SYNTACTIC PARSING OF MORPHOLOGICALLY RICH LANGUAGES USING DEEP NEURAL NETWORKS

Joël Legrand        Ronan Collobert

Idiap-RR-25-2015

JUNE 2015
Syntactic Parsing of Morphologically Rich Languages Using Deep Neural Networks

Joël Legrand
Idiap Research Institute
Martigny, Switzerland
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
joel.legrand@idiap.ch

Ronan Collobert *
Facebook AI Research
Menlo Park, CA, USA
Idiap Research Institute
Martigny, Switzerland
ronan@collobert.com

Abstract
This paper presents parsing results for the constituency track of the SPMRL shared task. We use the recurrent neural network model of (Legrand and Collobert, 2015) which leverages a new RNN-based compositional sub-tree representation. Results are provided for different scenarios were models are trained on the full corpus or on a subset of 5k sentences, using either the gold or the predicted POS tags. Our system outperforms the baseline and achieves significant improvements over the state of the art model for three languages, while being very fast due to its greedy nature.

1 Introduction
The SPMRL Shared Task 2013 (Seddah et al., 2013) provides standardized datasets, evaluation metrics and baseline results, in both constituency and dependency parsing, for nine different languages. This paper presents results for the constituency track, using the system introduced in (Legrand and Collobert, 2015). The model used is a greedy parser which leverages a new composition approach to keep a history of what has been predicted so far. The composition performs a syntactic and semantic summary of the contents of a sub-tree in form of a vector representation. The composition is performed along the tree: bottom tree node representations are obtained by composing continuous word and tag vector representations, and produces vector representations which are in turn composed together in subsequent nodes of the tree. The composition operation as well as tree node tagging and predictions are achieved with a Recurrent Neural Network (RNN). Both the composition and node prediction are trained jointly.

∗All research was conducted at the Idiap Research Institute, before Ronan Collobert joined Facebook AI Research.

Both the baseline (Berkeley parser) and the state-of-the-art (Björkelund et al., 2014) models rely on PCFG-based features. The latter uses a product of PCFG with Latent Annotations based models (Petrov, 2010), with a Coarse-to-Fine decoding strategy. The output is then discriminatively reranked (Charniak and Johnson, 2005) to select the best analysis. In contrast, our parser constructs the parse tree in a greedy manner and relies only on word and tag embeddings. Thanks to its greedy nature, our parser is very fast: it is able to parse around 80 (20 for its voting version) sentences per second on average (on a single CPU). Furthermore, as this model relies only on word and tag embeddings, it could be easily enhanced by leveraging unsupervised embeddings.

2 The Model
2.1 Greedy RNN Parsing

I_W : Did you hear the falling bombs ?
I_T : VBD PRP VB DT VBG NNS .
O : O S-NP O B-NP I-NP E-NP O

I_W : Did R1 hear R2 .
I_T : VBD NP VB NP .
O : O O B-VP E-VP .

I_W : Did R1 R3 ?
I_T : VDB NP VP .
O : B-SQ I-SQ L-SQ E-SQ

Figure 1: Greedy parsing algorithm (3 iterations), on the sentence “Did you hear the falling bombs ?”. I_W, I_T and O stand for input words (or composed word representations R_i), input syntactic tags (parazing or part-of-speech) and output tags (parazing), respectively.

The model of (Legrand and Collobert, 2015) is entirely based on neural networks and performs parsing in a greedy recurrent way. Our approach is a bottom-up iterative procedure: the tree is built
starting from the terminal nodes (sentence words), as shown in Figure 1. This greedy procedure is explained in detail in (Legrand and Collobert, 2015).

### 2.2 Word and Tag Embeddings

Each iteration of our parsing can be seen as a simple sequence tagging task. This is done using the model introduced in (Collobert and Weston, 2008) on various NLP tasks. This model relies on word and tag embeddings. Each word (resp. tag) in a finite dictionary $\mathcal{W}$ (resp. $\mathcal{T}$), is assigned a continuous vector representation which is, as all parameters of our architecture, trained by back-propagation. Note that we did not use pre-trained word embeddings as there were not any available for every language.

More formally, each word (resp. tag) is embedded in a $D$-dimensional (resp. $T$-dimensional) vector space by applying a lookup-table operation $LT_X(n) = X_n$, where $X$ is a $D \times |\mathcal{W}|$ (resp. $T \times |\mathcal{T}|$) matrix of parameters to be train. The column $X_n$ corresponds to the vector embedding of the $n^{th}$ word (resp. tag) in our dictionary $\mathcal{W}$ (resp. $\mathcal{T}$).

