Multilingual discriminative lexicalized parsing
Benoît Crabbé

To cite this version:
Benoît Crabbé. Multilingual discriminative lexicalized parsing. Empirical Methods in Natural Language Processing, 2015, Lisbon, Portugal. <hal-01186018>

HAL Id: hal-01186018
https://hal.inria.fr/hal-01186018
Submitted on 16 Dec 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Multilingual discriminative lexicalized phrase structure parsing

Benoit Crabbé
Alpage – Université Paris Diderot – Inria – IUF
Place Paul Ricoeur 75013 Paris
benoit.crabbe@univ-paris-diderot.fr

Abstract
We provide a generalization of discriminative lexicalized shift reduce parsing techniques for phrase structure grammar to a wide range of morphologically rich languages. The model is efficient and outperforms recent strong baselines on almost all languages considered. It takes advantage of a dependency based modelling of morphology and a shallow modelling of constituency boundaries.

1 Introduction
Lexicalized phrase structure parsing techniques were first introduced by Charniak (2000) and Collins (2003) as generative probabilistic models. Nowadays most statistical models used in natural language processing are discriminative: discriminative models provide more flexibility for modelling a large number of variables and conveniently expressing their interactions. This trend is particularly striking if we consider the literature in dependency parsing. Most state of the art multilingual parsers are actually weighted by discriminative models (Nivre and Scholz, 2004; McDonald et al., 2005; Fernández-González and Martins, 2015).

With respect to multilingual phrase structure parsing, the situation is quite different. Most parsers focus on fixed word order languages like English or Chinese as exemplified by Zhu et al. (2013). Despite a few exceptions (Collins et al., 1999), multilingual state of the art results are generally derived from the generative model of Petrov et al. (2006). Although more recently Hall et al. (2014) introduced a conditional random field parser that clearly improved the state of the art in the multilingual setting.

Both Petrov et al. (2006) and Hall et al. (2014) frame their parsing model to model in priority regular surfacic patterns and word order: Petrov et al. (2006) crucially infers category refinements (called category ‘splits’) in order to specialize the grammar on recurrent informative patterns observed on input spans. Hall et al. (2014) relies on a similar intuition: the model essentially aims to capture regularities on the spans of constituents and their immediate neighbourhood, following earlier intuitions of Klein and Manning (2004). This modelling strategy has two main motivations. First it reduces the burden of feature engineering, making it easier to generalize to multiple languages. Second it avoids modeling explicitly bilexical dependencies for which parameters are notoriously hard to estimate from small data sets such as existing treebanks.

On the other hand this strategy becomes less intuitive when it comes to modeling free word order languages where word order and constituency should in principle be less informative. As such, the good results reported by Hall et al. (2014) are surprising. It suggests that word order and constituency might be more relevant than often thought for modelling free word order languages.

Nevertheless, free word order languages also tend to be morphologically rich languages. This paper shows that a parsing model that can effectively take morphology into account is key for parsing these languages. More specifically, we show that an efficient lexicalized phrase structure parser - modelling both dependencies and morphology - already significantly improves parsing accuracy. But we also show that an additional modelling of spans and constituency provides additional robustness that contributes to yield state of the art results on almost all languages considered, while remaining quite efficient. Moreover, given the availability of existing multi-view treebanks (Bhatt et al., 2009; Seddah et al., 2013; Qiu et al., 2014), our proposed solution only requires a lightweight infrastructure to achieve multilin-
gual parsing without requiring costly language-dependent modifications such as feature engineering.

The paper is organized as follows. We first review the properties of multiview treebanks (Section 2). As these treebanks typically do not provide directly head annotation, an information required for lexicalized parsing, we provide an automated multilingual head annotation procedure (Section 3). We then describe in section 4 a variant of lexicalized shift reduce parsing that we use for the multilingual setting. It provides a way to integrate morphology in the model. Section 5 finally describes a set of experiments designed to test our main hypothesis and to point out the improvements over state of the art in multilingual parsing.

