and t^k_x are their respective tokens.

$$\begin{gathered}\mathrm{input}^{\mathrm{sp1}}(\mathrm{S_t}, \mathrm{S_d}) = \\ [\mathrm{CLS}] \quad t^t_1 \quad t^t_2 \quad ... \quad \$ \quad t^t_i \quad \$ \quad ... \quad t^t_n \\ [\mathrm{SEP}] \quad t^d_1 \quad t^d_2 \quad ... \quad t^d_m\end{gathered} \tag{2}$$

In the case of hypernyms (sub-task 2), the input on the left side is kept as in Equation 2, but the right side includes the tagging of each hypernym as presented in Equation 3.

$$\begin{gathered}\mathrm{input}^{\mathrm{sp2}}(\mathrm{S_t}, \mathrm{S_h}) = \\ [\mathrm{CLS}] \quad t^t_1 \quad t^t_2 \quad ... \quad \$ \quad t^t_i \quad \$ \quad ... \quad t^t_n \\ [\mathrm{SEP}] \quad s^h_1 \quad \$ \quad s^h_2 \quad \$ \quad ... \quad \$ \quad s^h_l\end{gathered} \tag{3}$$

3.4 Verifying the senses

We trained two separated models, one for each sub-problem using the architecture defined in Section 3.2. The output predictions of both models are used to solve the two-tasks problem. So, our overall prediction for the main problem is calculated by combining both prediction scores. First, we normalise the scores by applying a $softmax$ function to each model output, and then we select the prediction with the maximum probability as shown in Equation 5.

$$pred(x) = \begin{cases} 1, & \text{if } m^{sp1}_1(x) + m^{sp2}_1(x) \\ & \quad > m^{sp1}_0(x) + m^{sp2}_0(x). \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

where
$$m^{spk}_i = \frac{exp(p^{spk}_i)}{\sum_{j=\{0,1\}} exp(p^{spk}_j)} \tag{5}$$

and p^{spk}_i is the prediction value for the model k for the class i (m^{spk}_i).

4 Experiments and Results

4.1 Data Sets

The data set was manually created by the task organisers and some basic statistics are presented in Table 1. Detailed information can be found in the task description paper (Breit et al., 2020). No extra-annotated data was used for training.

	train	development	test
Positive	1206	198	-
Negative	931	191	-
Total	2137	389	1324

Table 1: WiC-TSV data set examples per class. Positive examples are identified as 'T' and negative as 'F' in the data set.

4.2 Implementation details

We implemented $BERT_{EM}$ of Baldini Soares et al. (2019) using the huggingface library (Wolf et al., 2019), and trained two models with each training set. We selected the model with best performance on the development set. Parameters were fixed as follows: 20 was used as maximum epochs, *Cross Entropy* as loss function, Adam as optimiser, *bert-base-uncased*[7] as pre-trained model, and other parameters were assigned following the library recommendations (Wolf et al., 2019). The final layer is composed of two neurons (negative or positive).

[6]We used '$' but any other special token may be used.

[7]https://github.com/google-research/bert

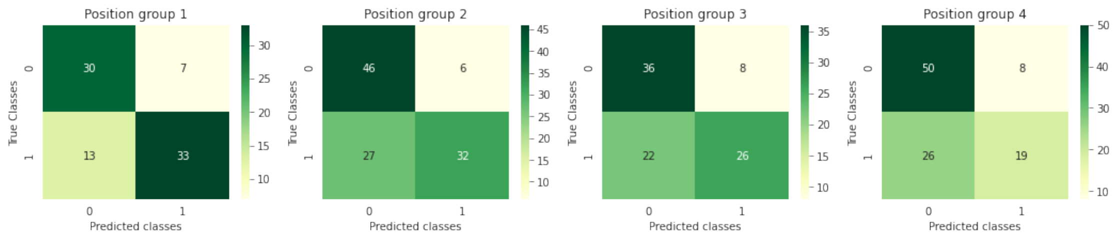

Figure 2: Confusion matrices for different position groups. Group 1 (resp. 2, 3, and 4) includes all sentences for which the target word appears in the first (resp. second, third, and fourth) quarter of the sentence.

4.3 Results

As the test labels are not publicly available, our following analysis is performed exclusively on the development set. Results on the test set were calculated by the task organisers.

We analyse confusion matrices depending on the position of the target word in the sentence as our strategy is based on marking the target word. These matrices are presented in Figure 2. The confusion matrix labelled as position group 1 shows our results when the target word is in the first 25% positions of the S_t sentence. Other matrices show the results of the remaining parts of the sentence (second, third, and fourth 25%, for respectively group 2, 3, and 4).

Confusion matrices show that the easiest cases are when the target word is located in the first 25%. Other parts are harder mainly because the system considers positive examples as negatives (high false negative rate). However, the system behaves correctly for negative examples independently of the position of the target word. To better understand this wrong classification of the positive examples, we calculated the true label distribution depending on the normalised prediction score as in Figure 3. Note that positive examples are mainly located on the right side but a bulk of them are located around the middle of the figure. It means that models m^{sp1} and m^{sp2} where in conflict and average results were slightly better for the negative class. In the development set, it seems important to correctly define a threshold strategy to better define which examples are marked as positive.

In our experiments, we implicitly used 0.5 as threshold[8] to define either the example belongs to the 'T' or 'F' class. When comparing Figures 3 and 4, we can clearly see that small changes in the threshold parameter would affect our results with

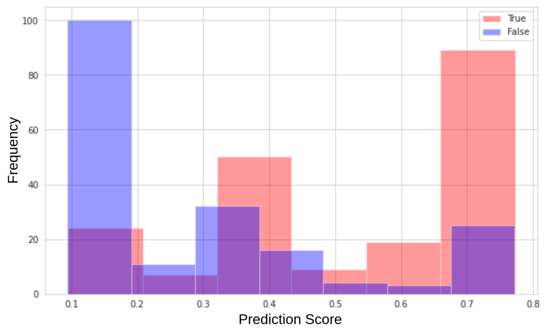

Figure 3: Histograms of predicted values in the dev set.

a larger impact in recall than in precision. This is mainly given to the fact that our two models contradict for some examples.

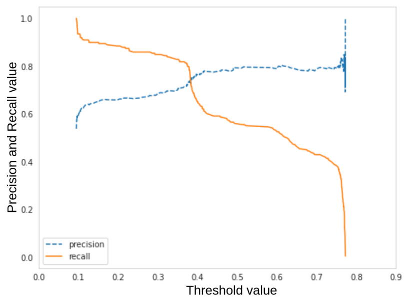

Figure 4: Precision/Recall curve for the development set for different threshold values.

We also considered the class distribution depending on a normalised distance between the target token and the beginning of the sentence. From Figure 5, we observe that both classes are less frequent at the beginning of the sentence with negative examples slightly less frequent than positive ones. It is interesting to remark that negative examples uniformly distribute after the first bin. On the contrary, the positive examples have a more unpredictable distribution indicating that a strategy based on only positions may fail. However, our strategy that combines markers to indicate the target word and a

[8]Because of the condition $m_1^{sp1}(x) + m_1^{sp2}(x) > m_0^{sp1}(x) + m_0^{sp2}(x)$.

		Global				WordNet/Wiktionary				Cocktails				Medical entities				Computer Science			
Run	User	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
run2	CTLR (ours)	**78,3**	**78,9**	78,0	**78,5**	72,1	75,8	70,7	73,2	87,5	82,4	90,3	86,2	85,9	86,7	85,8	86,3	83,3	78,4	88,5	83,1
szte	begab	66,9	61,6	92,5	73,9	70,2	66,5	89,6	76,4	55,1	48,9	96,8	65,0	65,4	60,5	95,3	74,0	70,2	61,3	97,4	75,2
szte2	begab	66,3	61,1	92,8	73,7	69,9	66,2	90,2	76,3	53,7	48,1	96,8	64,3	64,4	59,8	95,3	73,5	69,6	60,8	97,4	74,9
BERT	-	76,6	74,1	82,8	78,2	73,5	76,1	74,2	75,1	79,2	67,8	98,2	80,2	79,8	75,8	89,6	82,1	82,1	73,0	97,9	83,6
FastText	-	53,4	52,8	79,4	63,4	57,1	58,0	74,0	65,0	43,1	43,1	100,0	60,2	51,1	51,5	90,3	65,6	54,0	50,5	67,1	57,3
Baseline (true)		50,8	50,8	**100,0**	67,3	53,8	53,8	100,0	70,0	43,1	43,1	100,0	60,2	51,7	51,7	100,0	68,2	46,4	46,4	100,0	63,4
Human		85,3	80,2	96,2	87,4	82,1				92,0				89,1				86,5			

Table 2: Accuracy, Precision, Recall and F1 results of participants and baselines. Results where split by type. General results are included in column 'Global'. All results were calculated by the task organisers (Breit et al., 2020) as participants have not access to test labels. Best performance for each global metric is marked in **bold** for automatic systems.

strong neural language model (BERT) successfully manage to classify the examples.

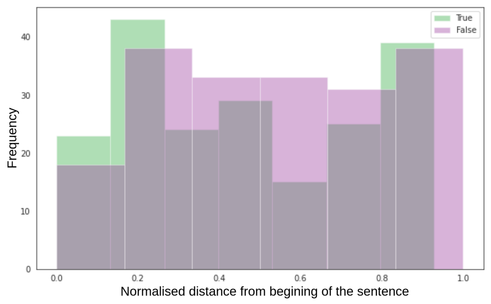

Figure 5: Position distribution based on the target token distances.

Finally, the main results calculated by the organisers are presented in Table 2. The global column presents the results for the global task, including definitions and hypernyms. Our submission is identified as run2-CTLR. In the global results, our strategy outperforms participants and baselines in terms of Accuracy, Precision, and F1. Best Recall performance is unsurprisingly obtained by the baseline (true) that corresponds to a system that predicts all examples as positives. Two strong baselines are included, FastText and BERT. Both baselines were calculated by the organisers with more details in (Breit et al., 2020). It is interesting to remark that the baseline BERT is very similar to our model but without the marked information. However, our model focuses more on improving Precision than Recall resulting with a clear improvement in terms of Accuracy but less important in terms of F1.

Organisers also provide results grouped by different types of examples. They included four types with three of them from domains that were not included in the training set[9]. From Table 2, we can also conclude that our system is able to adapt to out-of-domain topics as it is clearly shown for the *Cocktails* type in terms of F1, and also for the *Medical entities* type to a less extent. However, our system fails to provide better results than the standard BERT in terms of F1 for the *Computer Science* type. But, in terms of Accuracy, our strategy outperforms for a large margin the out-of-domain types (8.3, 6.1, and 1.2 improvements in absolute points for *Cocktails*, *Medical entities*, and *Computer Science* respectively). Surprisingly, it fails on both, F1 and Accuracy, for *WordNet/Wiktionary*.

