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-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1847–1856,
Lisbon, Portugal, 17-21 September 2015.

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 (Seddah 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 accross 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 reannotation (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 $W = w_1 \dots w_n$, the dependency annotation of this sentence is assumed to be a dependency forest (Kuhlmann and Nivre, 2006). A dependency graph $G = \langle V, E \rangle$ where $V = \{1 \dots n\}$ is the set of word indexes 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 $h : A \mapsto \{1 \dots n\}$ 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 $\gamma = a_1 \dots a_k$, is associated with a head index i ($1 \leq i \leq k$) that states that the head $h(\tau)$ of any node labeled τ 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:[1]

Base: $h(w_i) = i \qquad \forall w_i \in W$
Recurrence:

$$h(\tau) = \begin{cases} h(a_i) & \text{if } \forall_{a_j : a_j \in \gamma, i \neq j} \, (h(a_i), h(a_j)) \in E \\ \perp & \text{otherwise} \end{cases}$$

When $\exists a \in A$ such that $h(a) = \perp$ 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

[1] The additional case of unary rules is straightforward and left to the reader.

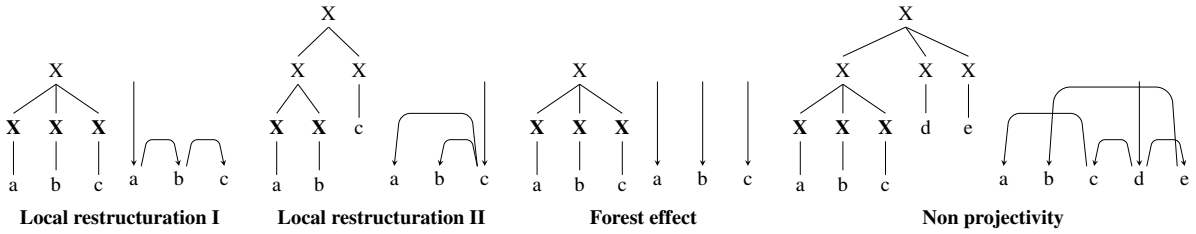


Figure 1: Patterns of the causes of problems taking place during head annotation

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 $(a, b) \in E$ nor $(b, a) \in E$. *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: $h(w_i) = i \qquad \forall w_i \in W$
Recurrence:

$$h(\tau) = \begin{cases} h(a_i) & \text{if } \forall_{a_j : a_j \in \gamma, i \neq j} (h(a_i), h(a_j)) \in E^+ \\ \perp & \text{otherwise} \end{cases}$$

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: $h(w_i) = i \qquad \forall w_i \in W$
Recurrence:

$$h(\tau) = \begin{cases} h(a_i) & \text{if } \forall_{a_j : a_j \in \gamma, i \neq j} (h(a_i), h(a_j)) \in E^+ \\ h(a_{\text{KNN}(\tau \rightarrow \gamma)}) & \text{otherwise} \end{cases}$$

where KNN($\tau \rightarrow \gamma$) 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. KNN($\tau \rightarrow \gamma$) supposes a dataset $D = (R_i, \mathcal{H}_i)_{i=1}^N$ of successfully head annotated rules. In this dataset, each rule $R_i = \tau \rightarrow \gamma$ is associated with $\mathcal{H}_i$ the position of the head in γ. We define the similarity between two rules $R_1 = \tau^{(1)} \rightarrow a_1^{(1)} \dots a_k^{(1)}$ and $R_2 = \tau^{(2)} \rightarrow a_1^{(2)} \dots a_{k'}^{(2)}$ to be the Levenshtein distance between $\tau^{(1)}, a_1^{(1)} \dots a_k^{(1)}$ and $\tau^{(2)}, a_1^{(2)} \dots a_{k'}^{(2)}$. In practice for a given rule R the function returns the most frequent $\mathcal{H}$ 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 succesfully 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.

Language	% Success (**Corrected**)	Silver UAS
ARABIC	61.7	92.0
BASQUE	54.2	82.7
ENGLISH	99.9	98.8
FRENCH	99.9	99.3
GERMAN	98.4	72.1
HEBREW	98.2	99.0
HUNGARIAN	85.9	80.1
KOREAN	100.0	100.0
POLISH	98.6	98.8
SWEDISH	99.6	98.8

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-

cedure. 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 suprisingly, 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 annoted as roots of the dependency trees. The *Local restructuration II* is 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 inferrable 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 idiosyncrasic 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).