2 Multi-view treebanks

Multi-view treebanks are treebanks annotated both for constituents and dependencies that have the property to be token-wise aligned (Bhatt et al., 2009; Seddah et al., 2013; Qiu et al., 2014). These double annotations are typically obtained by converting a constituency or dependency annotation into the other annotation type. This method was used for the construction of the dataset for the SPMRL 2013 shared task (Seddahe et al., 2013), which contains multi-view treebanks for a number of morphologically rich languages, for which either constituency or dependency treebanks were available. The same kind of process was applied to the Penn TreeBank using the Stanford conversion system to produce dependency annotations (de Marneffe et al., 2006). In this paper, we use both of these datasets.

Although in multi-view treebanks each sentence is annotated both for constituency and dependency, they are not normalized for categories nor lexical features across languages such as dependencies in the Google Universal Treebank (McDonald et al., 2013). What is more, the dependency and constituency structures may sometimes strongly differ. For some languages, like Hungarian, the conversion has involved some manual re-annotation (Vincze et al., 2010).

3 Head annotation procedure

Lexicalized phrase structure parsers traditionally use hand-crafted heuristics for head annotation (Collins, 2003). Although these heuristics are available for some languages, for others they are non-existent or non-explicit and typically hidden in conversion procedures. In order to leverage the burden of managing language specific heuristics, we first automate head annotation by taking advantage of the multi-view annotation.

We begin by introducing some notation. Assuming a sentence $W = w_1 \ldots w_n$, the dependency annotation of this sentence is assumed to be a dependency forest (Kuhlmann and Nivre, 2006). A dependency graph $G = (V, E)$ where $V = \{1 \ldots n\}$ is the set of word indices or vertices and $E \subseteq V \times V$ is a set of dependency links. By convention, a dependency $(i, j)$ means that $i$ governs $j$. A dependency forest is a dependency graph such that a node has at most a single incoming edge and where there is no cycle. A node with no incoming edge is a root of the dependency forest and a dependency tree is a dependency forest with a single root. For some languages, such as German or Basque, the dependency structures found in the data set are actually dependency forests.

Lexicalized parsing relies on head annotation, in other words each node in a constituency tree is associated with the word index of its head. More formally, let $A$ be the set of nodes in the c-tree, head annotation can be represented as a function $h : A \mapsto \{1 \ldots n\}$ which maps each node $a \in A$ to the index of its head in the input sentence. $h$ is obtained by leveraging head-related information associated with each rule in the grammar. More precisely, each rule $\tau \rightarrow \gamma$, with $\gamma = a_1 \ldots a_k$, is associated with a head index $i$ ($1 \leq i \leq k$) that states that the head $h(\tau)$ of any node labeled $\tau$ in a constituency tree that is built using this rule is the same as the head of the right-hand side symbol $a_i$.

A Naive $h$ function is straightforwardly defined as the annotation of each local rule part of the tree in a bottom-up fashion:

\[
\begin{align*}
\text{Base:} & \quad h(w_i) = i \quad \forall w_i \in W \\
\text{Recurrence:} & \quad h(\tau) = \begin{cases} 
 h(a_i) \quad & \text{if } \forall a_j \in \tau, i \neq j (h(a_i), h(a_j)) \in E \\
 \bot & \text{otherwise}
\end{cases}
\end{align*}
\]

When $\exists a \in A$ such that $h(a) = \bot$ we say that the annotation has failed.

However the naive procedure fails in a large number of cases. Failures fall into the four patterns that are illustrated in Figure 1. For each of

---

1The additional case of unary rules is straightforward and left to the reader.
the patterns we have highlighted in bold the symbols for which the Naive procedure currently fails. Local Restructuration I is where the c-structure is flatter than the d-structure. Here the Naive procedure fails because \((a, c) \notin E\). Local Restructuration II is where the d-structure is flatter than the c-structure. The procedure fails because neither \((a, b) \in E\) or \((b, a) \in E\). Forest Effect is where the d-structure is a dependency forest (here \(E = \emptyset\)). And finally Non Projectivity is where the d-tree is non projective.