5 Conclusion

This paper describes our participation in the WiC-TSV task. We proposed a simple but effective strategy for target sense verification. Our system is based on BERT and introduces markers around the target words to better drive the learned model. Our results are strong over an unseen collection used to verify senses. Indeed, our method (Acc=$78, 3$) outperforms other participants (second best participant, Acc=$66, 9$) and strong baselines (BERT, Acc=$76, 6$) when compared in terms of Accuracy, the official metric. This margin is even larger when the results are compared for the out-of-domain examples of the test collection. Thus, the results suggest that the extra information provided to the BERT model through the markers clearly boost performance.

As future work, we plan to complete the evaluation of our system with the WiC dataset (Pilehvar and Camacho-Collados, 2019) as well as the integration of the model into a recent multi-lingual entity linking system (Linhares Pontes et al., 2020) by marking the anchor texts.

[9]More details in (Breit et al., 2020).

Acknowledgements

This work has been partly supported by the European Union's Horizon 2020 research and innovation programme under grant 825153 (EMBEDDIA).

References

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In *ACL*. 2895–2905.

Lila Boualili, Jose G. Moreno, and Mohand Boughanem. 2020. MarkedBERT: Integrating Traditional IR Cues in Pre-Trained Language Models for Passage Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. Association for Computing Machinery, 1977–1980.

Anna Breit, Artem Revenko, Kiamehr Rezaee, Mohammad Taher Pilehvar, and Jose Camacho-Collados. 2020. WiC-TSV: An Evaluation Benchmark for Target Sense Verification of Words in Context. *arXiv:2004.15016* (2020).

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

Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2019. Sensebert: Driving some Sense into Bert. *arXiv:1908.05646* (2019).

Elvys Linhares Pontes, Jose G. Moreno, and Antoine Doucet. 2020. Linking Named Entities across Languages Using Multilingual Word Embeddings. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020 (JCDL '20)*. Association for Computing Machinery, 329–332.

Matthew E. Peters, Mark Neumann, Robert L. Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge Enhanced Contextual Word Representations. In *EMNLP*. 43–54.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 1267–1273.

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:1910.03771* (2019).

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 2335–2344.

Word Sense Disambiguation with Transformer Models

Pierre-Yves Vandenbussche
Elsevier Labs
Radarweg 29,
Amsterdam 1043 NX,
Netherlands
p.vandenbussche@elsevier.com

Tony Scerri
Elsevier Labs
1 Appold Street,
London EC2A 2UT, UK
a.scerri@elsevier.com

Ron Daniel Jr.
Elsevier Labs
230 Park Avenue,
New York, NY, 10169,
USA
r.daniel@elsevier.com

Abstract

In this paper, we tackle the task of Word Sense Disambiguation (WSD). We present our system submitted to the Word-in-Context Target Sense Verification challenge, part of the SemDeep workshop at IJCAI 2020 (Breit et al., 2020). That challenge asks participants to predict if a specific mention of a word in a text matches a pre-defined sense. Our approach uses pre-trained transformer models such as BERT that are fine-tuned on the task using different architecture strategies. Our model achieves the best accuracy and precision on Subtask 1 – make use of definitions for deciding whether the target word in context corresponds to the given sense or not. We believe the strategies we explored in the context of this challenge can be useful to other Natural Language Processing tasks.

1 Introduction

Word Sense Disambiguation (WSD) is a fundamental and long-standing problem in Natural Language Processing (NLP) (Navigli, 2009). It aims at clearly identifying which specific sense of a word is being used in a text. As illustrated in Table 1, in the sentence *I spent my spring holidays in Morocco.*, the word *spring* is used in the sense of *the season of growth*, and not in other senses involving coils of metal, sources of water, the act of jumping, etc.

The Word-in-Context Target Sense Verification challenge (WiC-TSV) (Breit et al., 2020) structures WSD tasks in particular ways in order to make the competition feasible. In Subtask 1, the system is provided with a sentence, also known as the *context*, the *target* word, and a definition also known as *word sense*. The system is to decide if the use of the target word matches the sense given by the definition. Note that Table 1 contains a *Hypernym* column. In Subtask 2 system is to decide if the use of the target in the context is a hyponym of the given hypernym. In Subtask 3 the system can use both the sentence and the hypernym in making the decision.

The dataset provided with the WiC-TSV challenge has relatively few sense annotated examples ($< 4,000$) and with a single target sense per word. This makes pre-trained Transformer models well suited for the task since the small amount of data would limit the learning ability of a typical supervised model trained from scratch.

Thanks to the recent advances made in language models such as BERT (Devlin et al., 2018) or XLNet (Yang et al., 2019) trained on large corpora, neural language models have established the state-of-the-art in many NLP tasks. Their ability to capture context-sensitive semantic information from text would seem to make them particularly well suited for this challenge. In this paper, we explore different fine-tuning architecture strategies to answer the challenge. Beyond the results of our system, our main contribution comes from the intuition and implementation around this set of strategies that can be applied to other NLP tasks.

2 Data Analysis

The Word-in-Context Target Sense Verification dataset consists of more than 3800 rows. As shown in Table 1, each row contains a target word, a context sentence containing the target, and both hypernym(s) and a definition giving a sense of the term. There are both positive and negative examples, the dataset provides a label to distinguish them.

Table 2 shows some statistics about the training, dev, and test splits within the dataset. Note the substantial differences between the test set and the training and dev sets. The longer length of the context sentences and definitions in the test set may have an impact on a model trained solely on the given training and dev sets. This is a known

Target Word	Pos.	Sentence	Hypernyms	Definition	Label
spring	3	I spent my spring holidays in Morocco .	season, time_of_year	the season of growth	T
integrity	1	the integrity of the nervous system is required for normal developments	honesty, honestness	moral soundness	F

Table 1: Examples of training data.

issue whose roots are explained in the dataset author's paper (Breit et al., 2020). The training and development sets come from WordNet and Wiktionary while the test set incorporates both general purpose sources WordNet and Wiktionary, and domain-specific examples from Cocktails, Medical Subjects and Computer Science. The difference in the distributions of the test set from the training and dev sets, the short length of the definitions and hypernyms, and the relatively small number of examples all combine to provide a good challenge for the language models.

3 System Description and Related Work

Word Sense Disambiguation is a long-standing task in NLP because of its difficulty and subtlety. One way the WiC-TSV challenge has simplified the problem is by reducing it to a binary yes/no decision over a single sense for a single pre-identified target. This is in contrast to most prior work that provides a pre-defined sense inventory, typically WordNet, and requires the system to both identify the terms and find the best matching sense from the inventory. WordNet provides extremely fine-grained senses which have been shown to be difficult for humans to accurately select (Hovy et al., 2006). Coupled with this is the task of even selecting the term in the presence of multi-word expressions and negations.

Since the introduction of the transformer self-attention-based neural architecture and its ability to capture complex linguistic knowledge (Vaswani et al., 2017), their use in resolving WSD has received considerable attention (Loureiro et al., 2020). A common approach consists in fine-tuning a single pre-trained transformer model to the WSD downstream task. The pre-trained model is provided with the task-specific inputs and further trained for several epochs with the task's objective and negative examples of the objective.

Our system is inspired from the work of Huang et al. (2019) where the WSD task can be seen as a binary classification problem. The system is given the target word in context (input sentence) and one

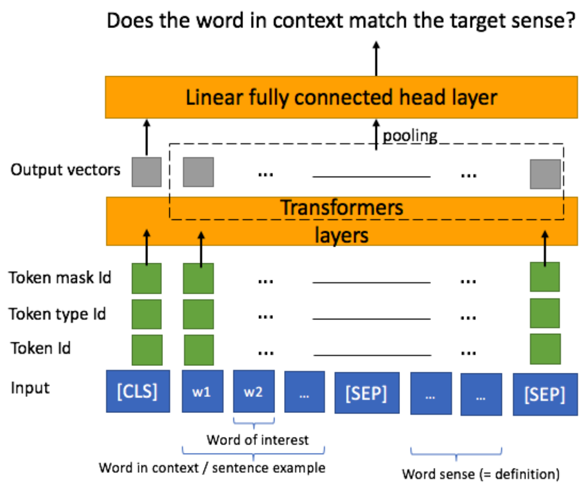
Does the word in context match the target sense?
Linear fully connected head layer
pooling
Output vectors
Transformers layers
Token mask Id
Token type Id
Token Id
Input
[CLS]
w1
w2
[SEP]
[SEP]
Word of interest
Word in context / sentence example
Word sense (= definition)

Figure 1: System overview

sense of the word separated by a special token (*[SEP]*). This configuration was originally used to predict whether two sentences follow each other in a text. But the learning power of the transformer architecture lets us learn this new task by simply changing the meaning of the fields in the input data while keeping the structure the same. We add a fully connected layer on top of the transformer model's layers with classification function to predict whether the target word in context matches the definition. This approach is particularly well suited for weak supervision and can generalise to word/sense pairs not previously seen in the training set. This overcomes the limitation of multi-class objective models, e.g. (Vial et al., 2019) that use a predefined sense inventory (as described above) and can't generalise to unseen word/sense pairs. An illustration of our system is provided in Figure 1.

4 Experiments and Results

The system described in the previous section was adapted in several ways as we tested alternatives. We first considered different transformer models, such as BERT v. XLNet. We then concentrated our efforts on one transformer model, BERT-base-uncased, and performed other experiments to improve performance.

All experiments were run five times with differ-

	Train Set	Dev Set	Test Set
Number Examples	2,137	389	1,305
Avg. Sentence Char Length	44 ± 27	44 ± 26	99 ± 87
Avg. Sentence Word Length	9 ± 5	9 ± 5	19 ± 16
Avg. Term Use	2.5 ± 2.7	1.0 ± 0.2	1.9 ± 4.4
Avg. Number Hypernyms	2.2 ± 1.5	2.2 ± 1.4	1.9 ± 1.3
Percentage of True Label	56%	51%	NA
Avg. Definition Char Length	54 ± 27	56 ± 27	157 ± 151
Avg. Definition Word Length	9.3 ± 4.7	9.6 ± 4.7	25.3 ± 23.9

Table 2: Dataset statistics. Values with $\pm$ are mean and SD

Model	Accuracy	F1
XLNet-base-cased	$.522 \pm .030$	$.666 \pm .020$
DistilBERT-base-uncased	$.612 \pm .017$	$.665 \pm .017$
RoBERTa-base	$.635 \pm .074$	$.717 \pm .030$
BERT-base-uncased	**$.723 \pm .023$**	**$.751 \pm .023$**

Table 3: Comparison of transformer models performance (lr=$5e^{-5}$; 3 epochs)

ent random seeds. We report the mean and standard deviation of the system's performance on the metrics of accuracy and F1. We believe this is more informative than a single 'best' number. All models in these experiments are trained on the training set and evaluated on the dev set.

