Cross-Lingual Dependency Parsing with Late Decoding for Truly Low-Resource Languages

Michael Sejr Schlichtkrull
University of Amsterdam*
m.s.schlichtkrull@uva.nl

Anders Søgaard
University of Copenhagen
soegaard@di.ku.dk

Abstract

In cross-lingual dependency annotation projection, information is often lost during transfer because of early decoding. We present an end-to-end graph-based neural network dependency parser that can be trained to reproduce matrices of edge scores, which can be directly projected across word alignments. We show that our approach to cross-lingual dependency parsing is not only simpler, but also achieves an absolute improvement of 2.25% averaged across 10 languages compared to the previous state of the art.

1 Introduction

Dependency parsing is an integral part of many natural language processing systems. However, most research into dependency parsing has focused on learning from treebanks, i.e. collections of manually annotated, well-formed syntactic trees. In this paper, we develop and evaluate a graph-based parser which does not require the training data to be well-formed trees. We show that such a parser has an important application in cross-lingual learning.

Annotation projection is a method for developing parsers for low-resource languages, relying on aligned translations from resource-rich source languages into the target language, rather than linguistic resources such as treebanks or dictionaries. The Bible has been translated completely into 542 languages, and partially translated into a further 2344 languages. As such, the assumption that we have access to parallel Bible data is much less constraining than the assumption of access to linguistic resources. Furthermore, for truly low-resource languages, relying upon the Bible scales better than relying on less biased data such as the EuroParl corpus.

In Agić et al. (2016), a projection scheme is proposed wherein labels are collected from many sources, projected into a target language, and then averaged. Crucially, the paper demonstrates how projecting and averaging edge scores from a graph-based parser *before* decoding improves performance. Even so, decoding is still a requirement between projecting labels and retraining from the projected data, since their parser (TurboParser) requires well-formed input trees. This introduces a potential source of noise and loss of information that may be important for finding the best target sentence parse.

Our approach circumvents the need for decoding prior to training, thereby surpassing a state-of-the-art dependency parser trained on decoded multi-source annotation projections as done by Agić et al. We first evaluate the model across several languages, demonstrating results comparable to the state of the art on the Universal Dependencies (McDonald et al., 2013) dataset. Then, we evaluate the same model by inducing labels from cross-lingual multi-source annotation projection, comparing the performance of a model with early decoding to a model with late decoding.

Contributions We present a novel end-to-end neural graph-based dependency parser and apply it in a cross-lingual setting where the task is to induce models for truly low-resource languages, assuming only parallel Bible text. Our parser is more flexible than similar parsers, and accepts any weighted or non-weighted graph over a token sequence as input. In our setting, the input is a dense weighted graph, and we show that our parser is superior to previous best approaches to cross-lingual parsing. The code is made available on GitHub.[1]

*Work done while at the University of Copenhagen.

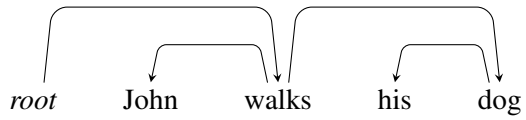
[1] https://github.com/MichSchli/Tensor-LSTM

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 220–229,
Valencia, Spain, April 3-7, 2017.

2 Model

The goal of this section is to construct a first-order graph-based dependency parser capable of learning *directly* from potentially incomplete matrices of edge scores produced by another first-order graph-based parser. Our approach is to treat the encoding stage of the parser as a tensor transformation problem, wherein tensors of edge features are mapped to matrices of edge scores. This allows our model to approximate sets of scoring matrices generated by another parser directly through non-linear regression. The core component of the model is a layered sequence of recurrent neural network transformations applied to the axes of an input tensor.

More formally, any digraph $G = (V, E)$ can be expressed as a binary $|V| \times |V|$-matrix M, where $M_{ij} = 1$ if and only if $(j, i) \in E$ – that is, if i has an ingoing edge from j. If G is a tree rooted at v_0, v_0 has no ingoing edges. Hence, it suffices to use a $(|V|-1)\times|V|$-matrix. In dependency parsing, every sentence is expressed as a matrix $S \in R^{w\times f}$, where w is the number of words in the sentence and f is the width of a feature vector corresponding to each word. The goal is to learn a function $P : \mathbb{R}^{w\times f} \rightarrow \mathbb{Z}_2^{w\times(w+1)}$, such that $P(S)$ corresponds to the matrix representation of the correct parse tree for that sentence – see Figure 1 for an example.



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

Figure 1: An example dependency tree and the corresponding parse matrix.

