Greedy Universal Dependency Parsing with Right Singular Word Vectors

Ali Basirat, Joakim Nivre

Department of Linguistics and Philology
Uppsala University
{ali.basirat,joakim.nivre}@lingfil.uu.se

Abstract

A set of continuous feature vectors formed by right singular vectors of a transformed co-occurrence matrix are used with the Stanford neural dependency parser to train parsing models for a limited number of languages in the corpus of universal dependencies. We show that the feature vector can help the parser to remain greedy and be as accurate as (or even more accurate than) some other greedy and non-greedy parsers.

1. Introduction

Greedy transition-based dependency parsing is appealing thanks to its efficiency, deriving a parse tree for a sentence in linear time using a discriminative classifier. A major drawback to this approach of parsing is the too early choices made by the greedy parsers at each step of the parsing algorithm which usually lead to some local optima. This problem can be enhanced by providing the classifier with more informative features about the contextual environment of words to help the parser to make a better decision at each step of parsing. Chen and Manning (2014) propose to use distributional representation of words as feature vectors in a 3-layered feed-forward neural network which functions as the core classifier in a transition-based dependency parser, known as Stanford neural dependency parser. These feature vectors, also known as *word vectors*, are so distributed in a high dimensional space that the similarities between vectors mirror the syntactic and semantic similarities between words (Sahlgren, 2006).

The successful applications of the word vectors in the natural language processing tasks (e.g., (Collobert et al., 2011; Chen and Manning, 2014; Dyer et al., 2015)) encourage many researchers to extract more qualified word vectors (Mikolov et al., 2013; Pennington et al., 2014; Lebret and Collobert, 2014). In this paper we propose to extract word vectors from right singular vectors of a matrix returned by a transformation function that takes a probability co-occurrence matrix as input and expand the data massed around zero.

Using these word vectors with Stanford neural dependency parser, we trained different parsing models over English, Persian, and Swedish parts of the corpus of universal dependencies (Nivre et al., 2016). We show that the word vectors can result in parsing models which are more accurate than some other greedy and non-greedy parsers.

2. Transition-Based Dependency Parsing

A greedy transition-based dependency parser aims to derive a parse tree from a sentence by predicting a sequence of transitions between a set of configurations. The process of parsing starts from an initial configuration and ends with some terminal configurations. A configuration consists of a stack Σ to store partially processed nodes, a buffer B to store unprocessed nodes in the input sentence, and a set of partial parse trees A associated with the processed nodes. Nodes are positive integers corresponding to the linear position of the words in the input sentence and one extra artificial root node 0. The transitions between configurations are controlled by a discriminative classifier which is trained on a history-based feature model combining features of the partially built dependency tree and attributes of input tokens.

The arc-standard algorithm (Nivre, 2004) is among the many different algorithms proposed for moving between configurations. The algorithm starts with the initial configuration in which all words are in B, Σ is empty, and A holds 0. It uses three actions *Shift*, *Right-Arc*, and *Left-Arc* for transition between the configurations and to build the parse tree. Shift pushes the head node in the buffer into the stack unconditionally. The two actions Left-Arc and Right-Arc are used to build left and right dependencies, respectively, and are restricted by the fact that the final dependency tree has to be rooted at node 0.

3. Stanford Dependency Parser

Stanford dependency parser is an arc-standard system with a feed-forward neural-network as its classifier. The neural network consists of three layers: An input layer connects the network to a configuration through 3 real-valued vectors representing words, POS-tags and dependency relations. The vectors that represent POS-tags and dependency relations are initialized randomly but those that represent words are initialized by word vectors systematically extracted from a corpus. Each of these vectors is independently connected to the hidden layer of the network through three distinct weight matrices. A cube activation function is used in the hidden layer to model the interactions between the elements of the vectors. The activation function resembles a third degree polynomial kernel that enables the network to take different combinations of vector elements into consideration. The output layer generates probabilities for decisions between different actions in the arc-standard system. The network is trained by the standard back-propagation algorithm that updates both network weights and vectors used in the input layer.