In addition to the experiments whose results are reported here, we tried a variety of other things such as pooling methods (layers, ops), a Siamese network with shared encoders for two input sentences, and alternative loss calculations. None of them gave better results in the time available.

4.1 Alternative Transformer Models

We compared the following pre-trained transformer models from the HuggingFace transformers library: XLNet (Yang et al., 2019), BERT (Devlin et al., 2018), and derived models including RoBERTa (Liu et al., 2019) or DistilBERT (Sanh et al., 2019).

Following standard practice, those pretrained models were used as feature detectors for a fine-tuning pass using the fully-connected head layer. The results for those models are given in Table 3. The BERT-base-uncased model performed the best so it was the basis for further experiments described in the next section.

It is worth mentioning that no attempt was made to perform a hyperparameter optimization for each model. Instead, a single set of hyperparameters was used for all the models being compared.

Model	Accuracy	F1
BERT-base-uncased	$.723 \pm .023$	$.751 \pm .023$
+mask	$.699 \pm .011$	$.748 \pm .009$
+target emph	**$.725 \pm .013$**	**$.752 \pm .012$**
+mean-pooling	**$.737 \pm .017$**	**$.762 \pm .015$**
+freezing	**$.734 \pm .008$**	**$.761 \pm .010$**
+data augment.	**$.749 \pm .009$**	**$.752 \pm .011$**
+hypernyms	**$.726 \pm .012$**	**$.755 \pm .011$**

Table 4: Influence of strategies on model performance. We note in bold those that had a positive impact on the performance

4.2 Alternative BERT Strategies

Having selected the BERT-base-uncased pretrained transformer model, and staying with a single set of hyperparameters (learning rate = $5e^{-5}$ and 3 training epochs), there are still many different strategies that could be used to try to improve performance. The individual strategies are discussed below. The results for all the strategies are presented in Table 4

4.2.1 Masking the target word

We wondered if the context of the target word was sufficient for the model to predict whether the definition is correct. By masking the target word from the input sentence, we test the ability of the model to learn solely from the contextual words. We hoped this might improve its generalisation. Masking led to a small decrease in performance. This small delta indicates that the non-target words in the context have strong influence on the model's prediction of the correct sense.

4.2.2 Emphasising the word of interest

We wondered about the impact of taking the opposite tack and calling out the target word. As illustrated in Figure 1, some transformer models make use of *token type ids* (segment token indices) to indicate the first and second portion of the inputs.

We set the token(s) type of the target word in the input sentence to match that of the definition. Applying this strategy leads to a slight improvement in accuracy.

4.2.3 CLS vs. pooling over token sequence

The community has developed several common ways to select the input for the head binary classification layer. We compare the performance using the dedicated *[CLS]* token vector v. mean/max-pooling methods applied to the sequence hidden states of selected layers of the transformer model. Applying mean-pooling to the last layer gave the best accuracy and F1 of the configurations tested.

4.2.4 Weight Training vs. Freeze-then-Thaw

Another strategy centers on whether, and how, to update the pre-trained model parameters during fine-tuning, in addition to the training of the newly initialized fully connected head layer. Updating the pre-trained model would allow it to specialize on our downstream task but might lead to "catastrophic forgetting" where we destroy the benefit of the pre-trained model. One strategy the community has evolved (Bevilacqua and Navigli, 2020) first freezes the transformer model's parameters for several epochs while the head layer receives the updates. Later the pre-trained parameters are unfrozen and updated too. This strategy provides some improvements in accuracy and F1.

4.2.5 Data augmentation

Due to the small size of the training dataset, we experimented with data augmentation techniques while using only the data provided for the challenge. For each word in context/target sense pair, we generated:

- one positive example by replacing the target word with a random hypernym, if any exist.
- one negative example by associating the target word to a random definition.

This strategy triples the size of the training dataset. This strategy gave the greatest improvement (3.6%) of all those tested. Further work could test the effect of more negative examples.

4.2.6 Using Hypernyms (Subtask 3)

For the WiC-TSV challenge's Subtask 3 , the system can use the additional information of hypernyms of the target word. We simply concatenate the hypernyms to the definition. This strategy leads to a slight performance improvement, presumably because the hypernym indirectly emphasizes the intended sense of the target word.

Model	Acc.	Prec.	Recall	F1
Baseline (BERT)	.753	.717	**.849**	**.777**
Run1	.775	.804	.736	.769
Run2	**.778**	**.819**	.722	.768

Table 5: Model's Results on the Subtask 1 of the WiC-TSV challenge

5 Challenge Submission

The challenge allowed each participant to submit two results per task. However there was no clear winner from the strategies above; most led to a minimal improvement with a substantial standard deviation. We therefore selected our system for submitted results by a grid search over common hyperparameter values including the strategies mentioned previously. We use the train set for training and dev set to measure the performance of each model in the grid search. We chose accuracy as the main evaluation metric. For Subtask 1 we opted for the following parameters:

- Run1: BERT-base-uncased model trained for 3 epochs using the augmented dataset, with a learning rate of $7e^{-6}$ and emphasising the word of interest. Other parameters include: max sequence length of 256; train batch size of 32.
- Run2: we kept the parameters from the previous run, updating the learning rate to $1e^{-5}$.

The results on the private test set of the Subtask 1 are presented in Table 5. The Run2 of our system demonstrated a 3.3% accuracy and 14.2% precision improvements compared to the baseline.

For Subtask 3 we arrived at the following parameters:

- Run1: BERT-base-uncased model trained for 3 epochs using the original dataset, with a learning rate of $1e^{-5}$. Other parameters include: max sequence length of 256; train batch size of 32.
- Run2: we kept the parameters from the previous run, extending the number of training epochs to 5.

Model	Acc.	Prec.	Recall	F1
Baseline (BERT)	**.766**	**.741**	.828	**.782**
Run1	.694	.643	**.893**	.747
Run2	.719	.669	.885	.762

Table 6: Model's Results on the Subtask 3 of the WiC-TSV challenge

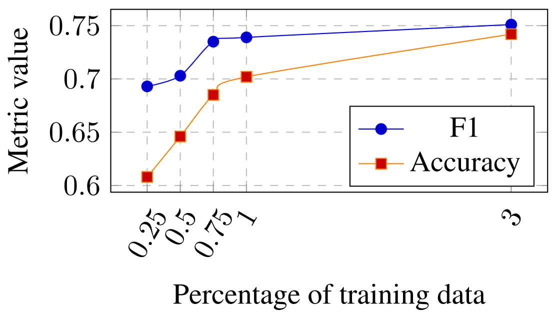

Figure 2: Influence of training data size on model performance. We used the augmented dataset to reach a proportion of 3. Parameters from Subtask 1 Run2 were used for this comparison.

The results on the private test set of the SubTask 3 are presented in Table 6. Compared to using the sentence and definition alone, our naive approach to handling hypernyms hurt performance.

6 Discussion and Future Work

We applied transformer models to tackle a Word Sense Disambiguation challenge. As in much of the current NLP research, pre-trained transformer models demonstrated a good ability to learn from few examples with high accuracy. Using different architecture modifications, and in particular the use of the token type id to flag the word of interest along with automatically augmented data, our system demonstrated the best accuracy and precision in the competition and third-best F1. There is still a noticeable gap to human performance on this dataset (85.3 acc.), but the level of effort required to create these kinds of systems is easily within reach of small groups or individuals. Despite the test set having a very different distribution than the training/development sets, our system demonstrated similar performance on both the development and test sets.

An analysis of the errors produced by our best performing model on the dev set (Subtask 1, Run2) is presented in Table 7. It shows a mix of obvious errors and more ambiguous ones where it has been difficult for the model to draw conclusions from the limited context provided by the sentence. For instance, the short sentence *it's my go* could very well correspond to the associated definition *a usually brief attempt* of the target word *go*.

As motivated by the construction of an augmented dataset, we believe that increasing the size of the training dataset would probably lead to improved performance, even without system changes. To test this hypothesis we measured the performance of our best model with increasing fractions of the training data. The results in Figure 2 show improvement as the fraction of the training dataset grows.

As a counterbalance to the positive note above, we must note that this challenge set up WSD as a binary classification problem. This is a considerable simplification from the more general sense inventory approach. Further work will be needed to obtain similar accuracy in that regime.

References

Michele Bevilacqua and Roberto Navigli. 2020. Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2854–2864.

Anna Breit, Artem Revenko, Kiamehr Rezaee, Mohammad Taher Pilehvar, and Jose Camacho-Collados. 2020. Wic-tsv: An evaluation benchmark for target sense verification of words in context. *arXiv preprint*, arXiv:2004.15016.

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*.

E. Hovy, M. Marcus, Martha Palmer, L. Ramshaw, and R. Weischedel. 2006. Ontonotes: The 90 In *HLT-NAACL*.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. Glossbert: Bert for word sense disambiguation with gloss knowledge. *arXiv preprint arXiv:1908.07245*.

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*.

Daniel Loureiro, Kiamehr Rezaee, Mohammad Taher Pilehvar, and Jose Camacho-Collados. 2020. Language models and word sense disambiguation: An overview and analysis. *arXiv preprint arXiv:2008.11608*.

Target Word	Pos.	Sentence	Definition	Label	Pred
criticism	4	the senator received severe criticism from his opponent	a serious examination and judgment of something	F	T
go	3	it 's my go	a usually brief attempt	F	T
reappearance	1	the reappearance of Halley 's comet	the act of someone appearing again	F	T
continent	5	pioneers had to cross the continent on foot	one of the large landmasses of the earth	T	F
rail	4	he was concerned with rail safety	short for railway	T	F

Table 7: Examples of errors in the development set for the model used in Subtask 1 Run2.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):1–69.

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*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2019. Sense vocabulary compression through the semantic knowledge of wordnet for neural word sense disambiguation. *arXiv preprint arXiv:1905.05677*.

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 5753–5763.

