Semantic Dependency Graph Convolution for Relation Extraction

Koen A. Hanneman
Bachelor Thesis 7.5 ETCS
bachelor Kunstmatige Intelligentie
Utrecht University

Meaghan Fowlie, Kees van Deemter
Utrecht University

02-02-2020

Abstract

Although in relation extraction, dependency based models are currently outclassed by a variety of models using different techniques, these dependency models have shown promising results. Here syntactic dependency trees are predominantly employed. However, consequently it is uncertain what performance semantic dependency graphs offer in comparison. Therefore we propose a modified graph convolutional network for relation extraction to work with semantic dependency graphs instead of syntactic dependency trees. The performance of this model is tested on the TACRED dataset, where for each entry in this dataset semantic dependency graphs are generated with a state-of-the-art model. Certain sentences within the set were not included for this study, as these could not be parsed. The results then of this model show increased performance when presented less training data, and equal performance when the amount of data managed to increase.

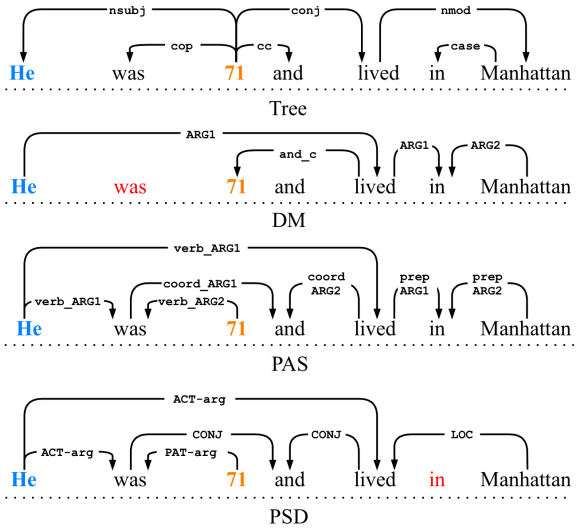

Figure 1: Example parsing on a sentence within the TACRED database, located above and below respectively. The subject ("He") and object ("71") are highlighted, as are the disconnected words ("was", "in") in the graph parsing.

1 Introduction

Relation extraction is the task of classifying the semantic relationship between known entities located in a text, for example in the form of a sentence. These, often two or more, entities are of some general type. (e.g. person, organization, fruit, city) And the to be extracted semantic relationship, between these entities, is also generalized in a category that fits the type (e.g. father-of, works-at, container-for, lives-in). Extracting semantic relations is thus a way of translating text into abstracted structured information which can be used in a range of applications. For example ontology construction (Hearst, 1992; Caraballo, 1999), knowledge base population (Zhang et al., 2017), question answering (Lin and Pantel, 2001; Lopez et al., 2005), and information collection (Voorhees, 1994). Achieving relation extraction can be done through learning relations via machine learning. For this a variety of models exist. Some of these models can be roughly categorized (Ruder, nd) as dependency-based models (Zhang et al., 2018; Cai et al., 2016; Yan Xu et al., 2016). This type has shown to have successful implementations for relation extraction.

Models that are dependency-based derive functionality from some category of directed connections between words, which are generally representative of the relation between word positions in a text. The strength of dependency-based models is that it uses as a feature this structure of dependency relations between words, which are not visible by just examining one sentence. In the way we arrange words, to for example construct a sentence, we operate according to numerous complicated rules in a languages syntax and seman-

tics rule set. Thus follows, that these rules create the relations between words, or the structure of a sentence. This is obvious to any user of the language but is not self-evident from examining just one sentence for a non-user, or in this case a program. So, by providing the dependency relations between words to a model, there is access to a path of multiple relations going from word to word that links two specified entities together. This path of relations can thus be used in the model as a feature for learning the classification of a semantic relation type between two specified entities.

Most dependency-based models (Zhang et al., 2018; Cai et al., 2016; Yan Xu et al., 2016) use syntactic dependency trees, which represent the grammatical relations between words in a sentence. The relations (e.g. prepositional, circumstantial, determinative) consist of a head and dependent argument, where the head is the central organizing word of a larger group of dependent words (Jurafsky and Martin, 2018). Each word has exactly one incoming connection, and can have multiple outgoing connections.