We can easily correct the naive procedure for Local Restructuration I by taking advantage of \(E^+\), the non reflexive transitive closure of \(E\), thus yielding the following Corrected procedure:

**Base:** \(h(w_i) = i\) \(^{∀w_i \in W}\)

**Recurrence:**

\[
h(τ) = \begin{cases} 
   h(a_i) & \text{if } \forall a_j, a_j \in γ, i \neq j (h(a_i), h(a_j)) \in E^+ \\
   & \text{otherwise}
\end{cases}
\]

The three other cases are more problematic, since their correction would somehow require altering the structure of either the c-tree or the d-tree. Refraining from altering the constituency data set we instead use a catch-all procedure that essentially creates the problematic head annotation by analogy with the rest of the data, yielding a fully Robust procedure that is guaranteed to succeed in any case:

**Base:** \(h(w_i) = i\) \(^{∀w_i \in W}\)

**Recurrence:**

\[
h(τ) = \begin{cases} 
   h(a_i) & \text{if } \forall a_j, a_j \in γ, i \neq j (h(a_i), h(a_j)) \in E^+ \\
   h_{\text{KNN}(τ → γ)} & \text{otherwise}
\end{cases}
\]

where \(\text{KNN}(τ → γ)\) is a function returning a guess for the position of the head in \(γ\), the right hand side of the rule, based on similarity to successfully head annotated rules.

The details are as follows. \(\text{KNN}(τ → γ)\) supposes a dataset \(D = (R_i, H_i)_{i=1}^N\) of successfully head annotated rules. In this dataset, each rule \(R_i = τ → γ\) is associated with \(H_i\) the position of the head in \(γ\). We define the similarity between two rules \(R_1 = τ^{(1)} → a_1^{(1)} \ldots a_k^{(1)}\) and \(R_2 = τ^{(2)} → a_1^{(2)} \ldots a_k^{(2)}\) to be the Levenshtein distance between \(τ^{(1)}, a_1^{(1)} \ldots a_k^{(1)}\) and \(τ^{(2)}, a_1^{(2)} \ldots a_k^{(2)}\). In practice for a given rule \(R\) the function returns the most frequent \(H\) among the 5 most similar rules in the data set.

The full head annotated data set \(D\) is built by reading off the rules from the trees successfully annotated in the treebank by the Corrected procedure in a first pass. A second pass yields the final annotation by running the Robust procedure.

**Analysis of the conversion** We report in Table 1 an overall quantification of the conversion procedure: % Success (Corrected) reports the number of trees successfully annotated by the Corrected procedure and Silver UAS reports an UAS score obtained by comparing the reference dependency trees to the conversion of those obtained from the Robust conversion of the head-annotated phrase structure trees back to dependency structures. The conversion works well apart from four languages (Arabic, Basque, German and Hungarian) which cause more difficulties.

<table>
<thead>
<tr>
<th>Language</th>
<th>% Success (Corrected)</th>
<th>Silver UAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>61.7</td>
<td>92.0</td>
</tr>
<tr>
<td>Basque</td>
<td>54.2</td>
<td>82.7</td>
</tr>
<tr>
<td>English</td>
<td>99.9</td>
<td>98.8</td>
</tr>
<tr>
<td>French</td>
<td>99.9</td>
<td>99.3</td>
</tr>
<tr>
<td>German</td>
<td>98.4</td>
<td>72.1</td>
</tr>
<tr>
<td>Hebrew</td>
<td>98.2</td>
<td>99.0</td>
</tr>
<tr>
<td>Hungarian</td>
<td>85.9</td>
<td>80.1</td>
</tr>
<tr>
<td>Korean</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>Polish</td>
<td>98.6</td>
<td>98.8</td>
</tr>
<tr>
<td>Swedish</td>
<td>99.6</td>
<td>98.8</td>
</tr>
</tbody>
</table>