4. Word Vectors

Following Lebret and Collobert (2014), we propose to extract word vectors from a co-occurrence matrix as follows. First we build a co-occurrence matrix $\mathbf{C}$ from a text. The element $\mathbf{C}_{i,j}$ is a maximum likelihood estimation of the probability of seeing word w_j in the context of word w_i, (i.e., $\mathbf{C}_{i,j} = p(w_j|w_i)$. It results in a sparse matrix whose data are massed around zero because of the disproportional contribution of the high frequency words in estimating the co-occurrence probabilities. Each column of $\mathbf{C}$ can be seen as a vector in a high-dimensional space whose dimensions correspond to the context words. In practice, we need to reduce the dimensionality of these vectors. This can be done by standard methods of dimensionality reduction such as principal component analysis, but the high density of data around zero and the presence of a small number of data points far from the data mass can lead to some meaningless discrepancies between word vectors.

In order to have a better representation of the data we expand the probability values in $\mathbf{C}$ by skewing the data mass from zero toward one. This can be done by applying any monotonically increasing concave function that magnifies small numbers in its domain while preserving the given order. Commonly used transformation functions with these characteristics are the logarithm function, the hyperbolic tangent, the power transformation, and the Box-cox transformation with some specific parameters. After applying f on $\mathbf{C}$ we centre the column vectors in $f(\mathbf{C})$ around their mean and build the word vectors as below:

$$\Upsilon = \gamma \mathbf{V}^T_{n,k} \tag{1}$$

where Υ is a set of k dimensional word vectors associated with n words, $\mathbf{V}^T_{n,k}$ is the matrix of first k right singular vectors of $f(\mathbf{C})$, and $\gamma = \lambda\sqrt{n-1}$ is a constant factor to scale the unbounded data in the word vectors. In the following, we will refer to our model as RSV, standing for Right Singular word Vector.

5. Experimental Setting

We set the RSV parameters as follows:

- transformation function: $f = \sqrt[7]{x}$
- number of dimensions in word vectors: 100

The Stanford neural parser in all of our experiments is used with 400 hidden units and maximum 20 000 training iterations. These parameters are chosen on the basis of our previous experiments on Wall Street Journal (Marcus et al., 1993). The transformation function has been chosen among many other transformation functions such as tanh, *coxbox*, and log. We have also tested word vectors with different dimensions and selected the best one.

We use the following corpora to extract word vectors for English, Persian, and Swedish. The English corpus consists of raw sentences in Wall Street Journal (WSJ) (Marcus et al., 1993), English Wikicorpus,[1] Thomson Reuters Text Research Collection (TRC2),[2] English Wikipedia corpus,[3] and the Linguistic Data Consortium (LDC) English corpus.[4] We concatenate all the corpora and split the sentences by the OpenNLP sentence splitting tool.[5] Text tokenization is performed by Stanford tokenizer.[6] Word vectors for Persian are extracted from the Hamshahri Corpus (AleAhmad et al., 2009), Tehran Monolingual Corpus,[7] and Farsi Wikipedia download from Wikipedia Monolingual Corpora.[8] The Persian text normalizer tool (Seraji, 2015) is used for sentence splitting and tokenization.[9] Word vectors for Swedish are extracted from Swedish Wikipedia available at Wikipedia Monolingual Corpora, Swedish web news corpora (2001-2013) and Swedish Wikipedia corpus collected by Språk-banken.[10] The OpenNLP sentence splitter and tokenizer are used for normalizing the corpora.

We replace all numbers with a special token NUMBER in all corpora and convert the uppercase letters to corresponding lowercase forms in English and Swedish. Word vectors are extracted only for the unique words appearing at least 100 times. The 10 000 most frequent words are used as context words in the co-occurrence matrix. Table 1 represents some statistics of the corpora.

	#Tokens	$\#W \geq 1$	$\#W \geq 100$	#Sents
English	8×10^9	14 462 600	404 427	4×10^8
Persian	4×10^8	1 926 233	60 718	1×10^7
Swedish	6×10^8	5 437 176	174 538	5×10^7

Table 1: Size of the corpora from which word vectors are extracted; #Tokens: total number of tokens; #W≥k: number of unique words appearing at least k times in the corpora; #Sents: number of sentences.

The word vectors are evaluated with respect to their contribution to the accuracy of parsing models trained with them. The parsing models are trained on the English, Persian, and Swedish parts of the corpus of universal dependencies (Nivre et al., 2016) version 1.2 using Stanford neural dependency parser with gold part-of-speech tags.

The experiments have been performed on Linux 3.10 running on 8 Intel Xeon 2.40 GHz processors and 100 GB main memory. Octave 4.0.0 enabled with 64 bit indexing is used to build the co-occurrence matrix and to extract word vectors from it. The parsing models are trained with 4 threads.