Embeddings for the Lexicon: Modelling and Representation

Christian Chiarcos[1] **Thierry Declerck**[2] **Maxim Ionov**[1]
[1] Applied Computational Linguistics Lab, Goethe University, Frankfurt am Main, Germany
[2] DFKI GmbH, Multilinguality and Language Technology, Saarbrücken, Germany
{chiarcos,ionov}@informatik.uni-frankfurt.de
declerck@dfki.de

Abstract

In this paper, we propose an RDF model for representing embeddings for elements in lexical knowledge graphs (e.g. words and concepts), harmonizing two major paradigms in computational semantics on the level of representation, i.e., distributional semantics (using numerical vector representations), and lexical semantics (employing lexical knowledge graphs). Our model is a part of a larger effort to extend a community standard for representing lexical resources, OntoLex-Lemon, with the means to connect it to corpus data. By allowing a conjoint modelling of embeddings and lexical knowledge graphs as its part, we hope to contribute to the further consolidation of the field, its methodologies and the accessibility of the respective resources.

1 Background

With the rise of word and concept embeddings, lexical units at all levels of granularity have been subject to various approaches to embed them into numerical feature spaces, giving rise to a myriad of datasets with pre-trained embeddings generated with different algorithms that can be freely used in various NLP applications. With this paper, we present the current state of an effort to connect these embeddings with lexical knowledge graphs.

This effort is a part of an extension of a widely used community standard for representing, linking and publishing lexical resources on the web, OntoLex-Lemon[1]. Our work aims to complement the emerging OntoLex module for representing Frequency, Attestation and Corpus Information (FrAC) which is currently being developed by the W3C Community Group "Ontology-Lexica", as presented in [4]. There we addressed only frequency and attestations, whereas core aspects of corpus-based information such as embeddings were identified as a topic for future developments. Here, we describe possible use-cases for the latter and present our current model for this.

2 Sharing Embeddings on the Web

Although word embeddings are often calculated on the fly, the community recognizes the importance of pre-trained embeddings as these are readily available (it saves time), and cover large quantities of text (their replication would be energy- and time-intense). Finally, a benefit of re-using embeddings is that they can be grounded in a well-defined, and, possibly, shared feature space, whereas locally built embeddings (whose creation involves an element of randomness) would reside in an isolate feature space. This is particularly important in the context of multilingual applications, where collections of embeddings are in a single feature space (e.g., in MUSE [6]).

Sharing embeddings, especially if calculated over a large amount of data, is not only an economical and ecological requirement, but unavoidable in many cases. For these, we suggest to apply established web standards to provide metadata about the lexical component of such data. We are not aware of any provider of pre-trained word embeddings which come with machine-readable metadata. One such example is language information: while ISO-639 codes are sometimes being used for this purpose, they are not given in a machine-readable way, but rather documented in human-readable form or given implicitly as part of file names.[2]

2.1 Concept Embeddings

It is to be noted, however, that our focus is not so much on word embeddings, since lexical information in this context is apparently trivial – plain

[1] https://www.w3.org/2016/05/ontolex/

[2] See the ISO-639-1 code 'en' in FastText/MUSE files such as https://dl.fbaipublicfiles.com/arrival/vectors/wiki.multi.en.vec.

tokens without any lexical information do not seem to require a structured approach to lexical semantics. This changes drastically for embeddings of more abstract lexical entities, e.g., word senses or concepts [17], that need to be synchronized between the embedding store and the lexical knowledge graph by which they are defined. WordNet [14] synset identifiers are a notorious example for the instability of concepts between different versions: Synset `00019837-a` means ‘incapable of being put up with’ in WordNet 1.71, but ‘established by authority’ in version 2.1. In WordNet 3.0, the first synset has the ID `02435671-a`, the second `00179035-s`.[3]

The precompiled synset embeddings provided with AutoExtend [17] illustrate the consequences: The synset IDs *seem* to refer to WordNet 2.1 (`wn-2.1-00001742-n`), but use an ad hoc notation and are not in a machine-readable format. More importantly, however, there *is* no synset `00001742-n` in Princeton WordNet 2.1. This *does* exist in Princeton WordNet 1.71, and in fact, it seems that the authors actually used this edition instead of the version 2.1 they refer to in their paper. Machine-readable metadata would not prevent such mistakes, but it would facilitate their verifiability. Given the current conventions in the field, such mistakes will very likely go unnoticed, thus leading to highly unexpected results in applications developed on this basis. Our suggestion here is to use resolvable URIs as concept identifiers, and if they provide machine-readable lexical data, lexical information about concept embeddings can be more easily verified and (this is another application) integrated with predictions from distributional semantics. Indeed, the WordNet community has adopted OntoLex-Lemon as an RDF-based representation schema and developed the Collaborative Interlingual Index (ILI or CILI) [3] to establish sense mappings across a large number of WordNets. Reference to ILI URIs would allow to retrieve the lexical information behind a particular concept embedding, as the WordNet can be queried for the lexemes this concept (synset) is associated with. A versioning mismatch can then be easily spotted by comparing the cosine distance between the word embeddings of these lexemes and the embedding of the concept presumably derived from them.

[3]See `https://github.com/globalwordnet/ili`

2.2 Organizing Contextualized Embeddings

A related challenge is the organization of contextualized embeddings that are not adequately identified by a string (say, a word form), but only by that string in a particular context. By providing a reference vocabulary to organize contextualized embeddings together with the respective context, this challenge will be overcome, as well.

As a concrete use case of such information, consider a classical approach to Word Sense Disambiguation: The Lesk algorithm [12] uses a bag-of-words model to assess the similarity of a word in a given context (to be classified) with example sentences or definitions provided *for different senses* of that word in a lexical resource (training data, provided, for example, by resources such as WordNet, thesauri, or conventional dictionaries). The approach was very influential, although it always suffered from data sparsity issues, as it relied on string matches between a very limited collection of reference words and context words. With distributional semantics and word embeddings, such sparsity issues are easily overcome as literal matches are no longer required.

Indeed, more recent methods such as BERT allow us to induce contextualized word embeddings [20]. So, given only a single example for a particular word sense, this would be a basis for a more informed word sense disambiguation. With more than one example per word sense, this is becoming a little bit more difficult, as these now need to be either aggregated into a single sense vector, or linked to the particular word sense they pertain to (which will require a set of vectors as a data structure rather than a vector). For data of the second type, we suggest to follow existing models for representing attestations (glosses, examples) for lexical entities, and to represent contextualized embeddings (together with their context) as properties of attestations.

3 Addressing the Modelling Challenge

In order to meet these requirements, we propose a formalism that allows the conjoint publication of lexical knowledge graphs and embedding stores. We assume that future approaches to access and publish embeddings on the web will be largely in line with high-level methods that abstract from details of the actual format and provide a conceptual view on the data set as a whole, as provided, for ex-

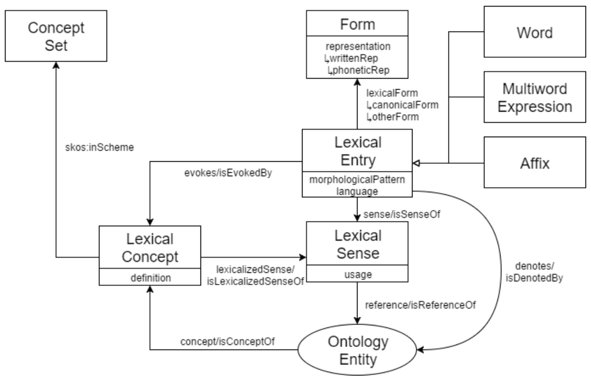

Figure 1: OntoLex-Lemon core model

ample, by the `torchtext` package of PyTorch.[4]

There is no need to limit ourselves to the currently dominating tabular structure of commonly used formats to exchange embeddings; access to (and the complexity of) the actual data will be hidden from the user.

A promising framework for the conjoint publication of embeddings and lexical knowledge graph is, for example, RDF-HDT [9], a compact binary serialization of RDF graphs and the literal values these comprise. We would assume it to be integrated in programming workflows in a way similar to program-specific binary formats such as Numpy arrays in pickle [8].

3.1 Modelling Lexical Resources

Formalisms to represent lexical resources are manifold and have been a topic of discussion within the language resource community for decades, with standards such as LMF [10], or TEI-Dict [19] designed for the electronic edition and/or search in individual dictionaries. Lexical data, however, does not exist in isolation, and synergies can be unleashed if information from different dictionaries is combined, e.g., for bootstrapping new bilingual dictionaries for languages X and Z by using another language Y and existing dictionaries for $X \mapsto Y$ and $Y \mapsto Z$ as a pivot.

Information integration beyond the level of individual dictionaries has thus become an important concern in the language resource community. One way to achieve this is to represent this data as a knowledge graph. The primary community standard for publishing lexical knowledge graphs on the web is OntoLex-Lemon [5].

OntoLex-Lemon defines four main classes of lexical entities, i.e., concepts in a lexical resource: (1) **LexicalEntry** representation of a lexeme in a lexical knowledge graph, groups together form(s) and sense(s), resp. concept(s) of a particular expression; (2) (lexical) **Form**, written representation(s) of a particular lexical entry, with (optional) grammatical information; (3) **LexicalSense** word sense of one particular lexical entry; (4) **LexicalConcept** elements of meaning with different lexicalizations, e.g., WordNet synsets.

As the dominant vocabulary for lexical knowledge graphs on the web of data, the OntoLex-Lemon model has found wide adaptation beyond its original focus on ontology lexicalization. In the WordNet Collaborative Interlingual Index [3] mentioned before, OntoLex vocabulary is used to provide a single interlingual identifier for every concept in every WordNet language as well as machine-readable information about it (including links with various languages).

To broaden the spectrum of possible usages of the core model, various extensions have been developed by the community effort. This includes the emerging module for frequency, attestation and corpus-based information, FrAC.

The vocabulary elements introduced with FrAC cover frequency and attestations, with other aspects of corpus-based information described as prospective developments in our previous work [4]. Notable aspects of corpus information to be covered in such a module for the purpose of lexicographic applications include contextual similarity, co-occurrence information and embeddings.

Because of the enormous technical relevance of the latter in language technology and AI, and because of the requirements identified above for the publication of embedding information over the web, this paper focuses on embeddings,

3.2 OntoLex-FrAC

FrAC aims to (1) extend OntoLex with corpus information to address challenges in lexicography, (2) model lexical and distributional-semantic resources (dictionaries, embeddings) as RDF graphs, (3) provide an abstract model of relevant concepts in distributional semantics that facilitates applications that integrate both lexical and distributional information. Figure 2 illustrates the FrAC concepts and properties for frequency and attestations (gray) along with new additions (blue) for embeddings as a major form of corpus-based information.

