Universal Dependency Parser: A Single Parser for Many Languages on *Arc-Swift*

Michelle Guo
Department of Computer Science
Stanford University
mguo95@stanford.edu

Frank Fan
Department of Statistics
Stanford University
ffan9@stanford.edu

Abstract

Dependency parsing has been a longstanding task in NLP with a recent boost in performance thanks to neural network models. However, most dependency parsers are monolingual–a single parser is trained per language–and utilize transition-based systems that are limited to local information. We utilize a novel transition-based system, *arc-swift*, proposed in [1] that incorporates both *local* and *global* information and achieves improved performance compared to current transition-based systems. We propose a universal, multilingual dependency parser model built on *arc-swift* and augment this model by aligning GloVe word vectors across languages and prepending the parser with a character-level language detector model to help the parser explicitly learn language embeddings. Our results show that word vector alignments greatly improve UAS and LAS scores, while language detector predictions show modest increase in parse scores. Our universal dependency parser performs comparably to monolingual parser baselines models, and is potentially able to help advance downstream applications, such as machine translation.

1 Introduction

Dependency parsing has recently improved in performance with the help of neural network models [5]. However, many of the dependency parsers today are monoligual, meaning that a single parser neural network is trained and tuned for each language (i.e., [4]). Ammar et al. [2] recently proposed a single universal, multilingual dependency parser to generate parses across multiple languages. In this work, we explore the problem of creating a universal dependency parser to parse sentences in several languages.

However, training the parser to learn multiple languages is on its own a challenging task. In machine translation, [3] showed that inserting a start token representation of the source language as part of the input sentence can help machine translation models learn internal language representations. Drawing inspiration from this, this work implements a language detector model that feeds language predictions to our universal parser model to help it learn multiple languages more explicitly.

Today, many dependency parsers employ transition-based systems that only utilize *local* information about the sentence when predicting transitions [6]. Recently, Qi and Manning [1] introduced a novel transition-based system, *arc-swift*, that takes advantage of both *local* and *global* information to produce parse trees that achieve improved results compared to other transition-based systems.

The contributions of this paper are three-fold: 1) this is the first time that a universal dependency parser model using the *arc-swift* transition system has been built, 2) augment the *arc-swift* parser with language detection predictions and aligned word vector embeddings, and 3) compare different variations of our universal dependency parser with their monolingual counterpart baselines, show-

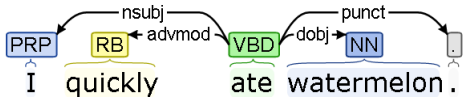

Figure 1: An example dependency parse tree, with arcs indicating the type of dependency between attached words

ing that the universal dependency parser model proposed in this work shows potentially promising results towards downstream applications such as machine translation tasks.

2 Background & Related Work

Dependency parsing is the problem of, given an input sentence, producing a tree structure of the dependencies between words in the sentence. For instance, in the sentence: *"I quickly ate watermelon,"* the word "quickly" would be attached to the word "ate" as a dependent, since "quickly" is an adverbial modification of "ate." Similarly, "I" is also a dependent of "ate" (Figure [1]).

Neural methods in dependency parsing have been explored since Chen and Manning (2014) [5] published the first state-of-the-art neural dependency parser. They use a transition-based system that operates on a buffer and stack of words for efficient parsing. Dozat and Manning (2016) [7] devised a deep biaffine attention model for neural dependency parsing which operates over a graph-based architecture.

2.1 Transition Systems

In transition-based parsing, there are several different schemes a model can use for defining transitions. The traditional scheme, *arc-standard*, is fairly rigid and only allows attachments to be formed between the top two tokens in the parsing stack. Dependency attachments are only formed on a word once all the dependents of a word have been assigned. It is difficult to tell, though, if a token has received all of its dependents, and whether it is an appropriate time to make it a dependent of a word and remove it from the stack.

A second transition system is *arc-eager* (Nivre 2003) [8], which allows defines arc transitions to operate between the stack and the buffer. With this, tokens no longer need to have received all their dependents by the time they are made a dependent.