In the arc-factored (first-order), graph-based model, P is a composite function $P = D \circ E$ where the encoder $E : \mathbb{R}^{w\times f} \rightarrow \mathbb{R}^{w\times(w+1)}$ is a real-valued scoring function and the decoder $D : \mathbb{R}^{w\times(w+1)} \rightarrow \mathbb{Z}_2^{w\times(w+1)}$ is a minimum spanning tree algorithm (McDonald et al., 2005). Commonly, the encoder includes only *local* information – that is, E_{ij} is only dependent on S_i and S_j, where S_i and S_j are feature vectors corresponding to dependent and head. Our contribution is the introduction of an LSTM-based *global* encoder where the entirety of S is represented in the calculation of E_{ij}.

We begin by extending S to a $(w+1)\times(f+1)$-matrix S^* with an additional row corresponding to the root node and a single binary feature denoting whether a node is the root. We now compute a 3-tensor $F = S \boxplus S^*$ of dimension $w \times (w + 1) \times (2f + 1)$ consisting of concatenations of all combinations of rows in S and S^*. This tensor effectively contains a featurization of every edge (u, v) in the complete digraph over the sentence, consisting of the features of the parent word u and child word v. These edge-wise feature vectors are organized in the tensor exactly as the dependency arcs in a parse matrix such as the one shown in the example in Figure 1.

The edges represented by elements F_{ij} can as such easily be interpreted in the context of related edges represented by the row i and the column j in which that edge occurs. The classical arc-factored parsing algorithm of McDonald et al. (2005) corresponds to applying a function $O : \mathbb{R}^{2f+1} \rightarrow \mathbb{R}$ pointwise to $S \boxplus S^*$, then decoding the resulting $w \times (w + 1)$-matrix. Our model diverges by applying an LSTM-based transformation $Q : \mathbb{R}^{w\times(w+1)\times(2f+1)} \rightarrow \mathbb{R}^{w\times(w+1)\times d}$ to $S \boxplus S^*$ before applying an analogous transformation $O : \mathbb{R}_d \rightarrow \mathbb{R}$.

The Long Short-Term Memory (LSTM) unit is a function $LSTM(x, h_{t-1}, c_{t-1}) = (h_t, c_t)$ defined through the use of several intermediary steps, following Hochreiter et al. (2001). A concatenated input vector $I = x \oplus h_{prev}$ is constructed, where $\oplus$ represents vector concatenation. Then, functions corresponding to input, forget, and output gates are defined following the form $g_{input} = \sigma(W_{input}I + b_{input})$. Finally, the internal cell state c_t and the output vector h_t at time t are defined using the Hadamard (pointwise) product $\bullet$:

$$c_t = g_{forget} \bullet c_{prev} + g_{input} \bullet tanh(W_{cell}I + b_{cell})$$
$$h_t = g_{output} \bullet tanh(c_t)$$

We define a function Matrix-LSTM inductively, that applies an LSTM to the rows of a matrix X. Formally, Matrix-LSTM is a function $\mathcal{M} : \mathbb{R}^{a\times b} \rightarrow \mathbb{R}^{a\times c}$ such that $(h_1, c_1) = LSTM(X_1, 0, 0)$, $\forall 1 < i \leq n$ $(h_i, c_i) = LSTM(X_i, h_{i-1}, c_{i-1})$, and $\mathcal{M}(X)_i = h_i$.

