EMNLP Project 2: Semantic Dependency Parsing

Sishuo Chen
School of Electronic Engineering and Computer Science,Peking University
chensishuo@pku.edu.cn

Abstract

This report is a description of my system for SemEval-2014 Task 8:Semantic Dependency Parsing (Oepen et al., 2014) [1], as the second course project for the Empirical Methods for Natural Langugae Processing course in Peking University, spring 2020, directed by Professor Yansong Feng and Weiwei Sun. My submission is an ensemble system made up of arc-factored classification models. The system is implemented in nearly pure Python without any packages about machine learning or natural language processing. It reaches 78.32% precision and 0.6888 F1 score on the test set provided by Oepen et al. (2014)[2],and the prediction results for the newly released test set (*esl.output* and *cesl.output*) has been uploaded with the report and system code.

1 Task Introduction

1.1 Semantic Dependency Parsing

Semantic Dependency Parsing (SDP) is defined as the task of recovering sentence-internal predicate–argument relationships for all content words (Oepen et al., 2014). While syntactic dependency annotations concentrate on the surface or functional structure of a sentence, semantic dependency annotations aim to capture between-word relationships that are more closely related to the meaning of a sentence, using graph-structured representations.

1.2 Data Format

The original SemEval tasks (Oepen et al., 2014) comprised three distinct target representations (over the same text), dubbed (a) DELPH-IN MRS-Derived Semantic Dependencies (DM), (b) Enju Predicate–Argument Structures (PAS), and (c) Prague Semantic Dependencies (PSD). In this project, the training and testing datasets are in the DM format.

2 Related Work

Graph-based semantic parsing approaches can be summarized as the following four categories (Koller et al., 2019): factorization-based, composition-based, transition-based and translation-based. Brief introduction about the four approaches is given below.

- **Factorization-based Approach.** It is inspired by the successful design of graph-based dependency parsers (McDonald and Pereira, 2006). As a structure prediction problem, it can be converted to a maximum subgraph parsing problem with proper scoring rules for a subgraph. The scoring rule can be first-order factorization (Kuhlmann and Jonsson, 2015) or neural-based scorers (Peng et al., 2017; Chen et al., 2018; Dozat and Manning, 2018).

- **Composition-based Approach.** Inspired by old school, rule-based approaches,it explicitly models the syntactico-semantic derivation process. It learns to find the best derivation from a large set of derivations that are licensed by a symbolic system. The derivation can be done with AM algebra (Groschwitz et al., 2017) or hyperedge replacement grammar (Peng et al., 2015).

- **Transition-based approach.** Inspired by the successful design of transition-based dependency parsers (Sagae and Tsujii, 2008),it consists of transitions and configurations that can be manipulated by the transitions. The configurations generally encode the information

[1] http://alt.qcri.org/semeval2014/task8/

[2] http://svn.delph-in.net/sdp/public/2014/test/11.tgz

of the current parsing state, especially including partial parsing results, and the transitions can be applied to a configuration, turning it into a new one. When the system reaches any acceptable configuration, a coherent semantic graph is also successfully built. The idea of greedy search is used here. Many transition-based models have been proposed and showed their good performance on the semantic dependency parsing task (Wang et al., 2015; Zhang et al., 2016; Buys and Blunsom, 2017; Gildea et al., 2018; Sun et al., 2019).

- **Translation-based approach.** It's inspired by the neural sequence-to-sequence machine translation model. Regarding the parsing graph as a foreign language and the original text as the source language, the sequence-to-sequence model can be easily used for semantic parsing problems (Konstas et al., 2017; Peng et al., 2018).

3 System Implementation

3.1 Resource Restriction

According to the course instructions this year, we can't use any packages about machine learning algorithms or natural language processing,so we can enhance out code ability and understanding about basic methods for natural language processing. Given that this is closed-track challenge,extra training data is also excluded.

So I implemented the system by nearly pure Python (*version 3.5.2*) with *numpy* [3] (Oliphant, 2006) package for accelerating vector computation and storing model parameters. All requirements are given in the *requirements.txt* file.

3.2 My Parsing System

3.2.1 Data Processing and Statistics

To unify the I/O interface, I discarded the Java tool provided by the SemEval host and implemented the I/O and scoring routines in Python. Some statistics about the traiging dataset(the *dm.sdp* file) is shown in Table 1. Figure 1 shows the distribution of the distance of two tokens in golden edges. We can observe that the token distances of most samples are less than 10.