However, Qi and Manning [1] propose a new transition system called *arc-swift*, which allows for more robust decision-making in forming attachments between words. It incorporates more global information in determining the transition, and is able to form less local attachments.

2.2 Universal Dependency Parsing

Universal Dependency Parsing is the problem of creating a unified model that parses sentences from several different languages. To our knowledge, there is only one other paper that has addressed this problem, by Ammar et al. (2016) [2]. Ammar et al. utilizes multilingual word clusters and embeddings, token-level language information, and language-specific features (fine-grained POS tags). We will similarly explore this problem of creating a universal dependency parser using aligned word embeddings and language detection predictions to inform our parser, but with the use of the *arc-swift* transition system.

3 Approach

In this work, in addition to the *arc-swift* dependency parser, we also augmented the performance of the parser by additionally including word vector alignments as well as language detector predictions as input to the parser model. The whole system architecture is outlined in Figure [2].

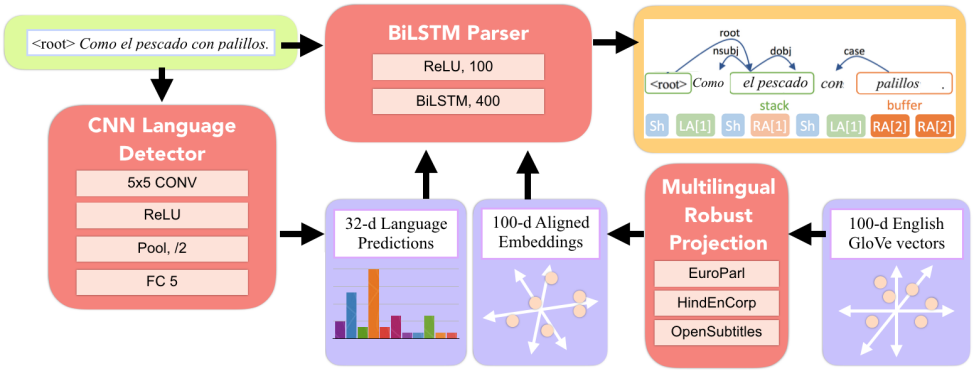

Figure 2: Formalization of the process, from word vector alignment, language detection, and dependency parsing. Dependency parse diagram adapted from [1].

3.1 Multilingual Robust Projection

To build a multilingual dependency parser, there must be some way to unify the vector embeddings of the languages we operate on. The naive approach of using pretrained GloVe vectors from different languages is flawed, since the vector spaces are not guaranteed to be aligned. For instance, the words *gato* in Spanish, and *cat* in English are not guaranteed to map to the same vector embedding, even though they intuitively mean the same thing in their respective languages.

To solve this issue, we use the Multilingual Robust Projection method devised in Guo et al. 2016 [18] to produce a set of aligned embeddings that span several languages. The method only requires specification of a seeder language, which we choose here to be English.

We process the parallel datasets using the Moses tokenizer [17], and feed each pair of parallel corpora (English language and corresponding foreign language) to the Berkeley Aligner [16] to produce alignments between tokens in corresponding sentences. We then produce an alignment matrix, $A_{FOR|ENG} \epsilon \mathbf{R}^{|V_{FOR}| \times |V_{ENG}|}$ where each element of A is given as:

$$A_{FOR|ENG}(i,j) = \frac{\#(V_{FOR}^{(i)} \leftrightarrow V_{ENG}^{(j)})}{\Sigma_k \#(V_{FOR}^{(i)} \leftrightarrow V_{ENG}^{(k)})}$$

The aligned embedding of each word in the foreign language is then the weighted average of its translated words in the parallel corpus:

$$E_{FOR} = A_{FOR|ENG} \cdot E_{ENG}$$

Where E_{FOR} is the matrix of vector embeddings for words in the foreign language, and E_{ENG} is the matrix of English GloVe vector embeddings.