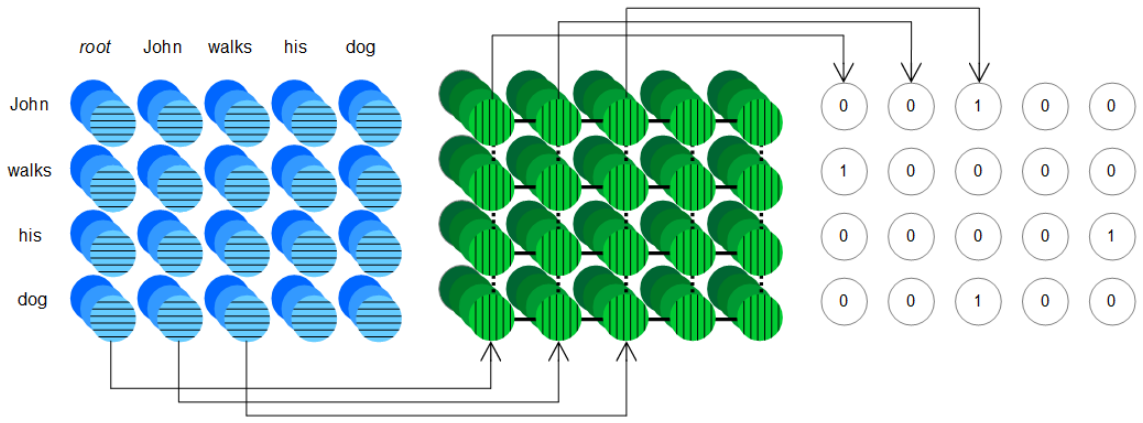


Figure 2: Four-directional Tensor-LSTM applied to the example sentence seen in Figure 1. The word-pair tensor $S \boxplus S^*$ is represented with blue units (horizontal lines), a hidden Tensor-LSTM layer H with green units (vertical lines), and the output layer with white units. The recurrent connections in the hidden layer along H and $H^{T(2,1,3)}$ are illustrated respectively with dotted and fully drawn lines.

An effective extension is the *bidirectional* LSTM, wherein the LSTM-function is applied to the sequence both in the forward and in the backward direction, and the results are concatenated. In the matrix formulation, reversing a sequence corresponds to inverting the order of the rows. This is most naturally accomplished through left-multiplication with an exchange matrix $J_m \in \mathbb{R}^{m \times m}$ such that:

$$J_m = \begin{bmatrix} 0 & \cdots & 1 \\ \vdots & \iddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix}$$

Bidirectional Matrix-LSTM is therefore defined as a function $\mathcal{M}_{2d} : \mathbb{R}^{a \times b} \rightarrow \mathbb{R}^{a \times 2c}$ such that:

$$\mathcal{M}_{2d}(S) = \mathcal{M}(S) \oplus_2 J_a \mathcal{M}(J_a S)$$

Here, $\oplus_2$ refers to concatenation along the second axis of the matrix.

Keeping in mind the goal of constructing a tensor transformation Q capable of propagating information in an LSTM-like manner between any two elements of the input tensor, we are interested in constructing an equivalent of the Matrix-LSTM-model operating on 3-tensors rather than matrices. This construct, when applied to the edge tensor $F = S \boxplus S^*$, can then provide a means of interpreting edges in the context of related edges.

A very simple variant of such an LSTM-function operating on 3-tensors can be constructed by applying a bidirectional Matrix-LSTM to every matrix along the first axis of the tensor. This forms the center of our approach. Formally, bidirectional Tensor-LSTM is a function $\mathcal{T}_{2d} : \mathbb{R}^{a \times b \times c} \rightarrow \mathbb{R}^{a \times b \times 2h}$ such that:

$$\mathcal{T}_{2d}(T)_i = \mathcal{M}_{2d}(T_i)$$

This definition allows information to flow *within* the matrices of the first axis of the tensor, but not *between* them – corresponding in Figure 2 to horizontal connection along the rows, but no vertical connections along the columns. To fully cover the tensor structure, we must extend this model to include connections along columns.

This is accomplished through tensor transposition. Formally, tensor transposition is an operator $T^{T\sigma}$ where σ is a permutation on the set $\{1, ..., rank(T)\}$. The last axis of the tensor contains the feature representations, which we are not interested in scrambling. For the Matrix-LSTM, this leaves only one option – $M^{T(1,2)}$. When the LSTM is operating on a 3-tensor, we have two options – $T^{T(2,1,3)}$ and $T^{T(1,2,3)}$. This leads to the following definition of four-directional Tensor-LSTM as a function $\mathcal{T}_{4d} : \mathbb{R}^{a \times b \times c} \rightarrow \mathbb{R}^{a \times b \times 4h}$ analogous to bidirectional Sequence-LSTM:

$$\mathcal{T}_{4d}(T) = \mathcal{T}_{2d}(T) \oplus_3 \mathcal{T}_{2d}(T^{T(2,1,3)})^{T(2,1,3)}$$

Calculating the LSTM-function on $T^{T(1,2,3)}$ and $T^{T(2,1,3)}$ can be thought of as constructing the recurrent links either "side-wards" or "down-wards" in the tensor – or, equivalently, constructing recurrent links either between the outgoing or between the in-going edges of every vertex in

the dependency graph. In Figure 2, we illustrate the two directions respectively with full or dotted edges in the hidden layer.

The output of Tensor-LSTM is itself a tensor. In our experiments, we use a multi-layered variation implemented by stacking layers of models: $\mathcal{T}_{4d,stack}(T) = \mathcal{T}_{4d}(\mathcal{T}_{4d}(...\mathcal{T}_{4d}(T)...))$. We do not share parameters between stacked layers. Training the model is done by minimizing the value $\mathcal{E}(G, O(Q(S \boxplus S^*)))$ of some loss function $\mathcal{E}$ for each sentence S with gold tensor G. We experiment with two loss functions.

