Data-Driven Morphological Analysis and Disambiguation for Morphologically Rich Languages and Universal Dependencies

Amir More
IDC Herzliya
habeanf@gmail.com

Reut Tsarfaty
Open University of Israel
reutts@openu.ac.il

Abstract

Parsing texts into *universal dependencies (UD)* in realistic scenarios requires infrastructure for *morphological analysis and disambiguation (MA&D)* of typologically different languages as a first tier. MA&D is particularly challenging in *morphologically rich languages (MRLs)*, where the ambiguous space-delimited tokens ought to be disambiguated with respect to their constituent morphemes. Here we present a novel, language-agnostic, framework for MA&D, based on a transition system with two variants, word-based and morpheme-based, and a dedicated transition to mitigate the biases of variable-length morpheme sequences. Our experiments on a Modern Hebrew case study outperform the state of the art, and we show that the morpheme-based MD consistently outperforms our word-based variant. We further illustrate the utility and multilingual coverage of our framework by morphologically analyzing and disambiguating the large set of languages in the UD treebanks.

1 Problem Statement

A decade following the emergence of statistical parsers for English (Charniak, 1996; Bod, 1995), the CoNLL data sets presented a new challenge: the development of data-driven statistical parsers that can be trained to parse any language given an appropriately annotated treebank (Buchholz and Marsi, 2006; Nivre et al., 2007). These data sets facilitated the development of accurate, language-agnostic, dependency parsers (Nivre et al. (2006), McDonald (2006) etc.), but not without shortcomings: they require that input tokens be *morphologically analyzed and disambiguated* in advance.

This latter assumption breaks down in realistic parsing scenarios where the morphological analysis of an input token may consist of multiple syntactic words to participate in the parse tree (Tsarfaty et al., 2010). The *universal dependencies (UD)* initiative aims to remedy this by presenting a harmonized set of treebanks, now 54 and counting, with a unified annotation scheme and multilayered annotation. Specifically, UD data distinguishes the input space-delimited tokens from the (morpho)syntactic words that participate in the parse tree (Nivre et al., 2016).

Efforts towards parsing texts into *universal dependencies* in realistic scenarios thus require language-agnostic infrastructure for automatic *morphological analysis and disambiguation (MA&D)* of data from typologically different languages. MA&D is particularly challenging in *morphologically rich languages (MRLs)*, where space-delimited input tokens may have multiple analyses, only one relevant in context. This *morphological ambiguity* of a token is typically represented as a lattice. The term *Morphological Disambiguation* (MD) refers to selecting a single path through the *morphological analysis* (MA) lattice.

In Semitic languages, MD is particularly challenging (Adler, 2007; Bar-haim et al., 2008; Shacham and Wintner, 2007; Pasha et al., 2014; Habash and Rambow, 2005). To illustrate, Figure 1 shows the MA lattice of the Hebrew phrase 'bclm hneim'[1] (literally: in-shadow-of-them the-pleasant, meaning: in their pleasant shadow). Different paths in the lattice represent different disambiguation decisions. In

[1] We transliterate as in Sima'an et al. (2001).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 337–348, Osaka, Japan, December 11-17 2016.

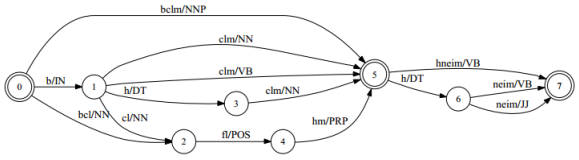

Figure 1: An example of an MA lattice of the phrase "bclm hneim" ("in their pleasant shadow") in transliterated Hebrew. Edges mark syntactic words, and double circles mark white spaces.

the context of 'bclm hneim', the correct path of 'bclm' is 'b-cl-(fl)-hm', "in-shadow-(of)-them". In the context of another sentence, the token 'bclm' may be "Betzelem", the name of a famous organization.

In MRLs, MD is also more subtle than simply segmenting the space-delimited tokens. Many MRLs are fusional, and clitics may be fused into hosts. In Figure 1, in the phrase 'b-cl-(fl)-hm' (literally: in-shadow-(of)-them), the possessive 'fl' (of) is fused into the pronoun 'hm' (them) and remains implicit in the surface form. Such fusion results in an ambiguous number of morphosyntactic nodes that participate in the analysis, impacting downstream applications as syntactic and semantic parsing, translation, etc.

Previous work on MA&D in MRLs, and in Semitic languages in particular (Adler, 2007; Bar-Haim et al., 2005; Shacham and Wintner, 2007; Pasha et al., 2014), relied on language-specific lexica and cannot be executed cross-linguistically. General CRF implementations, such as MarMoT (Müller et al., 2013), that can be applied across languages, assume an unrealistic, gold pre-segmented setting (Bjorkelund et al., 2013). For generic morphological segmentation, Morfessor (Smit et al., 2014) uses a max-likelihood in semi-supervised settings, but it cannot handle the rich labeling of morphological segments.