### 2.3 Word-Tag Composition

At each step of the parsing procedure, the tagger is fed with word and node representations. The node representation is a summary of the corresponding sub-tree. As shown in Figure 2, the vector representation is obtained by a simple recurrent procedure which outputs a representation living in the same space as the word representations (dimension $D$).

![Figure 2: Recurrent composition of the sub-tree (VP (VB hear) (NP (DT the) (VBG falling) (NNS bombs))). The representation $R_2$ is first computed using the 3-inputs module $C_3$ with the/DT falling/VBG bombs/NNS as input. $R_3$ is obtained by using the 2-inputs module $C_2$ with hear/VB R2/NP as input.](image)

### 2.4 Sliding Window BIOES Tagger

As illustrated in Figure 3, the tagging module of our architecture (see Figure 3) is a two-layer neural network which applies a sliding window of size $K$ over the input constituent representations (as computed in Section 2.3), as well as the input constituent tag representations. Considering

Compositional networks take as input both the merged node word or node representations (dimension $D$) and predicted tag representations (dimension $T$). There is one different network $C_k$ for each possible node with a number of $k$ merged constituent. In practice most tree nodes do not merge more than a few constituents. In our case, denoting $z \in \mathbb{R}^{(D+T)\times k}$ the concatenation of the merged constituent representations ($k$ vectors of tags and constituent representations), the compositional network is simply a matrix-vector operation followed by a non-linearity

$$C_k(z) = h(M^k z),$$

where $M^k \in \mathbb{R}^{D \times (k(D+T))}$ is a matrix of parameters to be trained, and $h(\cdot)$ is a simple non-linearity such as a pointwise hyperbolic tangent.

As the node and word representations are embedded in the same space, the compositional networks $C_k$ can compress information coming both from leaves and sub-trees. Similarly, the tagger network can be fed indifferently with word or sub-tree representations.
Table 1: Parseval results (the predicted POS were automatically predicted and provided with the corpus)