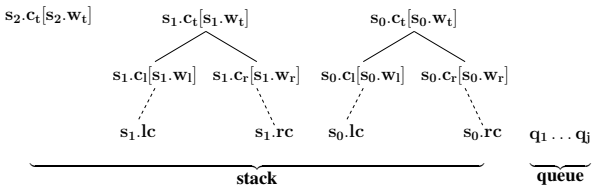
[2]The constituency conversion of the Basque treebank also contains a recurrent attachment error of the punctuations which we ignored when computing this statistic.

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 Σ 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, $\mathbf{S} = \ldots|s_2|s_1|s_0$, 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 $\langle j, \mathbf{S}, \cdot \rangle$ where $\mathbf{S}$ is a stack and j is the index of the first element of the queue. Given an initial configuration $C_0 = \langle 1, \epsilon, \perp \rangle$, a derivation step $C_{t-1} \overset{a_{t-1}}{\Rightarrow} C_t$ 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

[3]In order not to alter the tagset of the treebank.



Language	gen	num	case	mood	aspect	other
ARABIC	gender	number	case	mood	aspect	-
BASQUE	-	NUM	KAS	MDN	ASP	DADUDIO,ERL,NOR(I|K)?
ENGLISH	-	-	-	-	-	-
FRENCH	g	n	-	m	-	mwe
GERMAN	gender	number	case	mood	-	-
HEBREW	gen	num	-	-	-	tense
HUNGARIAN	-	Num	Cas	Mood	-	SubPOS
KOREAN	-	-	case-type	-	-	verb-type
POLISH	gender	number	case	-	aspect	post-prepositionality
SWEDISH	gender	number	case	verbform	-	perfectform

Figure 2: Features available for scoring. $\mathbf{s_x}$ denote a position in the stack. Stack positions are local trees of depth 1, features can access its **t**op, **l**eft and **r**ight nodes. The suffixes $\mathbf{c_p}, \mathbf{w_p}, \mathbf{lc}, \mathbf{rc}$ denote respectively the delexicalized category, the head token, the left corner token, the right corner token of a stack position. For tokens elements accessible from the stack ($\mathbf{s_x.w_x}$) and from the queue ($\mathbf{q_x}$), features can access the **word** form, pos **tag** or any morphological feature **m** available for that language as described in the table at the right

$$\textsc{Init} \quad \langle 1, \epsilon, \bot \rangle : 0$$
$$\textsc{Goal} \quad \langle n+1, \tau, \bot \rangle : w$$

$$\textsc{Shift} \quad \frac{\langle j, \mathbf{S}, \bot \rangle : w}{\langle j+1, \mathbf{S} \mid t_j.tag[t_j.word]), \top \rangle : w + F(S, \langle j, \mathbf{S} \rangle)}$$

$$\text{RL(X)} \quad \frac{\langle j, \mathbf{S}_\ominus \mid c_1[t_1]\ c_0[t_0], \bot \rangle : w}{\langle j, \mathbf{S}_\ominus \mid X[t_1], \bot \rangle : w + F(RL(X), \langle j, \mathbf{S} \rangle)}$$

$$\text{RR(X)} \quad \frac{\langle j, \mathbf{S}_\ominus \mid c_1[t_1]\ c_0[t_0], \bot \rangle : w}{\langle j, \mathbf{S}_\ominus \mid X[t_0], \bot \rangle : w + F(RR(X), \langle j, \mathbf{S} \rangle)}$$

$$\text{RU(X)} \quad \frac{\langle j, \mathbf{S}_\ominus \mid c_0[t_0], \top \rangle : w}{\langle j, \mathbf{S}_\ominus \mid X[t_0], \bot \rangle : w + F(RU(X), \langle j, \mathbf{S} \rangle)}$$

$$\text{GR(X)} \quad \frac{\langle j, \mathbf{S}_\ominus \mid c_0[t_0], \top \rangle : w}{\langle j, \mathbf{S}_\ominus \mid c_0[t_0], \bot \rangle : w + F(GR, \langle j, \mathbf{S} \rangle)}$$

Figure 3: Weighted inference rules

once the action C_{3n-1} is generated. A derivation sequence $C_{0 \Rightarrow \tau}$ is a sequence of derivation steps $C_0 \overset{a_0}{\Rightarrow} \dots \overset{a_{\tau-1}}{\Rightarrow} C_\tau$