Less used are semantic dependency graphs in relation extraction, which represents the semantic relations between words. This can be defined as the meaning of one word being a predicate and the meaning of another word being an argument of it (Meluk, 2003). So, semantic relations (e.g. Argument, Extension) represent what words in a sentence mean to each other. The difference between syntactic and semantic dependency type lies thus in the content of the relations between words.

However, with this difference in content there comes also difference in dependency graph structures because of the restrictions each type puts on the possible relations (Meluk, 2003). In Figure 1 the difference between these dependency types is illustrated with an example. Here it is clear that syntactic parsing is able to connect all words in a tree structure, which is the case with all sentences this is applied to. With graphs not all words are connected, and the structure is different because some words have more than one incoming edge.

With the goal of relation extraction being to extract the semantic relationships between two entities, it could be inferred that a model that uses the semantic relations between words could be very effective because this data is close to the semantic relationship the relation extraction model is already searching for. So, because the data of a semantic dependency graph is arguably more like the desired output data, of the relation extraction model, than syntactic dependency it could be hypothesized that a semantic dependency-based model would perform better over a syntactic dependency-based model.

To test this hypothesis, we will change the code of an existing syntactic dependency-based graph convolutional network model (Zhang et al., 2018) to work with semantic dependency graphs. The advantage of this process is that it allows for the creation of a direct comparison between the results of both models. And thus, can answer the question, if in the context of this graph convolutional network model, a semantic dependency graph in place of a syntactic dependency graph could improve relation extraction. The examination of this possibility also creates groundwork for further pursuing semantic dependency graph implementations in other dependency models. So, a possible useful contribution for advancement in the general field of relation extraction.

In this work we will first outline the details of the model that is used. This is followed by the method section which explains the experiments that are done with this model. And after that will come the results of said experiments and a comparison against other related models. The results will then be evaluated in the discussion section where it is brought in context of the general field. And for the final chapter there is the conclusion section.

2 Model

2.1 GCN

The model (Zhang et al., 2018) that will be modified uses as its foundation a graph convolutional network (GCN) (Kipf and Welling, 2017). This network is an adaptation of the convolutional neural network (CNN) (LeCun et al., 1998) that is build to efficiently operate on graphs. A defining design principle of the CNN is compressing, or convolving, some provided input with diverse techniques to an output that can be used in classification. This also translates to the GCN. In this network the input structure of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes $i \in \mathcal{V}$ is encoded within an adjacency matrix $A \in \mathbb{R}^{n \times n}$, where $A_{ij} = 1$ if there exists an edge $(i, j) \in \mathcal{E}$ between the nodes, else follows $A_{ij} = 0$. Below is an example adjacency matrix of the PSD semantic graph parsing in Figure 1:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

It follows that self-loops for each node is represented by a $n \times n$ identity matrix I. Which can be added in $\tilde{A} = A + I$ to include every node in $\mathcal{V}$ not already in $\mathcal{E}$, that allows multiplication with A to also sum up all feature vectors of these nodes.

The diagonal node degree matrix D shows on the diagonal the amount of outgoing connections, or degree, of each node. Which is the row-sum of the adjacency matrix $D_i = \sum_{j=1}^{n} A_{ij}$. This matrix used to normalize in A the bias towards high-degree nodes. Below is the example diagonal node degree matrix of the same sentence earlier:

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

For each node i at layer l in an L-layer GCN (Figure 2) with the input vector $h_i^{(l-1)}$ and the output vector $h_i^{(l)}$ the layer-wise propagation rule is:

$$h_i^{(l)} = \sigma(\sum_{j=1}^{n} \tilde{A}_{ij} W^{(l)} h_j^{(l-1)} / D_i + b^{(l)}), \quad (1)$$

where $W^{(l)}$ is the $n \times n$ layer-specific trainable weight matrix and $b^{(l)}$ is the bias term. And σ is the rectified linear unit (ReLU) activation function. This activation has a linear output value for every positive input, and has the output value 0 for a negative input.