<table>
<thead>
<tr>
<th>team</th>
<th>Arabic</th>
<th>Basque</th>
<th>French</th>
<th>German</th>
<th>Hebrew</th>
<th>Hungarian</th>
<th>Korean</th>
<th>Polish</th>
<th>Swedish</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS:SZEGED:CIS</td>
<td>82.20</td>
<td>90.04</td>
<td>83.98</td>
<td>82.07</td>
<td>91.64</td>
<td>92.60</td>
<td>86.50</td>
<td>88.57</td>
<td>85.09</td>
<td>86.97</td>
</tr>
<tr>
<td>BASE:BKY+POS</td>
<td>80.76</td>
<td>76.24</td>
<td>81.76</td>
<td>80.34</td>
<td>92.20</td>
<td>87.64</td>
<td>82.95</td>
<td>88.13</td>
<td>82.89</td>
<td>83.66</td>
</tr>
<tr>
<td>BASE:BKY:RAW</td>
<td>79.14</td>
<td>69.78</td>
<td>80.38</td>
<td>78.99</td>
<td>87.32</td>
<td>81.44</td>
<td>73.28</td>
<td>79.51</td>
<td>78.94</td>
<td>78.75</td>
</tr>
<tr>
<td>This Model</td>
<td>80.75</td>
<td>82.65</td>
<td>81.06</td>
<td>81.47</td>
<td>91.65</td>
<td>89.54</td>
<td>86.12</td>
<td>93.16</td>
<td>81.06</td>
<td>85.27</td>
</tr>
<tr>
<td>This Model (v4)</td>
<td>82.06</td>
<td>84.42</td>
<td>82.38</td>
<td>82.79</td>
<td>91.95</td>
<td>90.56</td>
<td>87.48</td>
<td>93.55</td>
<td>81.93</td>
<td>86.34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>team</th>
<th>Arabic</th>
<th>Basque</th>
<th>French</th>
<th>German</th>
<th>Hebrew</th>
<th>Hungarian</th>
<th>Korean</th>
<th>Polish</th>
<th>Swedish</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS:SZEGED:CIS</td>
<td>79.47</td>
<td>88.45</td>
<td>82.25</td>
<td>74.78</td>
<td>91.64</td>
<td>91.87</td>
<td>80.10</td>
<td>88.18</td>
<td>85.09</td>
<td>84.65</td>
</tr>
<tr>
<td>BASE:BKY+POS</td>
<td>77.54</td>
<td>74.06</td>
<td>78.07</td>
<td>71.37</td>
<td>92.20</td>
<td>86.74</td>
<td>72.85</td>
<td>87.91</td>
<td>82.89</td>
<td>80.40</td>
</tr>
<tr>
<td>BASE:BKY:RAW</td>
<td>75.22</td>
<td>67.16</td>
<td>75.91</td>
<td>68.94</td>
<td>87.32</td>
<td>79.34</td>
<td>60.40</td>
<td>78.30</td>
<td>78.94</td>
<td>74.61</td>
</tr>
<tr>
<td>This Model</td>
<td>77.34</td>
<td>78.83</td>
<td>83.57</td>
<td>75.25</td>
<td>91.65</td>
<td>88.01</td>
<td>79.12</td>
<td>93.12</td>
<td>81.06</td>
<td>82.55</td>
</tr>
<tr>
<td>This Model (v4)</td>
<td>79.16</td>
<td>80.82</td>
<td>79.21</td>
<td>76.97</td>
<td>91.95</td>
<td>89.00</td>
<td>80.82</td>
<td>93.43</td>
<td>81.93</td>
<td>83.69</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>team</th>
<th>Arabic</th>
<th>Basque</th>
<th>French</th>
<th>German</th>
<th>Hebrew</th>
<th>Hungarian</th>
<th>Korean</th>
<th>Polish</th>
<th>Swedish</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS:SZEGED:CIS</td>
<td>81.32</td>
<td>87.86</td>
<td>81.83</td>
<td>81.27</td>
<td>89.46</td>
<td>91.85</td>
<td>84.27</td>
<td>87.55</td>
<td>83.99</td>
<td>85.49</td>
</tr>
<tr>
<td>BASE:BKY+POS</td>
<td>78.66</td>
<td>74.74</td>
<td>79.76</td>
<td>78.28</td>
<td>85.42</td>
<td>85.22</td>
<td>78.56</td>
<td>86.75</td>
<td>80.64</td>
<td>80.89</td>
</tr>
<tr>
<td>BASE:BKY:RAW</td>
<td>79.19</td>
<td>70.50</td>
<td>80.38</td>
<td>78.30</td>
<td>86.96</td>
<td>81.62</td>
<td>71.42</td>
<td>79.23</td>
<td>79.18</td>
<td>78.53</td>
</tr>
<tr>
<td>This Model</td>
<td>78.55</td>
<td>81.00</td>
<td>79.27</td>
<td>79.29</td>
<td>91.65</td>
<td>87.00</td>
<td>81.08</td>
<td>92.23</td>
<td>79.47</td>
<td>82.28</td>
</tr>
<tr>
<td>This Model (v4)</td>
<td>79.82</td>
<td>82.54</td>
<td>80.62</td>
<td>80.79</td>
<td>91.95</td>
<td>88.36</td>
<td>82.93</td>
<td>92.67</td>
<td>79.61</td>
<td>84.37</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>team</th>
<th>Arabic</th>
<th>Basque</th>
<th>French</th>
<th>German</th>
<th>Hebrew</th>
<th>Hungarian</th>
<th>Korean</th>
<th>Polish</th>
<th>Swedish</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS:SZEGED:CIS</td>
<td>78.85</td>
<td>86.65</td>
<td>79.83</td>
<td>75.61</td>
<td>89.46</td>
<td>90.53</td>
<td>78.47</td>
<td>87.46</td>
<td>83.99</td>
<td>83.21</td>
</tr>
<tr>
<td>BASE:BKY+POS</td>
<td>74.84</td>
<td>72.55</td>
<td>76.19</td>
<td>69.40</td>
<td>85.42</td>
<td>83.82</td>
<td>67.97</td>
<td>87.17</td>
<td>80.64</td>
<td>77.53</td>
</tr>
<tr>
<td>BASE:BKY:RAW</td>
<td>74.57</td>
<td>66.75</td>
<td>75.76</td>
<td>68.68</td>
<td>86.96</td>
<td>79.35</td>
<td>58.49</td>
<td>78.38</td>
<td>79.18</td>
<td>74.24</td>
</tr>
<tr>
<td>This Model</td>
<td>75.41</td>
<td>77.37</td>
<td>77.09</td>
<td>73.23</td>
<td>91.65</td>
<td>85.63</td>
<td>72.69</td>
<td>92.18</td>
<td>79.47</td>
<td>80.52</td>
</tr>
<tr>
<td>This Model (v4)</td>
<td>77.01</td>
<td>78.99</td>
<td>77.57</td>
<td>75.01</td>
<td>91.95</td>
<td>87.00</td>
<td>75.09</td>
<td>92.66</td>
<td>79.61</td>
<td>81.65</td>
</tr>
</tbody>
</table>

$N$ input constituents $X_1, \ldots, X_N$, for each constituent $X_n$, the network tagger is fed with the concatenation of the constituent’s word and tag representations and its $\frac{N}{2} - 1$ left and right neighbors’ word and tag representations. The network output a score for every possible BIOES-prefixed parsing tag.