Prior to the introduction of embeddings, the

[4]https://pytorch.org/text/

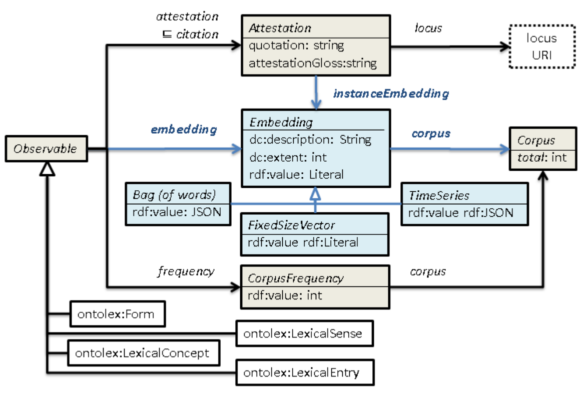

Figure 2: OntoLex-FrAC overview: Extensions for embeddings highlighted in blue

main classes of FrAC were (1) **Observable**, any unit within a lexical resource about which information can be found in a corpus, includes all main classes of OntoLex-Core, lexical forms, lexical entries, lexical senses and concepts; (2) **Corpus**, a collection of linguistic data from which empirical evidence can be derived, including corpora in the sense as understood in language technology; (3) **Attestation**, example for a specific phenomenon, usage or form found in a corpus or in the literature; (4) **CorpusFrequency**, absolute frequency of a particular observation in a given corpus.

4 Modelling Embeddings in FrAC

In the context of OntoLex, word embeddings are to be seen as a feature of individual lexical forms. However, in many cases, word embeddings are not calculated from plain strings, but from normalized strings, e.g., lemmatized text. For such data, we model every individual lemma as an `ontolex:LexicalEntry`. Moreover, as argued in Sec. 2, embeddings are equally relevant for lexical senses and lexical concepts; the `embedding` property that associates a lexical entity with an embedding is thus applicable to every `Observable`.

4.1 Word Embeddings

Pre-trained word embeddings are often distributed as text files consisting of the label (token) and a sequence of whitespace-separated numbers. E.g. the entry for the word *frak* from the GloVe embeddings [15]:

```
frak 0.015246 -0.30472 0.68107 ...
```

Since our focus on publishing and sharing embeddings, we propose to provide the value of an embedding as a literal `rdf:value`. If necessary, more elaborate representations, e.g., using `rdf:List`, may subsequently be generated from these literals.

A natural and effort-less modelling choice is to represent embedding values as string literals with whitespace-separated numbers. For decoding and verification, such a representation benefits from metadata about the length of the vector. For a fixed-size vector, this should be provided by `dc:extent`. An alternative is an encoding as JSON list. In order to support both structured and string literals, FrAC does not restrict the type of the `rdf:value` of embeddings.

Lexicalized embeddings should be published together with their metadata, at least procedure/method (`dct:description` with free text, e.g., *"CBOW"*, *"SKIP-GRAM"*, *"collocation counts"*), data basis (`frac:corpus`), and dimensionality (`dct:extent`).

We thus introduce the concept embedding, with a designated subclass for fixed-size vectors:

Embedding (Class) is a representation of a given `Observable` in a numerical feature space. It is defined by the methodology used for creating it (`dct:description`), the URI of the corpus or language resource from which it was created (`corpus`). The literal value of an Embedding is provided by `rdf:value`.
`Embedding` $\sqsubseteq$ `rdf:value exactly 1` $\sqcap$ `corpus exactly 1` $\sqcap$ `dct:description min 1`

embedding (ObjectProperty) is a relation that maps an `Observable` into a numerical feature space. An embedding is a structure-preserving mapping in the sense that it encodes and preserves contextual features of a particular `Observable` (or, an aggregation over all its attestations) in a particular corpus.

FixedSizeVector (Class) is an `Embedding` that represents a particular `Observable` as a list of numerical values in a k-dimensional feature space. The property `dc:extent` defines k.

For the GloVe example, a lemma (lexical entry) embedding can be represented as follows:

```
:frak  a  ontolex:LexicalEntry;
  ontolex:canonicalForm/
    ontolex:writtenRep  "frak"@en;
  frac:embedding  [
    a  frac:FixedSizeVector;
        rdf:value  "0.015246  ...";
    dct:source
      <https://catalog.ldc....>;
    dct:extent 50^^^xsd:int;
    dct:description "GloVe v.1.1,
        ..."@en. ].
```

4.2 Contextualized Embeddings

Above, we mentioned contextualized embeddings, and more recent methods such as ELMo [16], Flair NLP [1], or BERT [7] have been shown to be remarkably effective at many NLP problems.

In the context of lexical semantics, contextual embeddings can prove beneficial for inducing or distinguishing word senses, and in extension of the classical Lesk algorithm, for example, a lexical sense can be described by means of the contextualized word embeddings for the examples associated with that particular lexical sense, and words for which word sense disambiguation is to be performed can then just be compared with these. These examples then serve a similar function as attestations in a dictionary, and indeed, the link has been emphasized before [11]. Within FrAC, contextualized embeddings are thus naturally modelled as a property of `Attestation`.

instanceEmbedding (ObjectProperty) for a given `Attestation`. The property `instanceEmbedding` provides an embedding of the example in its current corpus context into a numerical feature space (see `Embedding`).

In this modelling, multiple contextualized embeddings can be associated with, say, a lexical sense by means of an attestation that then points to the actual string context. Considering *play*, multiple WordNet examples (glosses) per sense can thus be rendered by different fixed-size vectors:

```
wn31:play_n a ontolex:LexicalEntry;
  ontolex:sense wn31:07032045-n,
    wn31:play_n_4, ...
wn31:07032045-n
    a ontolex:LexicalSense;
    frac:attestation [
        frac:quotation "the play
        lasted two hours";
        frac:locus wn31:07032045-n;
        frac:instanceEmbedding
            wn31-bert:07032045-n-1
        ].
wn31-bert:07032045-n-1 a
    frac:FixedSizeVector;
    dc:extent "300"^^xsd:int;
    rdf:value "0.327246 0.48170 ...";
    dc:description "...";
    frac:corpus <http://wordnet-rdf.
       princeton.edu/static/wordnet.
       nt.gz> .
```

Like most RDF models, this appears to be overly verbose, but by introducing a subclass for a set of embeddings from which the embedding can inherit extent, corpus, and description information, e.g. `:WN31FixedSizeVector`, the BERT embedding in this example becomes much more digestable – without any loss in information:

```
wn31-bert:07032045-n-1 a
    :WN31FixedSizeVector;
    rdf:value "0.327246 0.48170 ...".
```

4.3 Other Types of Embeddings

FrAC is not restricted to uses in language technology; its definition of 'embedding' is thus broader than the conventional interpretation of the term in NLP, and based on its more general usage across multiple disciplines. In mathematics, the embedding f of an object X in another object Y ($f : X \to Y$) is defined as an injective, structure-preserving map (morphism). Important in the context of FrAC is the structure-preserving character of the projection into numerical feature spaces, as embeddings are most typically used to assess similarity, e.g., by means of cosine measure, the entire point being that these assessments remain stable when cooccurrence vectors are projected into a lower-dimensional space.

In computational lexicography, raw collocation counts continue to be used for the same purpose. In its classical form, these collocation counts get aggregated into bags of words, sometimes augmented with numerical weights. Bags of words are closely related to raw collocation vectors (and thus, prototypical embeddings), with the main difference being that collocation vectors consider only a finite set of diagnostic collocates (the reference dictionary), whereas a bag of words can include *any* word that occurs in the immediate context of the target expression.

In this understanding, a bag of words can be considered as an infinite-dimensional collocation vector, i.e., as an embedding in the broad sense introduced above. The practical motivation is that applications of bag-of-words models and fixed-size vectors are similar, and that bags of words remain practically important to a significant group of OntoLex or FrAC users.

A difference is that a bag of words model, if it provides frequency information or another form of numerical scores, must not be modelled by a plain list, but rather, by a dictionary or a map. As a data structure for this, we recommend JSON literals.

To represent this model, we introduce the following `Embedding` subclass:

BagOfWords (Class) is an `Embedding` that represents an `Observable` by a set of collocates or their mapping to numerical scores.

Another type of embeddings concerns sequential data, and one example for that are multimodal corpora. In a case study with Leiden University, we explored the encoding of dictionaries for African sign languages. In addition to graphics and videos, such dictionaries can also contain sensor data that records the position of hands and digits during the course of a sign or a full conversation. To search in this data, conventional techniques like dynamic time warp [2] are being applied – effectively in analogy with cosine distance among finite-size vectors. Furthermore, such data has also been addressed in NLP research in the context of neural time series transformation for the sake of translation between or from sign languages [18]. Also, in an NLP context, time series analysis is relevant for stream processing, so this would make this data structure of more general interest [13] in and beyond the language technology community.

Both from a modelling perspective and in terms of actual uses of such forms of lexical data, it is thus appealing to extend the concept of embeddings to time series data. In our current use case, we assume that time series data is mostly relevant in relation to attestations rather than OntoLex core concepts, but we can foresee that generalizations over multiple attestations could be developed at some point which would then used to extrapolate prototypical time series information for canonical forms, lexical entries, senses, or concepts.

To represent this, we introduce another `Embedding` subclass:

TimeSeries (Class) is an `Embedding` that represents an `Observable` or its `Attestation` as a sequence of a fixed number of data points recorded over a certain period of time.

The `dc:extent` of a time series specifies the number of data points per observation. The `rdf:value` should be a structured JSON literal.

5 Summary and Outlook

We identified a number of shortcomings in the (lack of) standards applied by providers of pre-trained embeddings on the web and aim to address them by providing a vocabulary that covers the relevant aspects of lexical data involved in this context, grounded in existing specifications for lexical metadata and publication forms for lexical knowledge graphs on the web of data. We provide an RDF vocabulary for encoding numerical, corpus-based information, in particular, embeddings such as those prevalent in language technology, in conjunction with, or as a part of lexical knowledge graphs. We cover a broad range of applications from NLP to computational lexicography, and a broad range of data structures that fall under a generalized understanding of embeddings.

Notable features of our proposal include (1) the coverage of a broad band-width of use cases, (2) its firm grounding in commonly used community standards for lexical knowledge graphs, (3) the possibility to provide machine-readable metadata and machine-readable lexical data along, and even in close integration with word, lexeme, sense or concept embeddings, (4) the extension beyond fixed-size vectors as currently dominating in language technology applications, (5) and a designated vocabulary for organizing contextualized embeddings with explicit links to both their respective context and the lexical entities they refer to.