2.2 Dependency Graph Implementation

Originally the GCN model was adapted to work with syntactic dependency trees. The dependency trees were therefore encoded in a representational object that would aid translation to the model. Dependency trees are by definition a type of graph and therefore translate directly to the adjacency matrix. Later when bi-directionality was added in this matrix the trees become graphs, instead of a type of graph. For semantic dependency graphs the translation to the adjacency matrix is just as direct. The difference was that the representational object was changed to accommodate for multiple incoming connections per node possible in a graph, as supposed to trees which only have exactly one incoming connection.

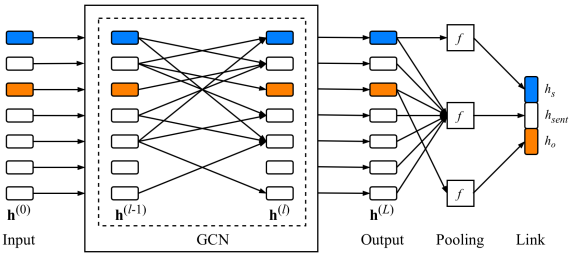

Figure 2: Relation extraction architecture. Inside the GCN module shows graph convolution computation for the example sentence with PSD parsing.

2.3 Relation Extraction

Relation extraction is intuitively defined as given some sentence $\mathcal{X}$ and two parts of $\mathcal{X}$, yield the relation between these parts. In the previously used example sentence "He was 71 and lived in Manhattan", these two parts were shown to be the *subject* "He" and *object* "71". The extraction goal for this sentence was a 'per:age' relation.

Formally defined, given some sentence $\mathcal{X} = [x_1, ..., x_n]$ and spans $\mathcal{X}_s = [x_{s_1}, ..., x_{s_n}]$, $\mathcal{X}_o = [x_{o_1}, ..., x_{o_n}]$ return relation $r \in \mathcal{R}$ finite set of relations, and return 'no-relation' otherwise. Noted are separate span lengths n where on position i token x_i is represented.

Span $\mathcal{X}$ is for the L-layer GCN translated to an equal span of GloVe (Pennington et al., 2014) input vectors $\mathbf{h}^{(0)}$. At layer L the output vectors $\mathbf{h}^{(L)}$ are the collective token hidden representations (Figure 2). Each token has a connection towards every token it has a direct connection with. So after L layers there is a hidden representation for every token connection to L distance. The sentence is then represented as:

$$h_{sent} = f(\mathbf{h}^{(L)}) = f(\text{GCN}(\mathbf{h}^{(0)})), \quad (2)$$

where the max pooling function $f : \mathbb{R}^{d \times n} \rightarrow \mathbb{R}^d$ with a max filter of $1 \times n$ selects the maximum value of row d_i in $\mathbf{h}^{(l)}$ within that window and finally returns the $d \times 1$ vector. This pooling function is used another two times on $\mathcal{X}_s$, and $\mathcal{X}_o$ for

the subject and object representations:

$$h_s = h_o = f(\mathbf{h}^{(L)}_{e_1:e_n}), \tag{3}$$

where selected from $\mathbf{h}^{(L)}$ are the n output vectors that are within the entity start tag e_1 to entity end tag e_n range for the respective entity. The entity and sentence representations are then concatenated and then fed through a feed-forward neural network (FFNN) (Santoro et al., 2017; Lee et al., 2017) for the final representation:

$$h_{final} = \text{FFNN}([h_{sent}; h_s; h_o]). \tag{4}$$

This h_{final} representation is then fed into a linear layer followed by a 1-dimension softmax function that results in a probability distribution over all relations $r \in \mathcal{R}$ with finally and argmax function for the relation predictions.