In our monolingual set-up, we exploit the fact that parse matrices by virtue of depicting trees are right stochastic matrices. Following this observation, we constrain each row of $O(Q(S \boxplus S^*))$ under a softmax-function and use as loss the row-wise cross entropy. In our cross-lingual set-up, we use mean squared error. In both cases, prediction-time decoding is done with Chu-Liu-Edmonds algorithm (Edmonds, 1968) following McDonald et al. (2005).

3 Cross-lingual parsing

Hwa et al. (2005) is a seminal paper for cross-lingual dependency parsing, but they use very detailed heuristics to ensure that the projected syntactic structures are well-formed. Agić et al. (2016) is the latest continuation of their work, presenting a new approach to cross-lingual projection, projecting edge scores rather than subtrees. Agić et al. (2016) construct target-language treebanks by aggregating scores from multiple source languages, before decoding. Averaging before decoding is especially beneficial when the parallel data is of low quality, as the decoder introduces errors, when edge scores are missing. Despite averaging, there will still be scores missing from the input weight matrices, especially when the source and target languages are very distant. Below, we show that we can circumvent error-inducing early decoding by training directly on the projected edge scores.

We assume source language datasets $\mathcal{L}_1, ..., \mathcal{L}_n$, parsed by monolingual arc-factored parsers. In our case, this data comes from the Bible. We assume access to a set of sentence alignment functions $A_s : \mathcal{L}_s \times \mathcal{L}_t \rightarrow \mathbb{R}_{0,1}$ where $A_s(S_s, S_t)$ is the confidence that S_t is the translation of S_s. Similarly, we have access to a set of word alignment functions $W_{\mathcal{L}_s,S_s,S_t} : S_s \times S_t \rightarrow \mathbb{R}_{0,1}$ such that $S_s \in \mathcal{L}_s$, $S_t \in \mathcal{L}_t$, and $W(w_s, w_t)$ represents the confidence that w_s aligns to w_t given that S_t is the translation of S_s

For each source language $\mathcal{L}_s$ with a scoring function $score_{\mathcal{L}_s}$, we define a local edge-wise voting function $vote_{S_s}((u_s, v_s), (u_t, v_t))$ operating on a source language edge $(u_s, v_s) \in S_s$ and a target language edge $(u_t, u_t) \in S_t$. Intuitively, every source language edge votes for every target language edge with a score proportional to the confidence of the edges aligning and the score given in the source language. For every target language edge $(u_t, v_t) \in S_t$:

$$\begin{aligned} vote_{S_s}((u_s, v_s), (u_t, v_t)) = W_{\mathcal{L}_s,S_s,S_t}(u_s, u_t) \\ \cdot W_{\mathcal{L}_s,S_s,S_t}(v_s, v_t) \\ \cdot score_{\mathcal{L}_s}(u_s, v_s) \end{aligned}$$

Following Agić et al. (2016), a sentence-wise voting function is then constructed as the highest contribution from a source-language edge:

$$vote_{S_s}(u_t, v_t) = \max_{u_s,v_s \in S_s} vote_{S_s}((u_s, v_s), (u_t, v_t))$$

The final contribution of each source language dataset $\mathcal{L}_s$ to a target language edge (u_t, v_t) is then calculated as the sum for all sentences $S_s \in \mathcal{L}_s$ over $vote_{S_s}(u_t, v_t)$ multiplied by the confidence that the source language sentence aligns with the target language sentence. For an edge (u_t, v_t) in a target language sentence $S_t \in \mathcal{L}_t$:

$$vote_{\mathcal{L}_s}(u_t, v_t) = \sum_{S_s \in \mathcal{L}_s} A_s(S_s, S_t)\ vote_{S_s}(u_t, v_t)$$

Finally, we can compute a target language scoring function by summing over the votes for every source language:

$$score(u_t, v_t) = \frac{\sum_{i=1}^{n} vote_{\mathcal{L}_i}(u_t, v_t)}{Z_{S_t}}$$

Here, Z_{S_t} is a normalization constant ensuring that the target-language scores are proportional to those created by the source-language scoring functions. As such, Z_{S_t} should consist of the sum over the weights for each sentence contributing to the scoring function. We can compute this as:

$$Z_{S_t} = \sum_{i=1}^{n} \sum_{S_s \in L_i} A_s(S_s, S_t)$$

The sentence alignment function is not a probability distribution; it may be the case that no source-language sentences contribute to a target language sentence, causing the sum of the weights *and* the sum of the votes to approach zero. In this case, we define $score(u_t, v_t) = 0$. Before projection, the source language scores are all standardized to have 0 as the mean and 1 as the standard deviation. Hence, this corresponds to assuming neither positive nor negative evidence concerning the edge.