Table 1: Quantification of the conversion

In order to better understand the problems faced by the conversion procedure, we manually inspected the errors returned by the Corrected pro-
procedure. For each language, we sampled 20 examples of failures encountered and we manually categorized the errors using the four patterns illustrated in Figure 1. Across languages, 49.9% of the errors come from the pattern Local Restructuration II and 50% from the pattern Forest effect and more surprisingly, we found only one example in our sample from the pattern Non projectivity in the Hungarian treebank. This overall average hides however an important variation across treebanks. The Forest effect is indeed massively found in the Basque\(^2\) (100%) and German treebanks and more marginally in the Hungarian data set. Most of the time, these are cases of short word sequences (2 to 5 tokens) where all nodes are annotated as roots of the dependency trees. The Local restructuration II mostly found in the Arabic, Hebrew and Polish treebanks and less frequently in Hungarian. Arabic and Hebrew tend to follow a binary annotation scheme partially inspired by X-Bar, hence creating additional constituent structures that are not directly inferable from the dependency annotation. Polish uses this restructuration in patterns involving coordination. More surprisingly, non projective patterns, which we expected to be a significant feature of these languages, remain marginal in comparison to annotation related idiosyncratic problems.

4 Parsing algorithm

This section provides an overview of the design of the constituent parsing system. There are three recent proposals for beam-based discriminative shift reduce parsing for phrase structure grammar with a structured perceptron and beam search (Zhu et al., 2013; Crabbé, 2014; Mi and Huang, 2015). All three proposals point out that for weighted phrase structure parsing, the shift reduce algorithm requires a special treatment of unary rules in order to compare derivations of the same length. They all provide different management schemes for these unaries.

The work described here draws on the LR algorithm introduced by Crabbé (2014), but provides a simpler algorithm, it precisely describes the management of unary rules and clarifies how spans and morphological information is represented (see section 5).

For each language, the grammar is induced from a treebank using the following preprocessing steps. The corpus is first head-annotated with the Robust head annotation procedure. Second, the treebank is head-markovized (order 0) and unary productions that do not emit tokens\(^3\) are collapsed into unique symbols. Once this has been done we assume that tokens to be parsed are a list of couples (tag, wordform). The preprocessing steps ensure the binarized treebank implicitly encodes a binary lexicalized grammar whose rules are either in Chomsky Normal Form (CNF) like in (a) \(X[h] \rightarrow A[x] B[h], X[h] \rightarrow A[h] B[x], X[t] \rightarrow t\) or are also of the form (b) \(X[h] \rightarrow A[h] t, X[t] \rightarrow A[h] t, X[h] \rightarrow t B[h], X[t] \rightarrow t B[h]\) where \(A, B, X\) are delexicalized non-terminals, \(h, x, t\) are tokens (terminals) and \(A[h], A[x] \ldots X[t]\) are lexicalized non-terminals. Given a grammar in CNF, we can prove that for a sentence of length \(n\), the number of derivation steps for a shift reduce parser is \(3n - 1\). However our tagset-preserving transformation also introduces rules of the form (b), which explains why the number of derivation steps may vary from \(2n - 1\) to \(3n - 1\).

To ensure that a derivation is of length \(3n - 1\), the parser forces each shift to be followed by either a unary reduction or an alternative dummy Ghost Reduction (GR). Given the pre-processed treebank we infer the set \(A\) of actions used by the parser. Let \(\Sigma\) be the set of non-terminal symbols (including temporary symbols) read off from the binary treebank. The set of actions contains one Shift (S), one Ghost Reduction (GR) a set of \(|\Sigma|\) unary reductions (RU-X), one for each symbol, a set of \(|\Sigma|\) binary left reductions (RL-X) and a set of \(|\Sigma|\) binary right reductions (RR-X) (see also Sagae and Lavie (2006) and Figure 3 for details).

The parser itself is organized around two data structures: a stack of symbols, \(S = \ldots |s_2|s_1|s_0\), whose topmost element is \(s_0\). Symbols are lexicalized non terminals or tokens of the form \(A[x]\). The second structure is a queue statically filled with tokens \(T = t_1 \ldots t_n\). Parsing is performed by sequentially generating configurations \(C\) of the form \(C(j, S, \cdot)\) where \(S\) is a stack and \(j\) is the index of the first element of the queue. Given an initial configuration \(C_0 = (1, \epsilon, \perp)\), a derivation step \(C_{t-1} \Rightarrow^{a_{t-1}} C_t\) generates a new configuration \(C_t\) by applying an action \(a_{t-1} \in A\) as defined in Figure 3. The derivation is complete and successful

\(^2\)The constituency conversion of the Basque treebank also contains a recurrent attachment error of the punctuations which we ignored when computing this statistic.