At the time of writing, one dataset (AutoExtend) has been successfully transformed from its traditional publication form and integrated with a lexical knowledge graph. However, we are still in the process of evaluating which WordNet edition the AutoExtend data actually refers to. Another dataset currently under construction is a dictionary of African sign languages, where time series information is being encoded as attestation-level embeddings.

We see our work as a building block for the development of convergencies between the Linguistic Linked Open Data and the NLP/ML community, as a conjoint modelling – or, at least, compatible levels of representation, allow to combine both technology stacks to common problems. For the NLP/ML community, machine-readable metadata and adherence to established community standards will facilitate the potential for verifying, enriching and building embeddings with or against lexical resources. For computational lexicography, closer ties with the language technology community will facilitate the uptake of machine-learning methods and their evaluation, and intensify synergies between both fields.

Acknowledgements

The research presented in this paper was in part supported by the European Horizon2020 Prêt-à-LLOD project (grant agreement no. 825182), by the project "Linked Open Dictionaries" (LiODi), funded by the German Ministry of Education and Science (BMBF), and the COST Action CA18209 European network for Web-centred linguistic data science ("NexusLinguarum").

References

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. 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 2018. Association for Computational Linguistics.

[2] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:, 1994.

[3] Francis Bond, Piek Vossen, John P McCrae, and Christiane Fellbaum. Cili: the collaborative interlingual index. In *Proceedings of the Global WordNet Conference*, volume 2016, 2016.

[4] Christian Chiarcos, Maxim Ionov, Jesse de Does, Katrien Depuydt, Fahad Khan, Sander Stolk, Thierry Declerck, and John Philip McCrae. Modelling frequency and attestations for ontolex-lemon. In *Proceedings of the 2020 Globalex Workshop on Linked Lexicography*, pages 1–9, 2020.

[5] Philipp Cimiano, John P. McCrae, and Paul Buitelaar. Lexicon Model for Ontologies: Community Report, 2016.

[6] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[8] Laurent Fasnacht. mmappickle: Python 3 module to store memory-mapped numpy array in pickle format. *Journal of Open Source Software*, 3(26):651, 2018.

[9] Javier D Fernández, Miguel A Martínez-Prieto, Claudio Gutiérrez, Axel Polleres, and Mario Arias. Binary rdf representation for publication and exchange (hdt). *Journal of Web Semantics*, 19:22–41, 2013.

[10] Gil Francopoulo, Monte George, Nicoletta Calzolari, Monica Monachini, Nuria Bel, Mandy Pet, and Claudia Soria. Lexical markup framework (lmf). In *International Conference on Language Resources and Evaluation-LREC 2006*, page 5, 2006.

[11] Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. Glossbert: Bert for word sense disambiguation with gloss knowledge. *arXiv preprint arXiv:1908.07245*, 2019.

[12] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, 1986.

[13] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *In Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11. ACM Press, 2003.

[14] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[15] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[16] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[17] Sascha Rothe and Hinrich Schütze. Autoextend: Combining word embeddings with semantic resources. *Computational Linguistics*, 43(3):593–617, 2017.

[18] S. S Kumar, T. Wangyal, V. Saboo, and R. Srinath. Time series neural networks for real time sign language translation. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 243–248, 2018.

[19] TEI. TEI P5: Guidelines for electronic text encoding and interchange. Technical report, 2019.

[20] Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings. *arXiv preprint arXiv:1909.10430*, 2019.

Relation Classification via Relation Validation

Jose G. Moreno
University of Toulouse
IRIT, UMR 5505 CNRS
F-31000, Toulouse, France
jose.moreno@irit.fr

Antoine Doucet
University of La Rochelle
L3i
F-17000, La Rochelle, France
antoine.doucet@univ-lr.fr

Brigitte Grau
LIMSI, UPR 3251 CNRS
F - 91405 Orsay , France
bg@limsi.fr

Abstract

Recognising if a relation holds between two entities in a text plays a vital role in information extraction. To address this problem, multiple models have been proposed based on fixed or contextualised word representations. In this paper, we propose a meta relation classification model that can integrate the most recent models by the use of a related task, namely relation validation. To do so, we encode the text that may contain the relation and a relation triplet candidate into a sentence-triplet representation. We grounded our strategy in recent neural architectures that allow single sentence classification as well as pair comparisons. Finally, our model is trained to determine the most relevant sentence-triplet pair from a set of candidates. Experiments on two public data sets for relation extraction show that the use of the sentence-triplet representation outperforms strong baselines and achieves comparable results when compared to larger models.

1 Introduction

Recognising and classifying relations between two entities in a text plays a vital role in knowledge base population (KBP), a major sub-task of information extraction (IE). Some examples of typical relations in knowledge bases (KB) are spouse, CEO, place of birth, profession, etc. Nowadays, there exist large KB that store millions of facts such as DBpedia (Bizer et al., 2009) or YAGO (Hoffart et al., 2013). However, more than 70% of people entities have not associated information for relations such as place of birth or nationality (Dong et al., 2014).

Most approaches model the relation classification (RC) (dos Santos et al., 2015; Nguyen and Grishman, 2015) task as a learning problem where it is required to predict if a passage contains a type of relation (multi-class classification). This setup requires annotated examples of each class, i.e. each type of relation, which can be difficult to obtain. To overcome this problem, distant supervision has been proposed (Mintz et al., 2009) for automatically annotating texts given relation triplets existing in a KB by projecting triplets into texts to increase the input data. Its main counterpart is that distant supervision models must deal with wrongly annotated examples. The difficulty of the task is shown by the results of the TAC KBP slot filling task[1]. For instance, in 2014, the maximum F1-score of the task was 0.3672 (Surdeanu and Ji, 2014). Another trend is trying to collect information directly from the web in an unsupervised setting, i.e. the open IE paradigm (Banko et al., 2007). In these two last settings, one crucial point is to be able to assess the validity of the extracted relations. This point motivated an extra track in TAC KBP 2015 following a divide-and-conquer setup. It consists in validating the relations extracted by relation extraction (RE) systems in order to improve their final scores.

The purpose of relation validation (RV) aims at taking advantage of several hypotheses, provided by one or several systems, for improving the recognition of relations in texts and discarding false ones. Given a candidate relation triplet $(e1, R, e2)$ and a passage, this task can be defined as learning to decide if the passage supports the relation in a binary classification setup. Trigger words and relation patterns are usually modelled in relation validation as features for representing the relation type. In Wang and Neumann (2008), the relation validation setup is modified and presented as an entailment problem, where systems learn whether the text entails the relation based on linguistic features.

In this paper, we propose not only to learn the representation of the relation type, but also to learn the representation of the validation knowledge by using a neural architecture for modelling relation

[1]https://catalog.ldc.upenn.edu/LDC2018T22

validation, inspired by neural entailment models. We aim to decide whether the text supports the relation by encoding the text and the triplet[2] in a transformer architecture as in (Baldini Soares et al., 2019; Zhao et al., 2019). Once a model for relation validation is learned, we use it to validate the output of a relation classification model. Our experiments show that our proposal outperforms robust neural models for relation classification but fails to improve most recent works.

The remainder of this paper is structured as follows: Section 2 presents some relevant models for relation classification and validation. Section 3 details our strategy to classify relations based on relation validation. Then, the experimental setup and results are presented in Sections 4. Finally, conclusions are drawn in Section 5.

2 Related Work

Different ensemble models (Viswanathan et al., 2015) have been defined for the relation validation KBP task based on the prediction made by the RE systems. However, Yu et al. (2014) show that relation validation requires considering linguistic features for recognising if a relation is expressed in a text by exploiting rich linguistic knowledge from multiple lexical, syntactic, and semantic levels. In Wang and Neumann (2008), the relation to validate is transformed by simple patterns in a sentence and an alignment between the two texts is performed by a kernel-based approach.

Traditional methods for relation extraction are based on feature engineering and rely on lexical and syntactic information. Dependency trees provide clues for deciding the presence of a relation in unsupervised relation extraction (Culotta and Sorensen, 2004; Bunescu and Mooney, 2005; Fundel et al., 2007). Gamallo et al. (2012) defined patterns of relation by parsing the dependencies in open information extraction. Words around the entity mentions in sentences give clues to characterise the semantics of a relation (Niu et al., 2012; Hoffmann et al., 2011; Yao et al., 2011; Riedel et al., 2010; Mintz et al., 2009). In addition to linguistic information, collective information about the entities and their relations were exploited for RV (Rahman et al., 2018) by adding features based on a graph of entities and for RE by Augenstein (2016) that integrated global information about the object of a relation. The latter model shows the importance of adding information about the entities in the triplet. The above approaches rely on Natural Language Processing (NLP) tools for syntactic analysis and on lexical knowledge for identifying triggers. Thus, it remains difficult to overcome the lexical gap between texts and relation names when learning relation patterns for different types of relations in an open domain.

Recently, end-to-end neural network (NN) based approaches have been emerged and getting lots of attention for the relation classification task (dos Santos et al., 2015; Nguyen and Grishman, 2015; Vu et al., 2016; Dligach et al., 2017; Zheng et al., 2016; Zhang et al., 2018). However, they do not leverage any triplet representation of a relation for better understanding the relatedness between the text and the triplet. A lot of NN models for evaluating the similarity of two sentences have been proposed. They encode each entry by a CNN or an RNN (e.g., LSTM or BiLSTM), and compute a similarity between the sentence representations (Severyn and Moschitti, 2015) or compute interactions between the texts by an attention layer (Yin et al., 2016).

Most recent models encode one or two sentences by using the pre-trained neural models. Their use in RC has been successfully tested by Baldini Soares et al. (2019) where entities are marked and the sentence representation is used. Then a simple but effective sequence classification is performed using the sentence representation token which encodes the full sentence including the marked tokens. Their performances are boosted by using more documents in an unsupervised fashion. Despite more information being used, Baldini Soares et al. (2019) do not use an explicit relation representation. In an effort to cope with this problem, we explore the use of pre-trained neural models into the RV problem by explicitly using a triplet-sentence representation.

3 Relation classification via relation validation

Our proposal first learns how to validate relations ground on a sentence-triplet representation in order to predict if a relation stands or not in a sentence. To do so, our model is based on a pre-trained BERT model for sequence classification (Devlin et al., 2018). Using pre-trained models to address RC is

[2]We are aware that our model mainly based its improvements on input modification. However, we strongly believe that this is unfairly underestimated in the field.

a promising strategy as shown by Baldini Soares et al. (2019). In both cases, i.e. RV or RC, a major consideration is the input definition to correctly identify the target entities, mainly because pre-trained models do not include this option by default. In this section, we present the details of the architecture together with the input transformations to correctly feed a sequence classification model such as BERT.