In this paper we present a general, language-agnostic solution the for the MA&D task. We target *joint* morphological segmentation and tagging, as has been advocated in monolingual cases (Zhang and Clark, 2011; Bar-haim et al., 2008; Adler and Elhadad, 2006; Habash and Rambow, 2005), in *universal* settings. Our technical approach extends the transition-based framework for structured prediction of Zhang and Clark (2011). We define and implement two MD variants: word-based and morpheme-based. We present the best MA&D results to date, and demonstrate that the morpheme-based variant consistently outperforms our word-based one, while providing state-of-the-art results on full-fledge, fine-grained, MD of Hebrew. Furthermore, our MA&D framework is intentionally designed with language independence in mind. Devoid of requiring language-specific resources, we show robust and competitive MA&D performance on MRLs and non-MRLs alike, for circa 50 languages in the most recent release of UD treebanks (Nivre et al., 2016).

2 Challenges and Formal Settings

We propose a data-driven framework for MA&D of MRLs and non-MRLs alike. The MA component implements a function that maps each input sentence to its MA lattice, and the MD component implements a transition-based model that accepts the lattice as input and returns a selected path as output.

Formally, our transition system is a quadruple $S = (C, T, c_s, C_t)$, where C is a set of configurations, T is a set of transitions, c_s is an initialization function, and $C_t \subseteq C$ is a set of terminal configurations. A transition sequence y of length n, $y = c_0, t_1(c_0), ..., t_n(c_{n-1})$, starts with an initial configuration $c_0 = c_s(x)$ for the input sentence x and ends with a terminal configuration $c_n \in C_t$, where $t_i \in T$ and $c_n = t_n(c_{n-1}) \in C_t$. We employ an objective function F where x is the input sentence and $GEN(x)$ is the set of possible transition sequences for x:

$$F(x) = argmax_{y \in GEN(x)} Score(y) \quad (1)$$

To compute $Score(y)$, $y \in GEN(x)$ is mapped to a global feature vector $\Phi(y) \in N_d$, where each feature is a count of occurrences of a pattern defined by a feature function ϕ. The feature vector $\Phi(y)$ is defined via a set of d feature functions $\{\phi_i\}_{i=1}^{d}$. The way Φ is defined effectively determines the quality of the parser, since the feature model captures linguistic information to which the model learns to assign

weights. Given this vector, $Score(y)$ is computed by multiplying $\Phi(y)$ with a weights vector $\vec{\omega} \in R^d$.

$$Score(y) = \Phi(y) \cdot \vec{\omega} = \sum_{c_j \in y} \sum_{i=1}^{d} \omega_i \phi_i(c_j) \quad (2)$$

Following Zhang and Clark (2011), our system learns the weights vector $\vec{\omega} \in R^d$ via the generalized perceptron, using the early-update averaged variant of Collins and Roark (2004). The algorithm iterates through a gold-annotated corpus, each sentence is disambiguated (decoded) with the last known weights, and if the decoded result differs from the gold standard, the weights are updated. As in Zhang and Clark (2011), decoding is based on the beam search algorithm, where a number of possible parse sequences are evaluated concurrently to mitigate irrecoverable prediction errors. At each step, the transition system applies all valid applicable transitions to all candidates. The B highest scoring expanded candidates are maintained and passed on to the next step. Those that don't make the B mark, fall off the beam.

Our MRL setting (cf. Tsarfaty et al. (2010)) poses three technical challenges to this general scheme:
(i) the formal challenge: how should we define a transition system for MA&D?
(ii) the learning challenge: how can we define feature functions that learn morphological phenomena?
(iii) the decoding challenge: how can we effectively compare morpheme sequences of variable length?

3 Our Proposed Solution

Let $x = x_1...x_k$ be an input sentence of k tokens and $L = MA(x_1), ..., MA(x_k)$ be the morphological ambiguity lattice for x, where L is a contiguous series of word-lattices $L_i = MA(x_i)$ connected top to bottom, as illustrated in Figure 1. Each word lattice L_i is a set of sequences of morphemes, and each sequence is a single disambiguated analysis for x_i. We define the *morphosyntactic representation* (MSR) of an arc in the lattice as a tuple $m = (s, e, f, t, g)$ with lattice nodes s and e marking the start and end of a morpheme, a form f, a part-of-speech tag t, and a set g of attribute:value grammatical properties.

Defining Configurations. A configuration for the MD transition system is a quadruple (L, n, i, M) where $L = MA(x)$ is the sentence-lattice, n is a node in L, i is the 0-based index of a word-lattice in L, and M is a set of disambiguated morphemes (i.e., selected arcs). The terminal configuration is defined to be $C_t = \{(L, top(L), tokens(L), M)\}$ for any L, M, where $tokens(L)$ is the number of word-lattices that form L. The initial configuration function c_s concatenates the L_i lattices of the tokens into a single structure $L = MA(x_1) + ... + MA(x_k)$, and sets $n = bottom(L)$, $i = 0$ and $M = \emptyset$.