We experiment with two methods of learning from the projected data – decoding with Chu-Liu-Edmonds algorithm and then training as proposed in Agić et al. (2016), or directly learning to reproduce the matrices of edge scores. For alignment, we use the sentence-level *hunalign* algorithm introduced in Varga et al. (2005) and the token-level model presented in Östling (2015).

4 Experiments

We conduct two sets of experiments. First, we evaluate the Tensor-LSTM-parser in the monolingual setting. We compare Tensor-LSTM to the TurboParser (Martins et al., 2010) on several languages from the Universal Dependencies dataset. In the second experiment, we evaluate Tensor-LSTM in the cross-lingual setting. We include as baselines the delexicalized parser of McDonald et al. (2011), and the approach of Agić et al. (2016) using TurboParser. To demonstrate the effectiveness of circumventing the decoding step, we conduct the cross-lingual evaluation of Tensor-LSTM using cross entropy loss with *early* decoding, and using mean squared loss with *late* decoding.

4.1 Model selection and training

Our features consist of 500-dimensional word embeddings trained on translations of the Bible. The word embeddings were trained using skipgram with negative sampling on a word-by-sentence PMI matrix induced from the Edinburgh Bible Corpus, following (Levy et al., 2017). Our embeddings are not trainable, but fixed representations throughout the learning process. Unknown tokens were represented by zero-vectors.

We combined the word embeddings with one-hot-encodings of POS-tags, projected across word alignments following the method of Agić et al. (2016). To verify the value of the POS-features, we conducted preliminary experiments on English development data. When including POS-tags, we found small, non-significant improvements for monolingual parsing, but significant improvements for cross-lingual parsing.

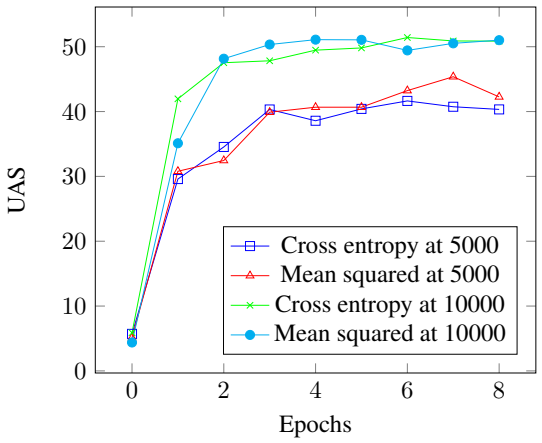


Figure 3: UAS per epoch on German development data training from 5000 or 10000 randomly sampled sentences with projected annotations.

The weights were initialized using the normalized values suggested in Glorot and Bengio (2010). Following Jozefowicz et al. (2015), we add 1 to the initial forget gate bias. We trained the network using RMSprop (Tieleman and Hinton, 2012) with hyperparameters $\alpha = 0.1$ and $\gamma = 0.9$, using minibatches of 64 sentences. Following Neelakantan et al. (2015), we added a noise factor $n \sim \mathcal{N}(0, \frac{1}{(1+t)^{0.55}})$ to the gradient in each update. We applied dropouts after each LSTM-layer with a dropout probability $p = 0.5$, and between the input layer and the first LSTM-layer with a dropout probability of $p = 0.2$ (Bluche et al., 2015). As proposed in Pascanu et al. (2012), we employed a gradient clipping factor of 15. In the monolingual setting, we used early stopping on the development set.

We experimented with 10, 50, 100, and 200 hidden units per layer, and with up to 6 layers. Using greedy search on monolingual parsing and evaluating on the English development data, we determined the optimal network shape to contain 100 units per direction per hidden layer, and a total of 4 layers.

For the cross-lingual setting, we used two additional hyper-parameters. We used the development data from one of our target languages (German) to determine the optimal number of epochs before stopping. Furthermore, we trained only on a subset of the projected sentences, choosing the size of the subset using the development data.

We experimented with either 5000 or 10000 randomly sampled sentences. There are two motivating factors behind this subsampling. First, while the Bible in general consists of about 30000 sentences, for many low-resource languages we do not have access to annotation projections for the full Bible, because parts were never translated, and because of varying projection quality. Second, subsampling speeds up the training, which was necessary to make our experiments practical: At 10000 sentences and on a single GPU, each epoch takes approximately 2.5 hours. As such, training for a single language could be completed in less than a day. We plot the results in Figure 3. We see that the best performance is achieved at 10000 sentences, and with respectively 6 and 5 epochs for cross entropy and mean squared loss.