3.1 BERT-based Architecture

We opted for a simplified version[3] of the architecture proposed in Baldini Soares et al. (2019) for relation classification, namely $BERT_{EM}$. It is based on fine-tuning of a pre-trained transformer called BERT (Devlin et al., 2018) where an extra layer is added to make the classification of the sentence representation, e.g. a classification task is performed using as input the [CLS] token. As reported by Baldini Soares et al. (2019), an important component is the use of mark symbols to identify the entities to classify.

3.2 Relation Classification

3.2.1 Problem definition

Given a tokenised sentence S = "t_1 t_2 ... t_n", an origin offset $o_o \in 1, n$, a target offset $o_t \in 1, n$, and a set of k relations $R = \{r_1, r_2, ..., r_k\}$. The relation extraction problem consists in determining which relation $r_p \in R$ stands in the sentence between the tokens in positions o_o and o_t, respectively.[4]

3.2.2 Input considerations

We follow the input considerations for RC proposed by (Baldini Soares et al., 2019). Thus, to introduce those markers, the original input of RC models

$$\text{input(S)} = \text{[CLS]} \quad t_1 \quad t_2 \quad \dots \quad t_n \quad \text{[SEP]} \tag{1}$$

is modified to include the entities markers

$$\text{input}'\text{(S)} = \text{[CLS]} \quad ...\$ \quad t_{o_o} \quad \$ \quad ...\# \quad t_{o_t} \quad \#... \quad \text{[SEP]} \tag{2}$$

Note that $\text{length}(\text{input}'(\text{S})) = \text{length}(\text{input}(\text{S})) + 4$, because we added the tokens $\$$ and $\#$ twice.

[3]We used the EntityMarkers[CLS] version. Other configurations were not explored and are left for future work.

[4]Note that a *non-relation* or *other relation* may be part of the set R.

3.3 Relation Validation

3.3.1 Problem definition

Given a tokenised sentence S = "t_1 t_2 ... t_n", an origin offset $o_o \in 1, n$, a target offset $o_t \in 1, n$, and a triplet $t =< t_{o_o}, r, t_{o_t} >$. The relation validation problem consists in determining whether the relation r between t_{o_o} and t_{o_t} is supported by the sentence S or not.

3.3.2 Input considerations

We transform triplets $t =< t_{o_o}, r, t_{o_t} >$ into a sequence of its label words. Then we use the sentence S on one side and the triplet t on the other side as input of the model to match the relation validation problem into a text entailment setup as suggested by Wang and Neumann (2008). So, in this case, the input is modified to

$$\begin{aligned}\text{input}''\text{(S)} = \text{[CLS]}...\$ \quad t_{o_o} \quad \$ \\ \dots \quad \# \quad t_{o_t} \quad \#...\text{[SEP]} \\ t_{o_o} \quad t_{o_t} \quad r_{w_1} \quad r_{w_2} \dots \quad r_{w_m}\text{[SEP]}\end{aligned} \tag{3}$$

Note that $\text{length}(\text{input}''(\text{S})) = \text{length}(\text{input}(\text{S})) + 4 + (m + 2)$, because of the tokens $\$$ and $\#$, and the triplet t is represented by $m + 2$ tokens (m words for the relation r and the two entities tokens). This architecture is possible because of the single or double input capabilities of transformer architectures such as BERT. Our proposed architecture is depicted in Figure 1. As for RC, we add the mark symbols in the sentence but not for the triplet. The final prediction is based on the sentence representation or the [CLS] token.

As our work focuses on relation extraction, a prior stage is needed to transform any relation classification data set into a relation validation one (i.e. as many examples as relations/classes). This transformation consists in generating $|R|$ relation validation examples for each relation extraction one, by considering the correct relation as positive and others as negatives. In this case, if $\mathcal{S}$ is the set of examples for RC, then the set of examples for RV ($\mathcal{S}_{\mathcal{RV}}$) is $|R|$ times larger than $\mathcal{S}$. However, to prevent imbalance, negative sampling is commonly used. In this case, $|\mathcal{S}_{\mathcal{RV}}| = (ns + 1) \times |\mathcal{S}|$ where ns is the number of negative examples used to build $\mathcal{S}_{\mathcal{RV}}$.

3.4 Validation of a classification prediction

Our main contribution is the definition of a new model for RC using RV, namely *BERT+RC+RV*.

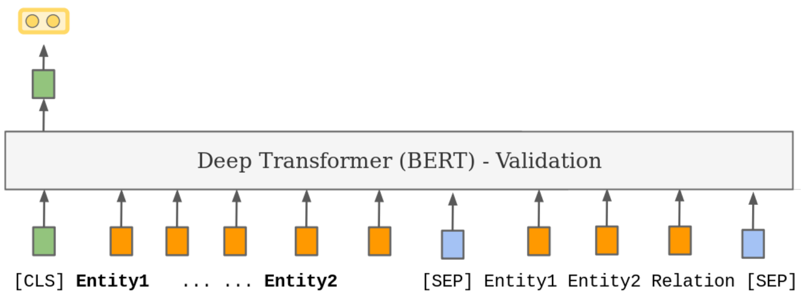
Deep Transformer (BERT) - Validation
[CLS] Entity1 ... ... Entity2 [SEP] Entity1 Entity2 Relation [SEP]

Figure 1: Our relation validation model. Tokens in bold are marked using "$" for the Entity1 and "#" for the Entity2.

During training time our RV model behaves as described in Algorithm 1. The set $\mathcal{S}_{\mathcal{RV}}$ used as input is built as described in Section 3.3.2. `createInput` generates an input such as in Equation 2. The output is a relation validation model (M_{RV}) capable of detecting if the input is valid or not.

```
Algorithm 1: BERT+RC+RV train
  Input: Set of examples $\mathcal{S}_{\mathcal{RV}}$ {Sentence ($S$), triplet ($t$), label ($l_{RV}$)}
  $epoch = 1$
  while $epoch < max_{epochs}$ do
      for $S, t, l_{RV} \in \mathcal{S}_{\mathcal{RV}}$ do
          input''($S$) = createInput($S$,$t$)
          update with Loss(input''($S$),$l_{RV}$)
  Output: Validation model ($M_{RV}$)
```

On the other hand, at inference time not all cases are evaluated. Our model can use as input the outputs of multiples RC models[5] ($\mathcal{S}_v$) as described in Algorithm 2. Each example in $\mathcal{S}_v$ is composed of a sentence and n_{RC} labels predicted by n_{RC} RC models, i.e. each example has a list (L) of n_{RC} predictions. Thus, our RV model defines the most suitable label based on the sentence and the triplet together instead of a classic RC model that only uses the sentence. `getTriplet` is a function based on a simple dictionary that returns the relation words ($r_{w_1}, ..., r_{w_m}$) related to a label l_{rc} and the entities (t_{o_o} and t_{o_t}) in S. This way, our model is not only capable of learning from the same data but also capable of aggregating multiple RC predictions.

```
Algorithm 2: BERT+RC+RV prediction
  Input: Set of examples to validate $\mathcal{S}_v$ {Sentence ($S$), labels ($L$)}, a Validation model ($M_{RV}$)
  $l_V = [\,]$
  for $S, L \in \mathcal{S}_v$ do
      $l_{i-valid} = [\,]$
      for $l_i \in$ unique($L$) do
          $t$ = getTriplet($l_i$,$S$)
          input''($S$) = createInput($S$,$t$)
          confid = predict($M_{RV}$, input''($S$))
          $l_{i-valid}$. append($l_i$, confid)
      $l_V$.append(labelMaxConfidence($l_{i-valid}$))
  Output: List of predictions ($l_V$)
```

4 Experiments and Results

4.1 Data Sets

In this study, we experimented on two publicly available data set: *SemEval10*[6] and *TACRED*[7]. Statistics of these standard relation classification data sets are presented in Table 1. We created a relation validation version from both data sets as described in Section 3.3.2. The input of our RV model needs a set of relation words which, originally, are not present in the data sets. Thus, to obtain these words, we used a rather simple strategy that consists of tokenising the relations names and using them as relation words. If needed it considers the relation direction by reversing the position of the tokenised words. Table 2 shows some examples of the selected words.

In both cases, we used the respective official F_1 metric[8] for evaluation.

[5]In our experiments, we used the outputs of our implementation of a state-of-the-art RC model, $BERT_{EM}$, described in Section 4.2.

[6]Task 8 (Hendrickx et al., 2010) from http://semeval2.fbk.eu/semeval2.php?location=tasks

[7]https://nlp.stanford.edu/projects/tacred/

[8]Macro-F1-measures are calculated using each script. Both scripts exclude the *other* class during evaluation.

Data set	Train	Dev	Test	# Relations
SemEval10	8000	-	2717	19
TACRED	68124	22631	15509	42

Table 1: Summary of SemEval10 and TACRED data sets for relation classification.

4.2 Implementation details

We implemented $BERT_{EM}$ (*EntityMarkers[CLS]* version) of Baldini Soares et al. (2019) for RC and adapted it to perform RV[9]. For SemEval10, we used 10% of training data as validation data which allows fair comparison against previous works. A maximum number of epochs was fixed to 5 and the best epoch in validation used for prediction[10]. Negative sampling was fixed to 10 where the input sentence remains and the entities remain the same but the words used for the relation representation ($r_{w_1}, r_{w_2}, ..., r_{w_m}$) are sampled from other classes. *Binary Cross Entropy* was used as loss function, Adam as optimiser, *bert-base-uncased*[11] as pre-trained model, and other parameters were assigned following the library recommendations (Wolf et al., 2019).[12] The final layer is composed of as many neurons as classes in each data set for RC and equal to two for RV (negative or positive).

Data set	Relation	Words
SemEval10	Cause-Effect(e1,e2)	Cause, Effect
	Cause-Effect(e2,e1)	Effect, Cause
	Content-Container(e1,e2)	Content, Container
TACRED	org:founded_by	org, founded, by
	per:city_of_death	per, city, of, death
	per:age	per, age

Table 2: Examples of words used per relation.

4.3 Results

Average and best result of 5 runs of our implementation of (Baldini Soares et al., 2019) using the SemEval10 data set are presented in Table 3 ($BERT_{EM}$*). The reported results are within the values reported in the original paper for this configuration, but we used *bert-base-uncased* instead

[9]Our code is publicly available at https://github.com/jgmorenof/rcviarv2020.

[10]Our models got the best validation performances at epoch 5, no further epochs were explored.