Weighted prediction The choice of the action $a \in A$ at each derivation step is naturally non-deterministic. Determinism is provided by a weighting function based on a linear model of the form:

$$W(C_{0 \Rightarrow \tau}) = \sum_{i=0}^{\tau-1} \mathbf{w} \cdot \mathbf{\Phi}(a_i, C_i) = \sum_{i=0}^{\tau-1} F(a_i, C_i)$$

where $\mathbf{w} \in \mathbb{R}^d$ is a weight vector and $\mathbf{\Phi}(a_i, C_i) \in \{0,1\}^d$ is a feature vector. The best parse is then the successful derivation with the maximum score:

$$\hat{C}_{0 \Rightarrow 3n-1} = \underset{C_{0 \Rightarrow 3n-1} \in \text{GEN}_{3n-1}}{\text{argmax}} \quad W(C_{0 \Rightarrow 3n-1})$$

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 $\mathbf{w}$ 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 bilexical dependencies on lexical head words. The model includes conjunctions of nonterminal 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 :

$$\mathbf{s_0.c_t\&s_0.lc.word\&s_0.rc.word}$$
$$\mathbf{s_1.c_t\&s_1.lc.word\&s_1.rc.word}$$
$$\mathbf{s_0.c_t\&s_0.lc.word\&s_1.rc.word}$$
$$\mathbf{q_1.word\&s_0.lc.word\&s_0.rc.word}$$
$$\mathbf{q_2.word\&s_0.lc.word\&s_0.rc.word}$$

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 jackkniffing 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 morphological features 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**:

$$\begin{array}{ll}
\mathbf{s_0.w_t.m\&s_1.w_t.m\&q_1.tag} & \forall \mathbf{m} \in M \\
\mathbf{s_0.w_t.m\&s_1.c_t\&q_1.m} & \forall \mathbf{m} \in M \\
\mathbf{s_0.c_t.m\&s_1.w_t.m\&q_1.m} & \forall \mathbf{m} \in M \\
\mathbf{s_0.w_t.m\&q_1.m\&q_2.tag} & \forall \mathbf{m} \in M \\
\mathbf{s_0.w_t.m\&q_1.tag\&q_2.m} & \forall \mathbf{m} \in M \\
\mathbf{s_0.c_t\&q_1.m\&q_2.m} & \forall \mathbf{m} \in M
\end{array}$$

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]. A

[4] Although not detailed in the paper, we also observe that for Arabic, the morphological features are generally predicted with a lower accurracy 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.

	Model	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
1	Base	79.46	74.67	79.66	82.61	90.43	84.34	81.96	91.68	75.60	82.26
2	Base+S	80.59	76.39	80.15	83.63	90.63	85.62	82.21	91.75	77.49	83.16
3	Base+M	80.17	83.69	**81.05**	83.66	90.40	87.75	82.79	92.72	77.50	84.41
4	Base+S+M	**81.25**	**84.01**	80.87	**84.08**	**90.69**	**88.27**	**83.09**	**92.78**	**77.87**	**84.77**
5	Hall-Klein 14	78.89	83.74	79.40	83.28	88.06	87.44	81.85	91.10	75.95	83.30

F1-scores provided by `evalb-spmrl` (Seddah et al., 2013). Takes punctuation into account and penalizes unparsed sentences.

Table 2: Development F-scores

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[6]. 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-parses from d-parses.

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 used 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[7].

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

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 5: Penn treebank test (WSJ 23)

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

[6]For Basque, our problem comes from a recurrent inconsistency in the SPMRL data set. As annotated in the c-trees, the punctuation induces a modification of the d-structure: c-trees encode a different governor for punctuation marks than d-trees. This not only causes problem to our head annotation procedure but also for the parser to solving these attachments. A simple correction results in a significant improvement of these parsing results. However we decided to leave the data untouched in order to preserve fair comparisons with other systems.

[7]This pattern does not seem to be systematic: on French we could also compare with head annotations described in (Arun and Keller, 2005) and we observed a slight improvement when using the automated procedure.