[1] http://www.cs.upc.edu/~nlp/wikicorpus

[2] http://about.reuters.com/researchandstandards/corpus

[3] https://dumps.wikimedia.org/enwiki/

[4] https://www.ldc.upenn.edu

[5] http://opennlp.apache.org

[6] http://nlp.stanford.edu

[7] http://ece.ut.ac.ir/system/files/NLP/Resources/TMC2.rar

[8] http://linguatools.org/tools/corpora/wikipedia-monolingual-corpora

[9] http://stp.lingfil.uu.se/~mojgan/preper.html

[10] https://spraakbanken.gu.se/eng/resources/corpus

6. Results

Table 2 shows the results we obtained from the parsing models trained on the corpus of universal dependencies using Stanford neural dependency parser with RSV word vectors.

	Development set		Test set	
	UAS	LAS	UAS	LAS
English	88.5	85.9	87.6	84.9
Persian	85.7	82.8	85.4	82.4
Swedish	83.4	79.4	86.2	82.5

Table 2: The accuracy of the Stanford neural dependency parser trained with RSV word vectors on the corpus of universal dependencies

We compare our results with the results reported for `MaltParser` (Nivre et al., 2006) and `Parsito` by Straka et al. (2015). `MaltParser` is a greedy transition-based dependency parser which uses a liblinear model as its core classifier. `Parsito` is a transition-based dependency parser inspired by the Stanford neural dependency parser (Chen and Manning, 2014). It added two main items to the original parser: first is a search-based oracle similar to a dynamic oracle and second is the set of morphological features provided by the corpus of universal dependencies. It also replaces the cube activation function used in the Stanford parser with the $tanh$ function.

Table 3 summarizes the results. All the parsing models are trained with the arc-standard system (Nivre, 2004). The `Parsito` results are for both the static oracle (Parsito-St) and the search-based oracle (Parsito-Sr). As shown in all cases the parsing models trained with Stanford neural dependency parser and RSV (Stanford-RSV) are more accurate than the other parsing models. The superiority of the results obtained from Stanford-RSV to the Parsito-Sr shows the importance of word vectors in dependency parsing in comparison with adding more features to the parser or performing the search-based oracle.

	Parsito-St	Parsito-Sr	MaltParser	Stanford-RSV
	UAS LAS	UAS LAS	UAS LAS	UAS LAS
English	86.7 84.2	87.4 84.7	86.3 82.9	87.6 84.9
Persian	83.8 80.2	84.5 81.1	80.8 77.2	85.4 82.4
Swedish	85.3 81.4	85.9 82.3	84.7 80.3	86.2 82.5

Table 3: Accuracy of dependency parsing. Parsito-St and Parsito-Sr refer to the `Parsito` models trained with static oracle and search-based oracle.

7. Conclusion

Distributional representations of words, called *word vectors*, have shown great improvements in a variety of natural language processing tasks. In this paper, we have proposed to use a set of word vectors formed by right singular vectors of a co-occurrence matrix that is transformed by a 7th-root transformation function. Our experiments on the corpus of universal dependencies show that the word vectors are well qualified to model the dependency relations in the corpus. Using the Stanford neural dependency parser with a large enough hidden layer, our word vectors result in some parsing models for English, Persian, and Swedish which are more accurate than other popular transition-based dependency parsers designed with the arc-standard system.

References

Abolfazl AleAhmad, Hadi Amiri, Ehsan Darrudi, Masoud Rahgozar, and Farhad Oroumchian. 2009. Hamshahri: A standard persian text collection. *Knowledge-Based Systems*, 22(5):382–387.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependeny parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th International Joint Conference on Natural Language Processing*, pages 334–343, July.

Rémi Lebret and Ronan Collobert. 2014. Word embeddings through hellinger pca. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–490, Gothenburg, Sweden, April. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics - Special issue on using large corpora*, 19(2):313 – 330, June.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cogni-*

tion Together, pages 50–57. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Magnus Sahlgren. 2006. *The Word-space model*. Ph.D. thesis, Stockholm University.

Mojgan Seraji. 2015. *Morphosyntactic Corpora and Tools for Persian*. Ph.D. thesis, Uppsala University.

Milan Straka, Jan Hajic, Jana Straková, and Jan Hajic jr. 2015. Parsing universal dependency treebanks using neural networks and search-based oracle. In *International Workshop on Treebanks and Linguistic Theories (TLT14)*, pages 208–220.