### 2.5 Coherent BIOES Predictions

The next module of our architecture aggregates the BIOES-prefixed parsing tags from our tagger module in a coherent manner. It is implemented as a Viterbi decoding algorithm over a constrained graph $G$, which encodes all the possible valid sequences of BIOES-prefixed tags over constituents: e.g. $B$-$A$ tags can only be followed by $I$-$A$ or $E$-$A$ tags, for any parsing label $A$. Each node of the graph is assigned a score produced by the previous neural network module (score for each BIOES-prefixed tag, and for each word). The score $S((t)_1^N, [X]_1^N, \theta)$ for a sequence of tags $[t]_1^N$ in the lattice $G$ is simply obtained by summing scores along the path $([X]_1^N$ being the input sequence of constituents and $\theta$ all the parameters of the model). This decoding is present only to ensure coherence in the predicted sequence of tags.

Both the composition network and tagging networks are trained by maximizing a likelihood over the training data using stochastic gradient ascent. For detailed information about the training procedure and how the training set is built please read (Le Grand and Collobert, 2015).

The score $S((t)_1^N, [X]_1^N, \theta)$ of the true sequence of BIOES-prefixed tags $[t]_1^N$, given the input node sequence $[X]_1^N$ can be interpreted as a conditional probability by exponentiating this score (thus making it positive) and normalizing it with respect to all possible path scores. The log-probability of a sequence of tags $[t]_1^N$ for the input sequence of constituents $[X]_1^N$ is given by:

$$
\log P((t)_1^N| [X]_1^N, \theta) = S((t)_1^N, [X]_1^N, \theta) - \log \sum_{[t']_1^N} S((t')_1^N, [X]_1^N, \theta) .
$$

The second term of this equation (which correspond to the normalisation term) can be computed in linear time thanks to a recursion similar to the Viterbi algorithm (Rabiner, 1989).

### 3 Experiments

#### 3.1 Corpus

The corpus used to conduct our experiments is the Statistical Parsing of Morphologically Rich Languages (SPMRL) corpus provided for the shared task 2014 (Seddah et al., 2014). It provides sentences and tree annotations for 9 different languages (Arabic, Basque, French, German, Hebrew, Hungarian, Korean, Polish, Swedish), coming from various sources (Sima’an et al., 2001; Tsarfaty, 2010; Goldberg, 2011; Tsarfaty, 2013;
In this paper, we showed that a simple greedy model is able to outperform the base-line systems. Furthermore, we achieve significant improvements over the state-of-the-art model for several languages.

3.2 Training details

Our systems are trained using a stochastic gradient descent over the available training data. Hyper-parameters were tuned on the validation set. The dimension for the words embedding and tag embeddings were respectively 100 and 20. The window size for the tagger is $K = 7$ (3 neighbours from each side). The size of the tagger’s hidden layer were $H = 500$. All parameter were initialized randomly. As suggested in (Plaut and Hinton, 1987), the learning rate was divided by the size of the input vector of each layer. We used the same dropout regularization and voting procedure as in (Legrand and Collobert, 2015).

3.3 Results

Table 1 and 2 present the results obtained for the Parseval and the Leaf-ancestor metrics. We included a voting procedure using several models trained starting from different random initializations. At each iteration of the greedy parsing procedure, the BIOES-tag scores are averaged and the new node representations are computed for each model by composing the sub-tree representations corresponding to the given model, using its own compositional network.

We compare our system with the baseline provided with the task (Berkeley parser trained in two modes: with provided POS Tags (gold or predicted depending on dataset) and in Raw mode where the parser do its own POS tagging) and with the best (and only) participant of the task (Björkelund et al., 2014) which uses a product of PCFG-LA based model (Petrov, 2010) followed by a discriminative reranking (Charniak and Johnson, 2005).

4 Conclusion

In this paper, we showed that a simple greedy RNN-based model is able to outperform the baseline systems. Furthermore, we achieve significant improvements over the state-of-the-art model for several languages.
References


S. Petrov. 2010. Products of random latent variable grammars. In NAACL-HLT.