3 Method

3.1 Data

Experiments are run on the TAC Relation Extraction Dataset (TACRED) (Zhang et al., 2017), as this is the dataset the original model is built for. It contains 106k entity pairs that are drawn from the yearly TAC Knowledge Base Population (Getman et al., 2018) challenge. Each classified relation type between entities is part of a set of 41 relations, or is a 'no-relation' type for when the relation between entities is not part of the relation set and possibly does not even exist. Entities in TACRED are categorized as *subject* and *object*. Here subjects can be of a person or organization type, and objects are divided over 16 different types (e.g., date and location).

Also present in this dataset is information such as part of speech tagging, and syntactic dependency relations for each sentence. However, the dataset does not contain the necessary information that is needed for the semantic dependency graphs of these sentences. Instead these graphs are constructed by a separate program and added to the TACRED dataset in order for the graph data to be used in the GCN model.

3.2 Dependency Graphs

The semantic dependency graphs are constructed with the ACL 2019 parser (Lindemann et al., 2019). These graphs come in three distinct representations named DELPH-IN MRS-Derived Semantic Dependencies (DM), Enju Predicae-Argument Structures (PAS), and Prague Semantic Dependencies (PSD). The representations differ from each other in labeling scheme, directionality of edges, and in general the edges placed between words. This is result of the different design decisions and techniques that are responsible for the creation of these annotations, and between them there is currently no obvious answer which one is truly better (Oepen et al., 2015). So, to test if semantic dependency graphs are effective in a GCN model all three representations are used to sufficiently explore this subject.

However, at this time, it was unfortunately not possible for us to effectively parse all sentences of the TACRED dataset. For longer sentences current the parser will hit a run-time limit because of the fixed-tree decoder that is used, which takes exponential time with the number of children of any node. Graphs of longer sentences have by definition the opportunity for a larger amount of interconnected words, which enables parsing to reach its maximum time limit. As a result these sentences are paired with empty graphs with no semantic dependency data available. And because these graphs are essential our model to work this means that long sentences could not be used in training and evaluating, so these were removed from the set.

The original TACRED training, development, and test datasets consist of 68124, 22631, and 15509 sentences respectively. The semantic dependency graph parsing was successful for 98.5%, 99.5%, and 99.8% sentences respectively, a total decrease of 1.1% sentences. After this a new parsing model was constructed for specifically PSD-type graphs as these were at the time expected to be the best performing type. This parser was able to result success for 99.0%, 99.7%, and 99.8% sentences respectively, a total decrease of 0.8% sentences. Because of the partial set it follows that the performance results of following experiments will in some way be impacted, positive or negative. It is hard to say if this performance shift will be significant, but is to be noted going forward.

3.3 Evaluation

As is standard, the model will be evaluated with micro-averaged F_1 scores on the TACRED dataset. This is calculated with the precision P and recall R of the model in the harmonic average F_1.

Model	Dev F_1	P	R	F_1
Tree:B/S	61.9	64.5	42.1	50.9
DM:B/S	61.4	**71.7**	40.7	51.9
PAS:B/S	61.1	68.5	44.9	54.2
PSD:B/S	60.3	65.7	46.2	54.2
DM:B	**63.2**	67.6	**56.0**	**61.2**
PAS:B	62.4	65.8	56.0	60.5
PSD:B	62.3	67.3	53.8	59.8

Table 1: Results of the first experiment on TACRED. Bold marks the highest number among current models. Letters B and S indicate if bidirectional relations or self-loops are active.

Model	Dev F_1	P	R	F_1
1:Tree:B/S	61.9	64.5	42.1	50.9
1:PSD:B/S	60.3	65.7	46.2	54.2
1:PSD:B	62.3	67.3	53.8	**59.8**
2:Tree:B/S	63.0	64.1	54.9	59.1
2:PSD:B/S	60.7	68.8	48.6	56.9
2:PSD:B	63.2	67.2	52.9	**59.2**

Table 2: Results on different size TACRED datasets. Numbers 1 and 2 show experiment numbers. Bold marks the highest number amongst the respective set. Letters B and S indicate if bidirectional relations or self-loops are active.