4.2 Results

In the monolingual setting, we compare our parser to TurboParser (Martins et al., 2010) – a fast, capable graph-based parser used as a component in many larger systems. TurboParser is also the system of choice for the cross-lingual pipeline of Agić et al. (2016). It is therefore interesting to make a direct comparison between the two. The results can be seen in Table 1.

Language	TurboParser	Tensor-LSTM
English*	83.84	**85.81**
German	81.45	**82.64**
Danish	81.82	**82.24**
Finnish	77.74	**78.83**
Spanish	83.19	**86.69**
French	81.17	**84.63**
Czech	81.32	**85.04**
Average	81.50	**83.70**

Table 1: Unlabeled Attachment Score on the UD test data for TurboParser and Tensor-LSTM with cross entropy loss. English development data was used for model selection (marked *).

Note that in order for a parser to be directly applicable to the annotation projection setup explored in the secondary experiment, it must be a *first-order graph-based* parser. In the monolingual setting, the best results reported so far (84.74, on average) for the above selection of treebanks were by the Parsito system (Straka et al., 2015), a transition-based parser using a dynamic oracle.

For the cross-lingual annotation projection experiments, we use the delexicalized system suggested by McDonald et al. (2011) as a baseline. We also compare against the annotation projection scheme using TurboParser suggested in Agić et al. (2016), representing the previous state of the art for truly low-resource cross-lingual dependency parsing. Note that while our results for the TurboParser-based system use the same training data, test data, and model as in Agić et al., our results differ due to the use of the Bible corpus rather than a Watchtower publications corpus as parallel data. The authors made results available using the Edinburgh Bible Corpus for unlabeled data. The two tested conditions of Tensor-LSTM are the mean squared loss model *without* intermediary decoding, and the cross entropy model *with* intermediary decoding. The results of the cross-lingual experiment can be seen in Table 2.

5 Discussion

As is evident from Table 2, the variation in performance across different languages is large for all systems. This is to be expected, as the quality of the projected label sets vary widely due to linguistic differences. On average, Tensor-LSTM with mean squared loss outperforms all other systems. In Section 1, we hypothesized that incomplete projected scorings would have a larger impact upon systems reliant on an intermediary decoding step. To investigate this claim, we plot in Figure 4 the performance difference with mean squared loss and cross entropy loss for each language versus the percentage of missing edge scores.

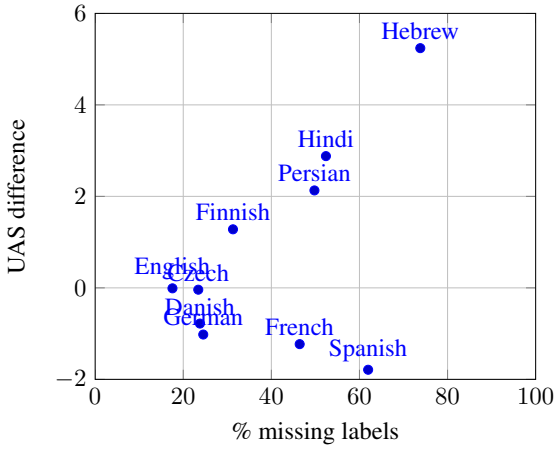


Figure 4: Percentage of missing edge scores versus performance difference for Tensor-LSTM with mean squared loss and cross entropy loss.

Language	Delexicalized	TurboParser	Tensor-LSTM (Decoding)	Tensor-LSTM (No decoding)
Czech (cs)	40.99	**43.81**	42.58	41.54
Danish (da)	49.65	54.87	**54.93**	54.15
English* (en)	48.08	52.52	**52.91**	52.90
Finnish (fi)	41.18	**46.08**	43.98	45.26
French (fr)	48.97	45.83	**55.06**	53.83
German* (de)	49.36	51.79	**54.87**	53.85
Spanish (es)	47.60	58.90	**59.60**	57.81
Persian (fa)	28.93	14.88	46.47	**48.60**
Hebrew (he)	19.06	**52.89**	26.17	31.41
Hindi (hi)	21.03	43.31	43.21	**46.09**
Average	39.49	46.29	47.98	**48.54**

Table 2: Unlabeled attachment scores for the various systems. Tensor-LSTM is evaluated using cross entropy and mean squared loss. We include the results of two baselines – the delexicalized system of McDonald et al. (2011) and the Turbo-based projection scheme of Agić et al. (2016). English and German development data was used for hyperparameter tuning (marked *).