\(^3\)In order not to alter the tagset of the treebank.
In practice, we use a beam of size $K$ at each time step and lossy feature hashing, which makes the inference approximative.

For the purpose of computing weights, we extend the representation of the stack and queue elements such that the feature functions have access to a richer context than just simple lexicalized symbols of the form $A[x]$. As described in Figure 2 (left), features can also access the immediate left and right children of $s_0$ and $s_1$ as well as their left and right corner tokens. This allows us to encode the span models described in Section 5. We also use tuple-structured tokens encoding not only the word-form and the tag but also additional custom lexical features such as those enumerated in Figure 2 (right). This allows us to express the morphological models described in Section 5.

Finally, the parameters $w$ are estimated with a parallel averaged structured perceptron designed to cope with inexact inference (beam search): we specifically rely on max-violation updates of Huang et al. (2012) and on minibatches to accelerate and parallelize training (Shalev-Shwartz et al., 2007; Zhao and Huang, 2013).

## 5 Experiments

The experiments aim to compare the contribution of span based features approximating some intuitions of Hall et al. (2014) for shift reduce parsing and morphological features for parsing free word order languages. We start by describing the evaluation protocol and by defining the models used.

We use the standard SPMRL data set (Seddah et al., 2013). Part of speech tags are generated with Marmot (Müller et al., 2013), a CRF tagger specifically designed to provide tuple-structured tags. The training and development sets are tagged by 10-fold jackknifing. Head annotation is supplied by the Robust procedure described in Section 3. The parser is systematically trained for 25 epochs with a max violation update perceptron, a beam of size 8 and a minibatch size of 24.
To enable a comparison with other published results, the evaluation is performed with a version of evalb provided by the SPMRL organizers (Seddah et al., 2013) which takes punctuation into account.

**Baseline model (B)** The baseline model uses a set of templates identical to those of Zhu et al. (2013) for parsing English and Chinese except that we have no specific templates for unary reductions.

**Span-based model (B+S)** This model extends the B model by modeling spans. The span model approximates an intuition underlying Hall et al. (2014): constituent boundaries contain very informative tokens (typically function words). These tokens together with the pattern of their neighborhood provide key clues for detecting and (sub-)typing constituents. Moreover, parameter estimation for frequent functional words should suffer less from data sparseness issues than the estimation of billexical dependencies on lexical head words. The model includes conjunctions of non-terminal symbols on the stack with their left and right corners (words or tags) and also their immediately adjacent tokens across constituents. Using the notation given in Figure 2 we specifically included the following matrix templates:

\[
\begin{align*}
s_0, c_t & \land s_0, l.c, w & \land s_0, r.c, w \\
s_1, c_t & \land s_1, l.c, w & \land s_1, r.c, w \\
s_0, c_t & \land s_0, l.c, w & \land s_0, r.c, w \\
q_1, w & \land s_0, l.c, w & \land s_0, r.c, w \\
q_2, w & \land s_0, l.c, w & \land s_0, r.c, w
\end{align*}
\]

from which we derived additional backoff templates where only a single corner condition is expressed and/or words are replaced by tags.

**Morphological model (B+M)** This model extends the B model by adding morphological features. This model aims to approximate the intuition that morphological features such as case are key for identifying the structure of free word order languages. As feature engineering may become in principle quite complex once it comes to morphology, we targeted fairly crude models with the goal of providing a proof of concept. Therefore the morphologically informed models use as input a rich set of morphological features specified in Figure 2 (right) predicted by the CRF tagger (Müller et al., 2013) with the same jackknifing as before. The content of Figure 2 provides an explicit indication of the actual features defined in the original treebanks (see Seddah et al. (2013) and references therein for details), while the columns are indicative normalized names. For Basque most of the additional normalized names further encode case and verbal subcategorization. For French the mwe field abbreviates IOB predicted tags derived from multi-word expression annotations found in the original dataset.