Our GCN model has no graph pruning so therefore we compare results to the baseline no pruning tree-based GCN model for a balanced analysis. This is chosen as comparing to results without tree pruning can be an approximation of performance where pruning to be implemented. Additionally when pruning is enabled training takes an amount of time that is not possible to currently be feasible. The time cost occurs from the pruning algorithm which is activated with each instance of loading a syntactic dependency tree, instead pruning beforehand and referencing already pruned trees. It should be noted however that with no pruning the models are not performing as optimally as with $K = 1$ pruning.

4 Results

4.1 Experiment 1

The first tasks goal is to compare each grammar type to the performance given when using trees on a level playing field. For this experiment the GCN model parameters are set to be equal with the settings provided with the original model; with self-loops, and bi-directionality. In early parameter configuration testing with semantic dependency graphs, promising results without self-loops active seemed to appear. These are also included in the experiment. As with the initially expected under performance of graph based models, and the inclusion of self-loops being based on aiming for an optimal tree based model.

The results in Table 1 show that each graph model outperformed the baseline tree model, regardless of the setting variations. Shown are the development set F_1 and test set precision, recall, and F_1. With the DM-type graph without self-loops reaching the best score by 0.7 F_1. And widely shown is that the graphs without self-loops exceed their self-loops included counterparts. Hypothetically considered this could be because of the words that were initially ignored by the semantic dependency graph parsing (Figure 1). Including these words through self-loops could then have lowered prediction accuracy. Increasing prediction accuracy for syntactic dependency trees by implementing pruning, also comes from the same principle of trimming irrelevant words.

The tree-based model was previously reported to have an estimated score of around 63.5 F_1 on the development set; the score was not provided in exact terms and only appeared in a graph. This is a decrease of 1.6 F_1 score to the current result of 61.9. Because the only variable changed in respect to the tree-based model is the amount of sentences, it can be concluded that this change must have been the cause. Evaluation score of the tree model without pruning was unfortunately not reported, so the full impact of sentence removal can not be measured. As the development set score roughly indicated performance on the evaluation set, the unknown score on this set is also expected to increase with an increase in development set score.

4.2 Experiment 2

For the second experiment we were able to increase the amount of sentences of the TACRED database that we were able to parse PSD-type semantic dependency graphs for. The goal here is to see what effect unavoidable database filtering had on comparable models results in the previous experiment. In Table 2 the all results of this experiment can be seen together. Compared to the previous experiment the model with PSD graphs

presented similar results; the new sentences seem to have had no effect on performance.

The tree based model raised previous performance with an increase of 1.1 F_1 development set score to 63.0 F_1. Following an inequality of 0.5 F_1 with the original reported 63.5 F_1 score. The remaining relatively slight difference is possibly the consequence of the last absent sentences in the database, assuming the score discrepancy would decrease at a similar rate to what these results present. As a result the hypothesized evaluation score increase proved to be 8.1 to 59.1 F_1. Performance growth, in this case of syntactic dependency trees, then suggests to be related to the amount of long sentences in the collective database.

5 Analysis

5.1 Discussion

Shown in the results section, the use of semantic dependency graphs in place of syntactic dependency trees does see a form of change in performance on the relation extraction task. But if this change is positive or negative seems to be conditional, evident by the variation of results between both previous experiments. When the average sentence length is reduced equal graph models outperformed the tree based model with a difference of 8.1 F_1 score. But when the average was increased the difference between the two reduced to an equal score, with graphs remaining consistent in task performance. This is however not currently proven to reflect every semantic dependency graph based model from the first experiment, just the PSD-type graph, but or ease of comparison consistency among semantic dependency graphs performance will be assumed.

The improvements gained by syntactic dependency trees by increasing the average sentence length can not be fully encompassed by the total 0.05% more sentences that were added to the test dataset, were trees to get a perfect score on all these entries. Therefore, the hypothesis could be that trees learn better from longer sentences because with relative large word counts there are more opportunities to learn connection patterns. Syntactic dependency trees have been shown to reliably connect every word in sentences together within their tree structure, these patterns seem then to actually improve classification on shorter sentences. Or it could be that the 0.47% sentences added to the training set in the second experiment happened to be crucial for learning, regardless of sentence length.