[11]https://github.com/google-research/bert

[12]We did not perform parameters search.

	SemEval10	TACRED
$BERT_{EM}$* - average	87.03	65.50
$BERT_{EM}$* - best	87.70	66.02
BERT+RC+RV - average (ours)	88.36	66.20
BERT+RC+RV - best (ours)	88.44	67.48
$BERT_{EM}$* *- voting*	89.02	68.67
BERT+RC+RV - voting (ours)	**89.41**	**69.13**
TRE (Alt et al., 2019)	87.1	67.4
BERT-LSTM-base (Shi and Lin, 2019)	-	67.8
C-GCN+PALSTM (Zhang et al., 2018)	-	68.2
C-AGGCN (Guo et al., 2019)	-	68.2
Att-Pooling-CNN (Wang et al., 2016)	88.0	-
Entity-Aware BERT (Wang et al., 2019)	89.0	-
KnowBert-W+W (Peters et al., 2019)	89.1	<u>71.5</u>
R-BERT (Wu and He, 2019)	89.25	-
$BERT_{EM}$ *(Baldini Soares et al., 2019)*	89.2	<u>70.1</u>
Span-BERT (Joshi et al., 2019)	-	<u>70.8</u>
$BERT_{EM}$*+MTB (Baldini Soares et al., 2019)*	<u>89.5</u>	<u>71.5</u>
EPGNN (Zhao et al., 2019)	<u>90.2</u>	-

Table 3: Results of official F_1 metric for the SemEval10 and TACRED data sets. Best result of our tested models is marked in **bold**. Results that outperform our method are <u>underlined</u>. '*' indicates that the result was obtained by our implementation of (Baldini Soares et al., 2019). Other values were taken from referenced papers.

	Number of candidates					
	2		3		4	
	Corr.	Incorr.	Corr.	Incorr.	Corr.	Incorr.
BERT+RC+RV	338	154	37	52	2	4
	68.69%	31.30%	41.57%	58.42%	33.33%	66.66%

Table 4: Percentage of correct (Corr.) and incorrect (Incorr.) predictions from RV model for the SemEval10 data set grouped by the number of candidates provided by RC.

	Epoch				
	1	2	3	4	5
BERT+RC+RV	0.8790	0.8807	0.8793	0.8802	0.8831
$BERT_{EM}$* - run1	-	-	-	-	0.8760
$BERT_{EM}$* - run2	-	-	-	-	0.8683
$BERT_{EM}$* - run3	-	-	-	-	0.8688
$BERT_{EM}$* - run4	-	-	-	-	0.8770
$BERT_{EM}$* - run5	-	-	-	-	0.8614

Table 5: Performances for one run of our method vs $BERT_{EM}$ runs in terms of F_1 using the SemEval10 data set. We calculated our results by epoch after training.

of *bert-large-uncased* due to computational constraints. In both cases, for average and best, our results using the relation validation model outperform their counterparts by a non-negligible margin. In order to understand the cases in which *BERT+RC+RV* makes the right prediction, we have reported the percentage of correct and incorrect predictions grouped by the number of candidates in Table 4. Note that at this stage *BERT+RC+RV* does not consider the number of predictions made for a candidate (as is made by voting) but analyse each candidate independently of its popularity. Although we used 5 runs, none of the examples obtained five candidates as for every test example at least two models predicted the same class. The number of correct predictions made by our validation model is 68.69% when there are only 2 candidates but decreases as the number of candidates increase (down to 33.33% for 4 candidates). However, in most of the cases, the predictions of the relation classification model only get 2 candidates (83.81%). Clearly, this result shows that there is still room for improvement by proposing better RV models.

Following this direction, we apply majority voting[13] over the predictions of $BERT_{EM}$ and *BERT+RC+RV*. Results are included in Table 3. Note that voting benefits our baseline but also our method by a similar margin. The lower part of Table 3 allows comparing our results to those of the most recent RC models. The best result, giving an F_1 score of 0.8941 is obtained based on majority voting of the prediction from the RV model. When compared against results reported in SemEval10, our method achieves the third position slightly behind $BERT_{EM}$+MTB, but quite far from *EPGNN* (Zhao et al., 2019). However, *BERT+RC+RV* remains an easy-to-implement model as no special modification is needed when compared with $BERT_{EM}$+MTB which uses extra auto-supervised training plus a larger model[14] and *EPGNN* which needs graph embeddings. Moreover, we believe that $BERT_{EM}$+MTB can be improved if more robust models are validated.

We also studied the performance of our method by epoch, as reported in Table 5. Results of $BERT_{EM}$* are presented for epoch 5 as this epoch got the best validation result. Note that our method outperforms all individual RC predictions from the first epoch and no underperformance is observed across epochs. This result suggests that our method is an effective way to mixture RC predictions.

Finally, we experimented with our model using the TACRED data set. Results are reported in Table 3. The results follow the same pattern as with the SemEval10 data set, except for one important difference: The performance obtained with $BERT_{EM}$* ($F_1 = 65.50$) is much lower than the value reported by the authors ($F_1 = 69.13$). This can be explained from the fact that the number of relations in TACRED is twice as high as in SemEval10. Subsequently, more parameters allowed a richer representation and a better starting point (+4.5 absolute points w.r.t. F_1).

5 Conclusion

In this paper, we presented a new strategy to improve the neural models for relation classification by using relation validation knowledge, i.e. the sentence-triplet representation. Experiments with two public data sets experimentally support our hypothesis. The proposed strategy enables new ways to improve existing methods as it can be easily plugged into more recent (or future) and powerful models. Future work will be focused on the use of this strategy across tasks from different (and far) domains as our relation validation architecture can validate triplets with unseen relations. This opened an interesting research direction for relation classification by focusing more on triplet-sentence representations rather than exclusively on the sentence.

Acknowledgements

This work has been partly supported by the European Union's Horizon 2020 research and innovation programme under grant 825153 (EMBEDDIA).
We also thank the anonymous reviewers for their careful reading of this paper and their many insightful comments and suggestions.

References

Christoph Alt, Marc Hübner, and Leonhard Hennig. 2019. Improving Relation Extraction by Pretrained Language Representations. In *Proceedings of AKBC*.

Isabelle Augenstein. 2016. *Web Relation Extraction with Distant Supervision*. Ph.D. Dissertation. University of Sheffield.

[13]The class that receives the highest number of votes will be chosen.

[14]*bert-large-uncased* uses three times more parameters (340 millions) than *bert-base-uncased* (110 millions).

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In *Proceedings of the 57th Annual Meeting of the ACL*. 2895–2905.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web.. In *IJCAI*, Vol. 7. 2670–2676.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia-A crystallization point for the Web of Data. *Journal of web semantics* 7, 3 (2009), 154–165.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on HLT and EMNLP*. Association for Computational Linguistics, 724–731.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of the 42nd Annual Meeting on ACL*.

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* (2018).

Dmitriy Dligach, Timothy Miller, Chen Lin, Steven Bethard, and Guergana Savova. 2017. Neural Temporal Relation Extraction. *EACL 2017* (2017), 746.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 601–610.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying Relations by Ranking with Convolutional Neural Networks. In *Proceedings of the 53rd Annual Meeting of ACL and the 7th International JCNLP*. 626–634.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. RelEx-Relation extraction using dependency parse trees. *Bioinformatics* 23, 3 (2007), 365–371.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In *Proceedings of the joint workshop on unsupervised and semi-supervised learning in NLP*. 10–18.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention Guided Graph Convolutional Networks for Relation Extraction. In *Proceedings of the 57th Annual Meeting of ACL*. 241–251.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *Proceedings of the 5th SemEval*. 33–38.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence* 194 (2013), 28–61.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of ACL*. 541–550.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529* (2019).

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th International JCNLP of the AFNLP*. 1003–1011.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. 39–48.

Feng Niu, Ce Zhang, Christopher Ré, and Jude W Shavlik. 2012. DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference. *VLDS* 12 (2012), 25–28.

Matthew E. Peters, Mark Neumann, Robert L Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge Enhanced Contextual Word Representations. In *EMNLP*.

Rashedur Rahman, Brigitte Grau, and Sophie Rosset. 2018. Impact of Entity Graphs on Extracting Semantic Relations. In *Information Management and Big Data*. 31–47.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 148–163.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR*. 373–382.

Peng Shi and Jimmy Lin. 2019. Simple BERT Models for Relation Extraction and Semantic Role Labeling. *arXiv preprint arXiv:1904.05255* (2019).

Mihai Surdeanu and Heng Ji. 2014. Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In *Proc. TAC*.

Vidhoon Viswanathan, Nazneen Fatema Rajani, Yinon Bentor, and Raymond Mooney. 2015. Stacked Ensembles of Information Extractors for Knowledge-Base Population. In *Proceedings of the 53rd Annual Meeting of ACL and the 7th International JCNLP*. 177–187.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining Recurrent and Convolutional Neural Networks for Relation Classification. In *Proceedings of the 2016 Conference of the NAACL-HTL*. 534–539.

Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. 2019. Extracting Multiple-Relations in One-Pass with Pre-Trained Transformers. In *Proc. of the 57th Annual Meeting of ACL*. 1371–1377.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation Classification via Multi-Level Attention CNNs. In *Proceedings of the 54th Annual Meeting of the ACL*. 1298–1307.

Rui Wang and Günter Neumann. 2008. Relation validation via textual entailment. *Ontology-based information extraction systems (obies 2008)* (2008).

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:1910.03771* (2019).

Shanchan Wu and Yifan He. 2019. Enriching Pretrained Language Model with Entity Information for Relation Classification. In *CIKM*.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the Conference on EMNLP*. 1456–1466.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics* 4 (2016), 259–272.

Dian Yu, Hongzhao Huang, Taylor Cassidy, Heng Ji, Chi Wang, and et al. 2014. The Wisdom of Minority: Unsupervised Slot Filling Validation based on Multi-dimensional Truth-Finding, In Proceedings of 2014 International CICLING.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph Convolution over Pruned Dependency Trees Improves Relation Extraction. In *Proceedings of the 2018 Conference on EMNLP*. 2205–2215.

Yi Zhao, Huaiyu Wan, Jianwei Gao, and Youfang Lin. 2019. Improving Relation Classification by Entity Pair Graph. In *ACML*. 1156–1171.

Suncong Zheng, Jiaming Xu, Peng Zhou, Hongyun Bao, Zhenyu Qi, and Bo Xu. 2016. A neural network framework for relation extraction: Learning entity semantic and relation pattern. *Knowledge-Based Systems* 114 (2016), 12–23.