Now let \( M \) be the set of values enumerated for a language in Figure 2 (right), we systematically added the following templates to model B:

\[
\begin{align*}
s_0, w_1, m & \land s_1, w_1, m & \land q_1, t, g \\
s_0, w_1, m & \land s_1, c_t & \land q_1, m \\
s_0, c_t, m & \land s_1, w_1, m & \land q_1, m \\
s_0, w_1, m & \land q_1, m & \land q_2, t, g \\
s_0, w_1, m & \land q_1, t, g & \land q_2, m \\
s_0, c_t & \land q_1, m & \land q_2, m
\end{align*}
\]

Essentially the model expresses interactions between morphological features from the constituent heads on the top of the stack and the morphological features from the tokens at the beginning of the queue.

**Mixed model (B+S+M)** Our last model is the union of the span model (B+S) and the morphological model (B+M).

**Results (development)** We measured the impact of the model variations on the development set for c-parsing on the SPMRL data sets (Table 2). We immediately observe that modelling spans tends to improve the results, in particular for languages where the head annotation is more problematic: Arabic\(^4\), Basque, German and Hungarian and also Swedish however. So the span-based model seems to improve the parser’s robustness in cases when dependencies lack precision. For this model, the average behaviour is similar to that of Hall et al. (2014) although the variance is high.

On the other hand, the morphological model tends to be most important for languages where head annotation is easier: French, Korean, Polish and Swedish. It is key for very richly inflected languages such as Basque and Hungarian even though our head annotation is more approximative\(^5\).

\(^4\)Although not detailed in the paper, we also observe that for Arabic, the morphological features are generally predicted with a lower accuracy by the tagger than for other languages.

\(^5\)As annotation schemes are not normalized across languages, it is important to stress that these observations are very unlikely to be representative of the linguistic properties of these languages. They are more likely to be a result of annotation choices. For example Korean is a strongly agglutinative language for which much of the morphology is already encoded in the tag set.
comparison with Hall et al. (2014) also reveals that for Basque, Hungarian and Swedish, taking into account morphological information largely explains our improved results.

**Results (test)** We observe in Table 3 that our joint B+S+M model yields a state of the art c-parser on almost all languages considered. It is quite clear that both our span and morphology enhanced models could be dramatically improved, but it shows that with reasonable feature engineering, these two sub-models are largely sufficient to improve the state of the art in c-parsing for these languages over strong baselines. Although in principle the Berkeley parsers (Petrov et al., 2006; Hall et al., 2014) are designed to be language-generic with an underlying design that is surprisingly accurate for free word order languages end up suffering from a lack of sensitivity to morphological information. Finally we also observe that our phrase structure parser clearly outperforms the TurboParser setup described by Fernández-González and Martins (2015) in which an elaborate output conversion procedure generates c-parse from d-parse.

**Comparison with related work** We conclude with a few comparisons with related work. This will enable us to show that our approach is not only accurate but also efficient. A comparison with dependency parsers will also allow us to better identify the properties of our proposal.

In order to test efficiency, we compared our parser to c-parsers trained on Penn Treebank (PTB) for which we have running times reported by Fernández-González and Martins (2015). This required first assigning heads, for which we used the Stanford tool for converting PTB to Basic Dependencies, and then using our Robust conversion method. We performed a simple test using the PTB standard split with the same experimental setting as before, except that we use the standard evalb scorer (Table 5). Although the time-comparison remains indicative, it is clear that the parsing framework described in this paper is not only reasonably accurate on a fixed word order language such as English but it is also quite efficient. Parsing accuracies might be different with other head annotation schemes (see e.g. Elming et al. (2013) for illustrations). In our case, we compare the (B+S) model with automated head annotation to the Collins head annotation as implemented in the Standord CORE NLP library (Manning et al., 2014), where we can see that the Collins handcrafted head annotation yields better results than the automated one on English.