[3]`https://github.com/numpy/numpy`

	Number
Sentences	34003
Tokens	38242
Lemmas	11876
POS Tags	45
Relations	51(+1)
%Singleton	22.62

Table 1: Statistics of the training dataset

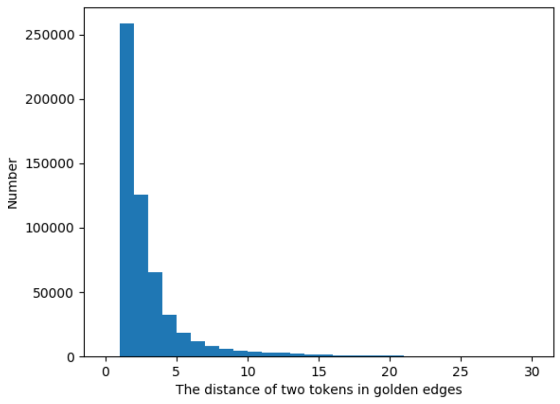

Figure 1: The distribution of the distance of two tokens in golden edges

3.2.2 The Arc-factored Model

I implemented the arc-factored model inspired by Thomson et al. (2014). It solves the semantic parsing problem as a three-stage pipeline: singleton classification $\rightarrow$ edge prediction $\rightarrow$ top classification.

1. **Singleton Classification**. Singletons are nodes that have no parents or children in the parsing graph. A singleton will be not considered for any attachments in later stages to make the system faster without affecting accuracy. For singleton prediction, we train a token-level log-linear classifier, with features including the token itself,its lemma and part of speech tag. The log-linear model is adapted from the one implemented in Project 1 with code optimization for sparse input feature. I trained the classifier for 10 epochs. Learning rate is 0.001 and batch size is 1000. No parameter regularization is used. It reaches 95.59% accuracy on the test set provided by SemEval-2014 Task 8 (Oepen et al., 2014), which is close to the logistic regression model trained by Thomson et al. (2014).

Tokens: The tokens t_i and t_j themselves.
POS tags: The part of speech tags of t_i and t_j.
Lemmas: The lemma of speech tags of t_i and t_j.
Linear Order: Fires if $i < j$.
Linear Distance: : $i - j$.
POS Context: Concatenated POS tags of tokens at $i-1, i, i+1, j-1, j, and\ j+1$.
Concatenated POS tags of tokens at $i-1, i$, $j-1, and\ j$.
Concatenated POS tags of tokens at $i, i+1$ $, j, and\ j+1$.

Table 2: Input feature design for directed edge (t_i, t_j)

The training time is only about one minute per epoch.

2. **Edge Prediction**. At this stage, the task is to predict the set of labeled directed edges in the graph. The system considers only token index pairs (i, j) where $|i - j| \leq 10, i \neq j$, and both t_i and t_j have been predicted to be non singletons by the first stage, the same as the setting in Thomson et al. (2014). For convenience, I still use the log-linear model for edge classification. The composition of the input feature is shown in Table 2. There are around 7 million edges in the training dataset(around 92% of them are non-existing edges). I trained the classifier for 10 epochs. Learning rate is 0.001 and batch size is 100k. No parameter regularization is used. It gives 95.82% accuracy for edge prediction on the test set provided by SemEval-2014 Task 8 (Oepen et al., 2014).

3. **Top classification**. I trained a separate token-level binary log-linear model to classify whether a token's node had the "top" attribute or not. Only nodes where there is at least one outbound edge (predicted by the edge classifier) will be considered. For a given token t at index i, the top classifier's features are shown in Table 3. As most sentences has only one "top",only the node with the highest prediction probability as a "top" node will be labeled "top",according to what is done in Thomson et al. (2014).

3.2.3 Model Ensemble

To improve the performance of my system, I trained 5 edge classifiers using different random seeds for training data shuffling. The performance of each model on the SemEval test set will be reported in the next section. The final submission is an ensemble system made up of them by simple voting.

Token: The tokens t itself.
Lemma: The lemma of token t.
POS tag: The part of speech tag of token t.
Index: i.
IsRoot: Fires if t is there is an incoming edge. (predicted by the edge classifier)

Table 3: the top classifier's features for a given token t at index i

4 Experiments

4.1 Experimental Setup

For there is no "lemma" column information in the test dataset given by this course and packages like *nltk* are not allowed, I implemented a lemmatization tool by creating a lemmatization dictionary from the training set and the English lemmatization list provided by Michal Měchura [4].

The scoring metrics for the final results contain labeled precision (LP), labeled recall (LR), labeled F1 (F1), and labeled whole-sentence match (EM) (Oepen et al., 2014). For there is no golden annotations for the test dataset given by the course, we will show the performance on both training and test set of SemEval-2014 Task 8 (Thomson et al., 2014) and submit the predictions for the test dataset given by this course.

Given to the huge number of dimensions of the sparse input feature for edge classification and the big amount of training data,I trained the system on a remote server instead of my PC. The hardware setting is: Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz * 16;RAM: 50GB;No GPUs. The training of singleton classifiers and top classifiers only needs several minutes,while the training of the edge classifier costs around 100 minutes per epoch.