3.2 Language Detector

In order for our dependency parser to be universal, or multilingual, it must be able to take in a source sentence in any language as input. As it stands, the universal parser is already able to implicitly learn the language of a sentence. This is because each language tends to have its own unique tokens, and if not, its own unique syntax in which these tokens are composed together. The BiLSTM model should be able to learn this structure. However, to help our universal model to better parse sentences from multiple languages, we wish to provide an explicit, consistent signal for the identity of the source language.

We implemented a character-level CNN model for our language detector. We used one set of CONV-RELU-POOL layers, using a filter size of 5 and pooling size of 2, followed by a fully-connected layer with 5 hidden units. We set a maximum sentence length of 140 characters, truncating and padding

arc-swift

Shift	$(\sigma, i\|\beta, A) \Rightarrow (\sigma\|i, \beta, A)$
LArc$[n]$	$(\sigma\|i_n\| \ldots \|i_1, j\|\beta, A)$
	$\Rightarrow (\sigma, j\|\beta, A \cup \{(j \rightarrow i_n)\})$
RArc$[n]$	$(\sigma\|i_n\| \ldots \|i_1, j\|\beta, A)$
	$\Rightarrow (\sigma\|i_n\|j, \beta, A \cup \{(i_n \rightarrow j)\})$

Figure 3: The *arc-swift* transition system, from Qi and Manning [1].

sentences as needed. We also represent unseen characters from datasets other than the training set with the <UNK> character.

3.3 Arc-*Swift* Transition System

Arc-swift, proposed by Qi and Manning (2017) [1], is a novel transition system for dependency parsing described in Figure [3]. Qi and Manning (2017) show that *arc-swift* has advantages over current transition based systems (i.e. *arc-standard* [9]). First of all, *arc-swift* utilizes both *global* and *local* information when generating parses trees. This is in a sense incorporating elements of both transition-based, which is local, and graph-based, which is global, systems into a single transition system. Furthermore, [1] show that the *arc-swift* parser achieves better accuracy on prepositional phrase and conjunction attachments, and is comparable to other parsers (i.e. *arc-standard* [9], *arc-eager* [8], *arc-hybrid* [10]) on other common dependency parser errors.

As a simple example of where *arc-swift* shines, consider the following sentences:

I eat steak with mustard.
I eat steak with chopsticks.

It is clear that in the first sentence, *mustard* is modifying and complementing the direct object *steak*, while in the second sentence, *chopsticks* modifies the main verb *ate*. In parsing these two sentences, the *Arc-swift* transition system is able to simultaneously compare different attachment points for the words, by allowing arcs to be formed with words deeper in the parser stack. This exploration is not possible with the traditional transition system, *arc-standard*, which only allows for arcs to be formed between the top two tokens in the stack. This forces the parser to make transition decisions early on that don't fully utilize the global context, such that errors can easily propagate through the entire parse.

3.4 Dependency Parser Network Architecture

The dependency parser is created using a bidirectional long short-term memory (BiLSTM) network. The BiLSTM has 400 hidden units in each direction, and the output is fed into a 100-dimensional dense layer of rectified linear units (ReLU). Two sets of dense layers are used to create representations for "head" and "dependent" for each token, which are then merged according to the parser state for predicting transitions.

For *arc-swift*, we featurize each arc-inducing transition using the following bi-affine combination:

$$\begin{aligned} v_{feat,i} &= [f(v_{head}, v_{dep})]_i \\ &= \text{ReLU}(v_{head}^\top W_i v_{dep} + v_i^\top v_{head} + c_i^\top v_{dep} + d_i) \end{aligned} \tag{1}$$

where $W_i \in \mathbb{R}^{32x32}$, $b_i, c_i \in \mathbb{R}^{32}$, d_i is a scalar for $i = 1, ..., 32$, and v_{head} and v_{dep} are the head and token of the arc, respectively. For each parsing step, we generate L scores from the equation where L is the set of all the possible arc labels. To obtain scores for Shift, we use Equation (1) to combine v_{head} and v_{dep} of the leftmost token in the buffer. Softmax is used to obtain a probability distribution over all possible transitions at a given time step. The parser is trained to maximize the log likelihood of the parse.