The question is now to which extend c-trees encode meaningful dependencies? As lexicalized c-trees encode unlabeled dependency trees, our parser also directly outputs unlabeled d-trees by

<table>
<thead>
<tr>
<th>System (single parsers)</th>
<th>F1 (EVALB)</th>
<th>(Toks/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hall-Klein 14</td>
<td>88.6</td>
<td>12</td>
</tr>
<tr>
<td>StanfordSR</td>
<td>89.1</td>
<td>655</td>
</tr>
<tr>
<td>Charniak 00</td>
<td>89.5</td>
<td>-</td>
</tr>
<tr>
<td><strong>This paper (B+S)</strong></td>
<td>89.7</td>
<td>2150°</td>
</tr>
<tr>
<td><strong>This paper (B+S) [Collins]</strong></td>
<td>90.0</td>
<td>2150°</td>
</tr>
<tr>
<td>Petrov 06</td>
<td>90.1</td>
<td>169</td>
</tr>
<tr>
<td>Fernandez-Martins 15</td>
<td>90.2</td>
<td>957</td>
</tr>
<tr>
<td>Zhu et al. 13</td>
<td>90.4</td>
<td>1290</td>
</tr>
</tbody>
</table>

Alls scores and times except ◦ are measured by Fernández-González and Martins (2015) on an intel Xeon 2.3Ghz. ◦ denotes the use of a different architecture (2.4Ghz intel).

### Table 2: Development F-scores

<table>
<thead>
<tr>
<th>Model</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>1 Base</td>
<td>79.46</td>
<td>74.67</td>
<td>79.66</td>
<td>82.61</td>
<td>90.43</td>
<td>84.34</td>
<td>81.96</td>
<td>91.68</td>
<td>75.60</td>
<td>82.26</td>
</tr>
<tr>
<td>2 Base+S</td>
<td>80.59</td>
<td>76.39</td>
<td>80.15</td>
<td>83.63</td>
<td>90.63</td>
<td>85.62</td>
<td>82.21</td>
<td>91.75</td>
<td>77.49</td>
<td>83.16</td>
</tr>
<tr>
<td>3 Base+M</td>
<td>80.17</td>
<td>83.69</td>
<td><strong>81.05</strong></td>
<td>83.66</td>
<td>90.40</td>
<td>87.75</td>
<td>82.79</td>
<td>92.72</td>
<td>77.50</td>
<td>84.41</td>
</tr>
<tr>
<td>4 Base+S+M</td>
<td><strong>81.25</strong></td>
<td><strong>84.01</strong></td>
<td>80.87</td>
<td><strong>84.08</strong></td>
<td><strong>90.69</strong></td>
<td><strong>88.27</strong></td>
<td><strong>83.09</strong></td>
<td><strong>92.78</strong></td>
<td><strong>77.87</strong></td>
<td><strong>84.77</strong></td>
</tr>
<tr>
<td>5 Hall-Klein 14</td>
<td>78.89</td>
<td>83.74</td>
<td>79.40</td>
<td>83.28</td>
<td>88.06</td>
<td>87.44</td>
<td>81.85</td>
<td>91.10</td>
<td>75.95</td>
<td>83.30</td>
</tr>
</tbody>
</table>

F1-scores provided by evalb-spmrl (Seddah et al., 2013). Takes punctuation into account and penalizes unparsed sentences.
simply reading them off from the lexicalized c-structure. We report in Table 4 the UAS evaluation of those dependencies and we compare them to the best results obtained by dependency parsers in both SPMRL13 and SPMRL14 shared tasks. For each language, the comparison is made with the best single dependency parsing system. For English we compare against Standard TurboParser - which seems to be the most similar to our system- when parsing to Basic Stanford dependencies. The comparison with semi-supervised and ensemble parsers still provides a reasonable upper-line (Björkelund et al., 2013).