For languages outside the Germanic and Latin families, our claim holds – the performance of the cross entropy loss system decreases faster with the percentage of missing labels than the performance of the mean squared loss system. To an extent, this confirms our hypothesis, as we for the average language observe an improvement by circumventing the decoding step. French and Spanish, however, do not follow the same trend, with cross entropy loss outperforming mean squared loss despite the high number of missing labels.

In Table 2, performance on French and Spanish for both systems can be seen to be very high. It may be the case that Indo-European target languages are not as affected by missing labels as most of the *source* languages are themselves Indo-European. Another explanation could be that some feature of the cross entropy loss function makes it especially well suited for Latin languages – as seen in Table 1, French and Spanish are also two of the languages for which Tensor-LSTM yields the highest performance improvement.

To compare the effect of missing edge scores upon performance without influence from linguistic factors such as language similarity, we repeat the cross-lingual experiment on one language with respectively 10%, 20%, 30%, and 40% of the projected and averaged edge scores artificially set to 0, simulating missing data. We choose the English data for this experiment, as the English projected data has the lowest percentage of missing labels across any of the languages. In Figure 5, we plot the performance for each of the two systems versus the percentage of deleted values.

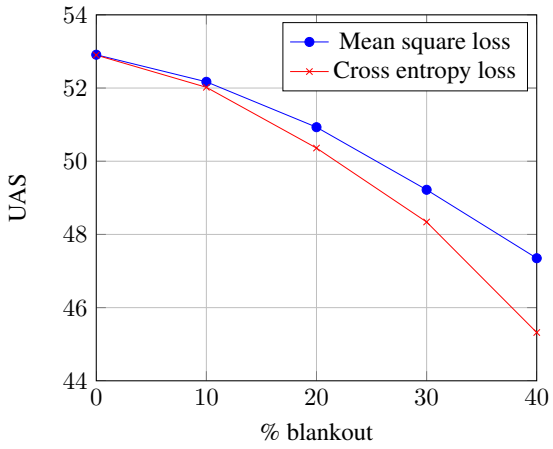


Figure 5: Performance for Tensor-LSTM on English test data with 0-40% of the edge scores artificially maintained at 0.

As can be clearly seen, performance drops faster with the percentage of deleted labels for the cross entropy model. This confirms our intuition that the initially lower performance using mean squared loss compared to cross entropy loss is mitigated by a greater robustness towards missing labels, gained by circumventing the decoding step in the training process. In Table 2, this is reflected as dramatic performance increases using mean squared error for Finnish, Persian, Hindi, and Hebrew – the four languages furthest

removed from the predominantly Indo-European source languages and therefore the four languages with the poorest projected label quality.

Several possible avenues for future work on this project are available. In this paper, we used an extremely simple feature function. More complex feature functions is one potential source of improvement. Another interesting direction for future work would be to include POS-tagging directly as a component of Tensor-LSTM prior to the construction of $S \boxplus S^*$ in a multi-task learning framework. Similarly, incorporating semantic tasks on top of dependency parsing could lead to interesting results. Finally, extensions of the Tensor-LSTM function to deeper models, wider models, or more connected models as seen in e.g. Kalchbrenner et al. (2015) may yield further performance gains.

6 Related Work

Experiments with neural networks for dependency parsing have focused mostly on learning higher-order scoring functions and creating efficient feature representations, with the notable exception of Fonseca et al. (2015). In their paper, a convolutional neural network is used to evaluate local edge scores based on global information. In Zhang and Zhao (2015) and Pei et al. (2015), neural networks are used to simultaneously evaluate first-order and higher-order scores for graph-based parsing, demonstrating good results. Bidirectional LSTM-models have been successfully applied to feature generation (Kiperwasser and Goldberg, 2016). Such LSTM-based features could in future work be employed and trained in conjunction with Tensor-LSTM, incorporating global information both in parsing and in featurization.

An extension of LSTM to tensor-structured data has been explored in Graves et al. (2007), and further improved upon in Kalchbrenner et al. (2015) in the form of GridLSTM. Our approach is similar, but simpler and computationally more efficient as no within-layer connections between the first and the second axes of the tensor are required.

Annotation projection for dependency parsing has been explored in a number of papers, starting with Hwa et al. (2005). In Tiedemann (2014) and Tiedemann (2015) the process in extended and evaluated across many languages. Li et al. (2014) follows the method of Hwa et al. (2005) and adds a probabilistic target-language classifier to determine and filter out high-uncertainty trees. In Ma and Xia (2014), performance on projected data is used as an additional objective for unsupervised learning through a combined loss function.