Parser (single)	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
Petrov 06	79.19	70.50	80.38	78.30	86.96	81.62	71.42	79.23	79.19	78.45
Petrov 06 + tags	78.66	74.74	79.76	78.28	85.42	85.22	78.56	86.75	80.64	81.17
Hall-Klein 14	78.75	83.39	79.70	78.43	87.18	88.25	80.18	90.66	82.00	83.72
Fernandez-Martins 15	-	**85.90**	78.75	78.66	88.97	88.16	79.28	91.20	82.80	84.22
This paper (B+S+M)	**81.31**	84.94	**80.84**	**79.26**	**89.65**	**90.14**	**82.65**	**92.66**	**83.24**	**85.42**
Best semi/ensemble	81.32	88.24	82.53	81.66	89.80	91.72	83.81	90.50	85.50	86.72

F-scores provided by `evalb-spmrl` (Seddah et al., 2013). It takes punctuation into account and penalizes unparsed sentences. The average ignores Arabic for comparison with TurboParser. Petrov 06 + tags is the Berkeley parser with externally predicted pos tags (Seddah et al., 2013)

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

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

Unlabeled Accuracy Scores. Best other is the best single parser UAS result reported either in SPMRL 13 or SPMRL 14 shared tasks.
Best ensemble is the best semi-supervised or ensemble system from either SPMRL 13 or SPMRL 14 (Björkelund et al., 2013; Björkelund et al., 2014).

Table 4: Multilingual test (UAS, dependency parsing)

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[8]. 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 upperline (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.

[8]In practice it turns out that these are either DYALOG-SR (de La Clergerie, 2013) or sometimes MALTOPTIMIZER (Ballesteros and Nivre, 2012)

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.

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 insigthful discussions regarding the work reported in this paper as well as R. Bawden, M. Coavoux and B. Sagot for their careful proofreading.

References

Abhishek Arun and Frank Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: The case of french. In *Association for Computational Linguistics*.

Miguel Ballesteros and Joakim Nivre. 2012. Maltoptimizer: An optimization tool for maltparser. In *13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Sharma, and Fei Xia. 2009. A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*.

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Anders Björkelund, Özlem Cetinoğlu, Agnieszka Faleńska, Richárd Farkas, Thomas Müller, Wolfgang Seeker, and Zsolt Szántó. 2014. The ims-wrocaw-szeged-cis entry at the spmrl 2014 shared task: Reranking and morphosyntax meet unlabeled data. In *Fifth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Daniel M. Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *Proceedings of the Language Ressources and Evaluation Conference (LREC)*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *ANLP*, pages 132–139.

Michael Collins, Jan Hajic, Lance A. Ramshaw, and Christoph Tillmann. 1999. A statistical parser for Czech. In *27th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Benoit Crabbé. 2014. An LR-inspired generalized lexicalized phrase structure parser. In *25th International Conference on Computational Linguistics (COLING)*.

Eric Villemonte de La Clergerie. 2013. Exploring beam-based shift-reduce dependency parsing with dyalog: Results from the spmrl 2013 shared task. In *4th Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL2013)*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. 2006. Generating typed dependency parses from phrase structure parses. In *Language Ressources and Evaluation Conference (LREC)*.

Jakob Elming, Anders Johannsen, Sigrid Klerke, Emanuele Lapponi, Hector Martinez, and Anders Sogaard. 2013. Down-stream effects of tree-to-dependency conversions. In *Proceedings of NAACL-HLT*.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceeding of the Association of Computational Linguistics (ACL), 2015*.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 228–237.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (NAACL-HLT)*.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 478–485.

Lingpeng Kong and Noah A. Smith. 2014. An empirical comparison of parsing methods for stanford dependencies. *CoRR*, abs/1404.4314.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL/COLING)*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Ryan T. McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Ryan McDonald, J. Nivre, Y. Quirmbach-Brundage, Y. Goldberg, D. Das, K. Ganchev, K. Hall, S. Petrov, H. Zhang, O. Tackstrom, C. Bedini, N. Bertomeu Castello, and J. Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceeding of the Association of Computational Linguistics (ACL)*.

Haitao Mi and Liang Huang. 2015. Shift-reduce constituency parsing with dynamic programming and pos tag lattice. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order crfs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *COLING 2004, 20th International Conference on Computational Linguistics*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL/COLING)*.

Likun Qiu, Yue Zhang, Peng Jin, and Houfeng Wang. 2014. Multi-view chinese treebanking. In *25th International Conference on Computational Linguistics (COLING)*.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL/COLING)*.

Djamé Seddah, Reut Tsarfaty, Sandra Kubler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Yuval Marton, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliski, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the spmrl 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth SPMRL Workshop, Seattle, USA*.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for SVM. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML)*.

Veronika Vincze, Dora Szauter, Attila Almasi, Gyorgy Mora, Zoltan Alexin, and Janos Csirik. 2010. Hungarian dependency treebank. In *Proceedings of Language Ressources and Evalutation Conference (LREC)*.

Kai Zhao and Liang Huang. 2013. Minibatch and parallelization for online large margin structured learning. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (NAACL-HLT)*.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, (ACL)*.