As can be seen in Table 4, our results partly generalize the observation summarized by Cer et al. (2010) and Kong and Smith (2014) that phrase structure parsers tend to provide better dependencies than genuine dependency parsers for parsing to Stanford Dependencies. For English, our UAS is similar to that of TurboParser, but in a broader multilingual framework, the left side of the table shows that the unlabeled dependencies are clearly better than those of genuine dependency parsers. On the right side of the table are languages for which our dependencies are actually worse. This is not a surprise, since these are also the languages for which head annotation was more problematic in the first place. This last observation suggests that a lexicalized c-parser can also provide very accurate dependencies. A way to further generalize this observation to problematic languages would be either to design a less immediate post-processing conversion scheme or to further normalize the data set to obtain the correct heads from the outset.

6 Conclusion

Lexicalized phrase structure parsing of morphologically rich languages used to be difficult since existing implementations targeting essentially English or Chinese do not allow a straightforward integration of morphology. Given multi-view treebanks, we achieve multilingual parsing with a language-agnostic head annotation procedure. Once this procedure has created the required data representation for lexicalized parsing, only modest and weakly language dependent feature engineering is required to achieve state-of-the-art accuracies on all languages considered: a minimal interface with morphology already contributes to improving accuracy, and this is specifically the case when heads are accurately identified. When heads are only approximatively identified, span-based configurational modelling tends to correct the approximation.

Leaving aside details concerning conversion and data normalization, we generally found that the unlabeled dependencies modelled by the lexicalized c-parser also tend to be highly accurate. For languages where c-annotations and d-annotations are less compatible, additional language renormalizations would help to get better comparisons.

<table>
<thead>
<tr>
<th>Parser (single)</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>Petrov 06</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>79.19</td>
<td>78.45</td>
</tr>
<tr>
<td>Petrov 06 + tags</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>81.17</td>
</tr>
<tr>
<td>Hall-Klein 14</td>
<td>78.75</td>
<td>83.39</td>
<td>79.70</td>
<td>78.43</td>
<td>87.18</td>
<td>88.25</td>
<td>80.18</td>
<td>90.66</td>
<td>82.00</td>
<td>83.72</td>
</tr>
<tr>
<td>Fernandez-Martins 15</td>
<td>85.90</td>
<td>88.97</td>
<td>88.16</td>
<td>87.41</td>
<td>91.20</td>
<td>92.66</td>
<td>83.24</td>
<td>85.42</td>
<td>86.72</td>
<td>87.72</td>
</tr>
</tbody>
</table>

Table 3: Multilingual test (F-scores, phrase structure parsing)

<table>
<thead>
<tr>
<th>System</th>
<th>English</th>
<th>French</th>
<th>German</th>
<th>Hebrew</th>
<th>Polish</th>
<th>Swedish</th>
<th>Arabic</th>
<th>Basque</th>
<th>German</th>
<th>Hungarian</th>
</tr>
</thead>
<tbody>
<tr>
<td>This paper (B+S+M)</td>
<td>91.75</td>
<td>86.68</td>
<td>87.22</td>
<td>85.28</td>
<td>88.61</td>
<td>86.22</td>
<td>80.64</td>
<td>73.68</td>
<td>67.20</td>
<td>74.46</td>
</tr>
<tr>
<td>Best d-parser (single)</td>
<td>91.95</td>
<td>85.80</td>
<td>85.84</td>
<td>81.05</td>
<td>88.12</td>
<td>84.54</td>
<td>84.57</td>
<td>84.33</td>
<td>87.65</td>
<td>83.71</td>
</tr>
<tr>
<td>Best semi/ensemble</td>
<td>89.19</td>
<td>89.10</td>
<td>87.41</td>
<td>91.75</td>
<td>88.48</td>
<td>88.32</td>
<td>89.96</td>
<td>91.64</td>
<td>89.81</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Multilingual test (UAS, dependency parsing)

8In practice it turns out that these are either DIALOG-St (de La Clergerie, 2013) or sometimes MALTOPTIMIZER (Ballesteros and Nivre, 2012)
As suggested in this paper, future work for parsing morphologically rich languages will require to focus both on feature selection and on the interface between syntax and morphology, which means in our case the interface between the segmenter, the tagger and the parser.

Acknowledgments

The author wishes to thank Djamé Seddah for insightful discussions regarding the work reported in this paper as well as R. Bawden, M. Coavoux and B. Sagot for their careful proofreading.

References