4 Experiments

4.1 Datasets

Table 1: Datasets

Language	UD Size	Parallel Corpus	Parallel Corpus Size
BG	128956	OpenSubtitles	250000
CS	1035841	EuroParl	646605
EN	198438	—	—
ES	364217	EuroParl	1965734
ET	170293	EuroParl	651746
EU	44657	OpenSubtitles	173384
FA	114951	OpenSubtitles	3715719
FI	155968	EuroParl	1924942
HI	241965	HindEnCorp	273885
IT	242958	EuroParl	1909115
NO	230859	OpenSubtitles	5584616

This work investigates eleven languages: Bulgarian (BG), Czech (CS), English (EN), Spanish (ES), Estonian (ET), Basque (EU), Farsi/Persian (FA), Finnish (FI), Hindi (HI), Italian (IT), Norwegian (NO). The languages were chosen as part of the CoNLL shared task [19], which consists of 30 languages.

We use the Universal Dependencies (UD) [11] dataset for training and evaluating our dependency parser. We used parallel corpora from OpenSubtitles [12], EuroParl [13], and HindEnCorp [14] to align 100-dimensional GloVe word vectors across the target languages. Table [1] summarizes the datasets we used for dependency parses and parallel corpora.

4.2 Language Detection

We trained three versions of the character-level CNN language detector model, using learning rates of $\alpha = 0.1, 0.01, 0.001$. We found that the best learning rate is 0.01 for the language detector, and achieved 97.8% accuracy on the test UD dataset.

4.3 *Arc-Swift* Dependency Parser

For all of the experiments training the *arc-swift* dependency parser, we utilize the `ADAM` optimizer [15] with $\beta_2 = 0.9$, an initial learning rate of 0.001, and use minibatch size of 32 sentences. We train each of our parser models for 30 epochs each through the UD dataset. We also annealed the learning rate by a factor of 0.5 for every 3 epochs after the 15th epoch, which had been found to improve performance.

4.4 Universal Dependency Parser

We ran three different experiments on our universal dependency parser, and the UAS and LAS scores for each of these models are shown in Figure [4]. As a baseline, we trained one multilingual dependency parser without word vector alignments or language detection. To understand the effects of having language ID as part of the parser, we ran another experiment where the language detector fed language predictions to the multilingual parser, and the parser was trained on language embeddings. Finally we trained the complete system, where we used both language detection and robust projection for inputs to the multilingual parser.

We also tuned the model by doing a hyperparameter search. The training loss curve for dropout rates 0.05 and 0.5 are shown in Figure [5]. We found that the optimal dropout rate for this network model is 0.05.