4.2 Evaluation for a Single Model

I evaluated the the 5 seperate systems on the SemEval test set and the results are shown in Table 4.

[4] https://github.com/michmech/lemmatization-lists/blob/master/lemmatization-en.txt

Seed	LP	LR	F1	EM
123	0.7050	0.5936	0.6445	0.0178
666	0.7064	0.5874	0.6414	0.0186
789	0.7323	0.5859	0.6510	0.0178
888	0.7065	0.5954	0.6462	0.0252
999	0.7156	0.5784	0.6398	0.0178

Table 4: The performance of 5 separately trained models.(LP:Labeled Precision;LP:Labeled Recall; F1: Labeled F1 Score;EM: Exact Match)

LP	LR	F1	EM
0.7832	0.6147	0.6888	0.2222

Table 5: The performance of the ensemble system.

4.3 Evaluation for the Ensemble System

The evaluation results of the 5-classifier ensemble system is shown in Table 5. It surpasses the performance of a single model shown in Table 4 by a significant margin,showing the power of the ensemble trick.

5 Conclusion and Future Work

In this project, I developed a semantic dependency parsing system by integrating arc-factored models. Experiments on the SemEval test set show that my ensemble system is effective with labeled precision at 78.32% and F1 score at 0.6888. For the labels of the final test set is not given, I submit the predictions given by my ensemble system (*esl.output* and *cesl.output*) along with the code and report. The files storing the model parameters is too large for me to upload to the course system(1.4GB for an edge classification model), so I will public them later in a proper way for whom in need or the check by TAs,

Due to the rules of the course, fancy deep learning and NLP packages such as *Pytorch* [5] (Paszke et al., 2019) and *transformers* [6] (Wolf et al., 2019) are not to allowed to use for this project, so it's a pity that we can't apply complex neural models such as LSTM (Hochreiter and Schmidhuber, 1997) and BERT (Devlin et al., 2019) for the semantic dependency parsing task. For future work, I will try to apply popular pre-trained language models like BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) to improve the performance of my SDP system. Besides, as there is currently no popular open-source semantic parsing tools in the NLP community, I would like to develop one if possible.

[5]https://github.com/pytorch/pytorch

[6]https://github.com/huggingface/transformers

6 Acknowledgement

I would like to thank Professor Yansong Feng and Weiwei Sun for their efforts at the EMNLP course for Peking University. I also want to thank Quzhe Huang and other teaching assistants for their work. ♡

References

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226, Vancouver, Canada. Association for Computational Linguistics.

Yufei Chen, Sheng Huang, Fang Wang, Junjie Cao, Weiwei Sun, and Xiaojun Wan. 2018. Neural maximum subgraph parsing for cross-domain semantic dependency analysis. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 562–572, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Daniel Gildea, Giorgio Satta, and Xiaochang Peng. 2018. Cache transition systems for graph parsing. *Computational Linguistics*, 44(1):85–118.

Jonas Groschwitz, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2017. A constrained graph algebra for semantic parsing with AMRs. In *IWCS 2017 - 12th International Conference on Computational Semantics - Long papers*.

S Hochreiter and J Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Alexander Koller, Stephan Oepen, and Weiwei Sun. 2019. Graph-based meaning representations: Design and processing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 6–11, Florence, Italy. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: sequence-to-sequence models for parsing and generation. *CoRR*, abs/1704.08381.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics*, 3:559–570.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72, Dublin, Ireland. Association for Computational Linguistics.

Travis E Oliphant. 2006. *A guide to NumPy*, volume 1. Trelgol Publishing USA.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2048, Vancouver, Canada. Association for Computational Linguistics.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 32–41, Beijing, China. Association for Computational Linguistics.

Xiaochang Peng, Linfeng Song, Daniel Gildea, and Giorgio Satta. 2018. Sequence-to-sequence models for cache transition systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1842–1852, Melbourne, Australia. Association for Computational Linguistics.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 753–760, Manchester, UK. Coling 2008 Organizing Committee.

Weiwei Sun, Yufei Chen, Xiaojun Wan, and Meichun Liu. 2019. Parsing Chinese sentences with grammatical relations. *Computational Linguistics*, 45(1):95–136.

Sam Thomson, Brendan O'Connor, Jeffrey Flanigan, David Bamman, Jesse Dodge, Swabha Swayamdipta, Nathan Schneider, Chris Dyer, and Noah A. Smith. 2014. CMU: Arc-factored, discriminative semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 176–180, Dublin, Ireland. Association for Computational Linguistics.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado. Association for Computational Linguistics.

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics*, 42(3):353–389.