A common thread in these papers is the use of high-quality parallel data such as the EuroParl corpus. For truly low-resource target languages, this setting is unrealistic as parallel resources may be restricted to biased data such as the Bible. In Agić et al. (2016) this problem is addressed, and a parser is constructed which utilizes averaging over edge posteriors for many source languages to compensate for low-quality projected data. Our work builds upon their contribution by constructing a more flexible parser which can bypass a source of bias in their projected labels, and we therefore compared our results directly to theirs.

Annotation projection procedures for cross-lingual dependency parsing has been the focus of several other recent papers (Guo et al., 2015; Zhang and Barzilay, 2015; Duong et al., 2015; Rasooli and Collins, 2015). In Guo et al. (2015), distributed, language-independent feature representations are used to train shared parsers. Zhang and Barzilay (2015) introduce a tensor-based feature representation capable of incorporating prior knowledge about feature interactions learned from source languages. In Duong et al. (2015), a neural network parser is built wherein higher-level layers are shared between languages.

Finally, Rasooli and Collins (2015) leverage dense information in high-quality sentence translations to improve performance. Their work can be seen as opposite to ours – whereas Rasooli and Collins leverage high-quality translations to improve performance when such are available, we focus on improving performance in the *absence* of high-quality translations.

7 Conclusion

We have introduced a novel algorithm for graph-based dependency parsing based on an extension of sequence-LSTM to the more general Tensor-LSTM. We have shown how the parser with a cross entropy loss function performs comparably to state of the art for monolingual parsing. Furthermore, we have demonstrated that the flexibility of our parser enables learning from non well-formed data and from the output of other parsers. Using this property, we have applied our parser to a cross-lingual annotation projection problem

for truly low-resource languages, demonstrating an average target-language unlabeled attachment score of 48.54, which to the best of our knowledge are the best results yet for the task.

Acknowledgments

The second author was supported by ERC Starting Grant No. 313695.

References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4.

Theodore Bluche, Christopher Kermorvant, and Jerome Louradour. 2015. Where to apply dropout in recurrent neural networks for handwriting recognition? In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 681–685. IEEE.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 845–850. Association for Computational Linguistics.

Jack Edmonds. 1968. Optimum branchings. In *Mathematics and the Decision Sciences, Part 1*, pages 335–345. American Mathematical Society.

Erick R. Fonseca, Avenida Trabalhador São-carlense, and Sandra M. Aluísio. 2015. A deep architecture for non-projective dependency parsing. In *Proceedings of the 2015 NAACL-HLT Workshop on Vector Space Modeling for NLP*, pages 56–61. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 2010 International conference on Artificial Intelligence and Statistics*, pages 249–256. Society for Artificial Intelligence and Statistics.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2007. Multi-dimensional recurrent neural networks. *arXiv preprint arXiv:0705.2011*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1234–1244. Association for Computational Linguistics.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A Field Guide to Dynamic Recurrent Neural Networks*. IEEE press.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2342–2350. International Machine Learning Society.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv:1603.04351*.

Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word representations from sentence alignments. In *EACL*.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Soft cross-lingual syntax projection for dependency parsing. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 783–793. Association for Computational Linguistics.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1337––1348. Association for Computational Linguistics.

André F.T. Martins, Noah A. Smith, Eric P. Xing, Pedro M.Q. Aguiar, and Mário A.T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on*

Empirical Methods in Natural Language Processing, pages 62–72. Association for Computational Linguistics.

Ryan T. McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B. Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2015. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*.

Robert Östling. 2015. *Bayesian models for multilingual word alignment*. Ph.D. thesis, Department of Linguistics, Stockholm University.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 313–322. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338. Association for Computational Linguistics.

Milan Straka, Jan Hajič, Jana Straková, and Jan Hajič jr. 2015. Parsing universal dependency treebanks using neural networks and search-based oracle. In *Proceedings of the 14th International Workshop on Treebanks and Linguistic Theories*, pages 208–220. Association for Computational Linguistics.

Jörg Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1854–1864. Association for Computational Linguistics.

Jörg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted pos labels. *Proceedings of the Third International Conference on Dependency Linguistics*, pages 340–349.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4:2.

Dániel Varga, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. 2005. Parallel corpora for medium density languages. In *Proceedings of the 2005 Conference on Recent Advances in Natural Language Processing*. Association for Computational Linguistics.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1857–1867. Association for Computational Linguistics.

Zhisong Zhang and Hai Zhao. 2015. High-order graph-based neural dependency parsing. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 114–123. Association for Computational Linguistics.