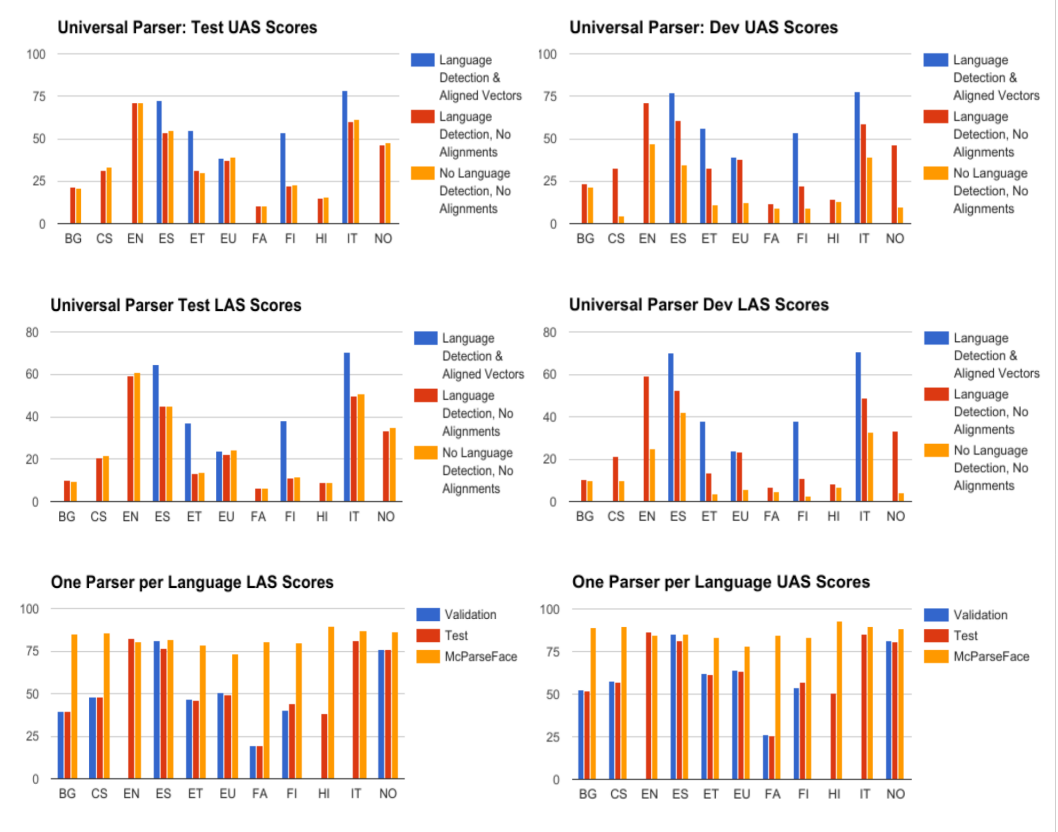

Figure 4: Rows 1 and 2: Universal Parser results for three different models: 1) Using language detection and word vector alignments (blue); 2) Using language detection and no word vector alignments (red); 3) No language detection and no word vector alignments (yellow). Row 3: Eleven separate monolingual parser results, trained one parser model for each of the eleven languages (blue and red), compared to Google's Parser models [4].

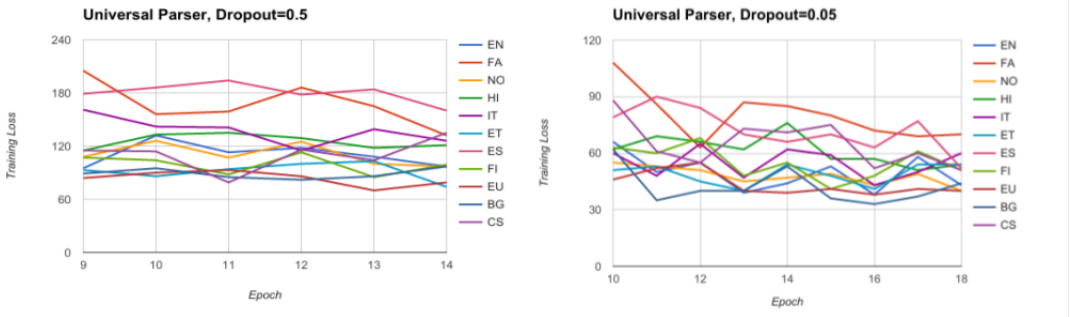

Figure 5: Hyperparameter tuning on our universal parser with language detection and word vector alignments for dropout rates of 0.5 and 0.05.

4.5 Monolingual Dependency Parser

We additionally trained eleven baseline neural network models, one model for each target language. These models employed a BiLSTM parser with an *arc-swift* transition system. We report the UAS and LAS validation and test scores for each model in Figure [4], and compare these results with Google's Parser [4] results.

5 Conclusion

From inspection of the graphs, it seems that during training and validation, the language detection and foreign vector alignments provide an appreciable boost in performance. However, we see that in the test evaluation, the benefit from language detection is negligible and only the multilingual projections provide improvement in the model. This may be because in the multilingual models, the language of a sentence is already implicitly learned. Each language tends to have its own unique tokens, and if not, its own unique syntax in which the tokens are composed together. So the neural network weights learn this structure, while the word embeddings are continuously updated to effectively form a set of multilingual aligned vectors. The initial hope, though, was for the language detector to provide a more consistent signal to the parser about the language of the sentence.