Semantic dependency graphs stay consistent regardless of the average sentence length in the database. The difficulty of classifying relations of longer sentences seems intuitively harder than shorter sentences, because fewer words should present less points of failure. Which in this case would be connections between words that distract from critical connections. This also follows from the fact that pruning words had a positive impact on syntactic dependency trees. The hypothesis for this consistency is that semantic dependency graphs, as shown earlier in Figure 1, already does not connect irrelevant words. This would mean that longer sentences do not give nearly as much distracting words, thus semantic dependency graphs remain consistent.

5.2 Future Work

For future studies a possible exploration of the topic could be to compare the behaviour of syntactic dependency trees and semantic dependency graphs in different models. The current effectiveness of semantic dependency graphs are only tested with the current GCN implementation. This could be to change it for a Simple Graph Convolution (SGC), or an entirely different model that could learn from graphs. Another change could be made to the semantic dependency parser. A parser that could efficiently parse all TACRED sentences can definitely give better insight into the current findings of this study.

Pruning on graphs could also be considered. Semantic dependency graphs already have shown to inherently prune words. And while there are less words to prune, a stronger pruning has the possibility for performance improvements. Implementing a minimum spanning tree algorithm to semantic dependency graphs as the replacement for pruning is then a further option. For this to work connection weights should be introduced with some sort of defined heuristic, but this is then a reasonably experimental option.

6 Conclusion

Shown was that semantic dependency graphs provide a consistent performance with data constructed of varied sentence length. And that syntactic dependency trees without pruning decrease

in performance when sentence length is decreased. This does not confirm the initial proposed hypothesis of semantic dependency graphs being strictly superior. Thus follows the implication that, depending on the application, the consistent performance of semantic dependency graphs could be considered an improvement or a deterioration over syntactic dependency trees. Or more broadly, there is promising data for semantic dependency graphs but further research must follow to fully encompass the potential.

References

Cai, R., Zhang, X., and Wang, H. (2016). Bidirectional recurrent convolutional neural network for relation classification. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 756–765.

Caraballo, S. A. (1999). Automatic construction of a hypernym-labeled noun hierarchy from text. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

Getman, J., Ellis, J., Strassel, S., Song, Z., and Tracey, J. (2018). Laying the groundwork for knowledge base population: Nine years of linguistic resources for tac kbp. *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. *Proceedings of COLING-92*, pages 539–545.

Jurafsky, D. and Martin, J. H. (2018). *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, New Jersey.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR 2017)*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11):22782324*.

Lee, K., He, L., Lewis, M., and Zettlemoyer, L. (2017). End-to-end neural coreference resolution. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Lin, D. and Pantel, P. (2001). Discovery of inference rules for question answering. *Natural Language Engineering, 7(4):343-360*.

Lindemann, M., Groschwitz, J., and Koller, A. (2019). Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy. Association for Computational Linguistics.

Lopez, V., Uren, V., Motta, E., and Pasin, M. (2005). Aqualog: An ontology-driven question answering system of organizational semantic intranets. *Proceedings of the 2nd European Semantic Web Conference*.

Meluk, I. A. (2003). Dependency in linguistic description. *unknown*.

Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Cinkov, S., Flickinger, D., Haji, J., and Ureov, Z. (2015). Semeval 2015 task 18: Broad-coverage semantic dependency parsing. *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 915–926.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Ruder, S. (n.d.). Relationship extraction. Retrieved from http://nlpprogress.com/english/relationship_extraction.html.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. *Advances in neural information processing systems*, page 4974 4983.

Voorhees, E. M. (1994). Query expansion using lexical-semantic relations. *The seventeenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 61–69.

Yan Xu, R. J., Lili Mou, G. L., Chen, Y., Lu, Y., and Jin, Z. (2016). Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint*, arXiv:1601.03651.

Zhang, Y., Qi, P., and Manning, C. D. (2018). Graph convolution over pruned dependency trees improves relation extraction. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

Zhang, Y., Zhong, V., Chen, D., Angeli, G., and Manning, C. D. (2017). Position-aware attention and supervised data improve slot filling. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.