Furthermore, the model is underfitting the data. The results from the monolingual models are uniformly superior to the multilingual model results. The plots of loss versus epoch show that the .5 dropout model exhibits barely any improvement, while the .05 model displays a slightly better downward trend in loss.

Compared to our baselines, the universal dependency parser model does not do significantly worse than the counterpart, which is a single neural network trained per language. We hope that this work can lead to interesting insights and improvements in downstream applications, such as machine translation.

5.1 Future Work

In the future, we would like to combine the language detection model with the dependency parser model in a jointly trained end-to-end system. We are also interested in constructing our universal dependency parser model using various other transition systems such as arc-standard, arc-eager, and arc-hybrid. Finally, we would like to utilize the insights from this work to create a universal model for machine translation.

Acknowledgments

We would like to thank Peng Qi, for the amazing support and guidance he has given us throughout this project.

We would also like to acknowledge the teaching staff for CS 224N, for all the support and help they have provided us throughout the quarter, and for this amazing opportunity to work on interesting work in NLP.

References

[1] Qi, P. and Manning, C. (2017). *Arc-swift*: A Novel Transition System for Dependency Parsing.

[2] Ammar, W., Mulcaire, G., Ballesteros, M., Dyer, C., Smith, N. A. (2016). Many languages, one parser. arXiv preprint arXiv:1602.01595.

[3] Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viegas, F., Wattenberg, M., Corrado, G., Hughes, M., Dean, J. (2016). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. arXiv preprint arXiv:1611.04558.

[4] Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., Collins, M. (2016). Globally normalized transition-based neural networks. arXiv preprint arXiv:1603.06042.

[5] Chen, D., Manning, C. D. (2014, October). A Fast and Accurate Dependency Parser using Neural Networks. In EMNLP (pp. 740-750).

[6] McDonald, R. T. and Nivre, J. (2007). Characterizing the errors of data-driven dependency parsing models. In EMNLP-CoNLL. pages 122131. http://www.aclweb.org/anthology/D07-1013.

[7] Dozat, Timothy, and Christopher D. Manning. "Deep biaffine attention for neural dependency parsing." arXiv preprint arXiv:1611.01734 (2016).

[8] Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In Proceedings of the 8th International Workshop on Parsing Technologies (IWPT. pages 149160.)

[9] Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In Proceedings of the Workshop on Incremental Parsing: Bring- ing Engineering and Cognition Together. Associ- ation for Computational Linguistics, pages 5057. https://www.aclweb.org/anthology/W04-0308.

[10] Marco Kuhlmann, Carlos Go mez-Rodr guez, and Gior- gio Satta. 2011. Dynamic programming alg rithms for transition-based dependency parsers. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, pages 673682. http://www.aclweb.org/anthology/P11-1068.

[11] 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). pages 16591666.

[12] Tiedemann, J.: News from opus - a collection of multilingual parallel corpora with tools and interfaces. In: Nicolov, N., Bontcheva, K., Angelova, G., Mitkov, R. (eds.) Recent Advances in Natural Language Processing, vol. V, pp. 237248. John Benjamins, Amsterdam (2009)

[13] Koehn, P. (2005, September). Europarl: A parallel corpus for statistical machine translation. In MT summit (Vol. 5, pp. 79-86).

[14] Bojar, O., Diatka, V., Rychl, P., Strank, P., Suchomel, V., Tamchyna, A., Zeman, D. (2014, May). HindEnCorp-Hindi-English and Hindi-only Corpus for Machine Translation. In LREC (pp. 3550-3555).

[15] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[16] Liang, Percy, Ben Taskar, and Dan Klein. Alignment by agreement. In Proc. of HLT-NAACL06, pages 104111, New York City, USA, 2006.

[17] Koehn, Philipp, et al. "Moses: Open source toolkit for statistical machine translation." Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions. Association for Computational Linguistics, 2007.

[18] Guo, Jiang, et al. "A Representation Learning Framework for Multi-Source Transfer Parsing." AAAI. 2016.

[19] http://universaldependencies.org/conll17/evaluation.html

[20] DeNero, J., Liang, P. (2007). The Berkeley Aligner.