Defining Transitions. There are two conceivable ways to make morphological disambiguation decisions, in a *word-based* (WB), and in a *morpheme-based* (MB), fashion, in the terminology of Tsarfaty and Goldberg (2008). In WB models (a.k.a *token-level* in the UD terminology), the disambiguation decision determines a complete path of morphemes between token-boundaries. In the lattice, this refers to selecting a path between two token-boundary nodes (double circles). MB disambiguation decisions (also termed *lexical-level*, or *word-level* in UD) occur at any node in the lattice indicating a morpheme boundary, with more than one outgoing arc, choosing a specific arc m among them.

Interim: Word-Based or Morpheme-Based? WB and MB strategies face contradicting, and complementary, challenges. In WB models, disambiguation decisions are complex, and learning how to score them is expected to suffer from data sparseness. MB models, on the other hand, over-generalize in terms of possible morphological combinations, and learning to score combinations may fail to generalize and be prone to over-fitting. On top of that, morpheme sequences are longer than word sequences, which, in a transition-based system, is known to be more error prone. Finally, variable-length sequences introduce length biases which negatively impact performance. Since MA&D is the base for the NLP pipeline, it is critical to settle this debate empirically and establish the basis for downstream tasks.

Parameterizing Transitions. A transition system is required to distinguish between all possible decisions it can make at a given point. At the same time, the model should be able to generalize from seen decisions to unseen ones, and effectively learn to disambiguate open-class words and out-of-vocabulary

items. To satisfy these desiderata, we define a *delexicalization projection* for a pre-defined set of parts-of-speech tags O capturing open-class categories. Simply put, this projection neutralizes the lattice-nodes specific indices, and, for any tag $t \in O$, it further neutralizes the lexical form. Formally:

$$DLEX_O(m) = \begin{cases} (\_, \_, \_, t, g) & \text{if } t \in O \\ (\_, \_, f, t, g) & \text{otherwise} \end{cases} \quad (3)$$

3.1 Word-Based Modeling

The Transition System For word-based (WB) modeling, a single transition morphologically disambiguates whole word-lattices such that the node n of a configuration is always at a word boundary (a node that is a bottom, top, or both, of word-lattices of L). We define the transitions in the WB system as an open set of transitions termed MD_s, specifying the parameter s as a single path:

$$MD_s : (L, n, i, M) \rightarrow (L, q, i+1, M \cup \{m_0, .., m_j\}) \quad (4)$$

Here, $\{m_0, ..., m_j\} \in L$ form a contiguous path of arcs, where m_0 starts at node n, m_j ends at node q (they can be the same arc), and s is the projected paths $s = DLEX_O(m_0), ..., DLEX_O(m_j)$. A terminal configuration will therefore contain the union of contiguous paths of word-lattices in L, together forming a complete morphological disambiguation of the ambiguous tokens of the input sentence.

Learning We define three types of word-lattice properties: o - the surface form of the token itself, a - the $DLEX$-projected lattice (all MSRs projected by the delexicalization function), and p - a chosen disambiguated path, which only exists for previously processed lattices. Using these properties, we define baseline feature templates modeled after POS tagging: unigram, bigram, and trigram combinations of o and a, and p-based features, which predict the next disambiguation decision based on the previous one(s).

3.2 Morpheme-Based Modeling

The Transition System For morpheme-based (MB) modeling, a single transition chooses an outgoing arc of the current node n in the lattice, requiring a disambiguation decision if (and only if) there is more than one outgoing arc. Again we define the transitions as an open set of transitions termed MD_s, now specifying s as a single *arc*:

$$MD_s : (L, n, i, M) \rightarrow (L, q, j, M \cup \{m\}) \quad (5)$$

Here, m is a morpheme $(n, q, f, t, g) \in L$, and $s = DLEX_O(m)$. If node q is at a word boundary, then $j = i + 1$, otherwise $j = i$. For a terminal configuration, each $m \in M$ is an outgoing arc of the end node of another arc in M (with the exception of the first morpheme, starting at $bottom(L)$) forming a contiguous path that disambiguates x.

Learning In the MB model we can access specific information concerning the current node inside the word-lattice. We define the properties f, t and g, corresponding to arcs' form, part-of-speech and morphological attribute:value pairs. We use these properties in various unigram, bigram, and trigram combinations, in parallel with the WB model. As in the WB model we also define the property p as the path in the previously disambiguated word-lattices. We define the property n to be the set of $DLEX$-projected outgoing morphemes of the current node (this parallels the property a of WB models, but at morpheme granularity). Similarly to the WB case, we use unigram, bigram, and trigram combinations of these properties as well.

Decoding Since the number of arcs in lattices' paths for x may vary, so do the number of transitions in our morpheme-based transition system. This violates a basic assumption of standard beam search decoding — that the number of transitions is a deterministic function of the input.

There are two inherent biases in varied-length transition sequences driven by the general perceptron algorithm. The beam search algorithm tests the best candidate after each step for goal fulfillment. A short

sequence may temporarily be the best candidate and fulfill the goal, while longer (and possibly correct) sequences are incomplete and may be lost. On the other hand, long sequences have more features, therefore their score may be arbitrarily inflated. So the score may be higher for longer paths, even though a shorter one may be correct and may fall off the beam.

To address these challenges we introduce a special transition we call *ENDTOKEN (ET)*, that explicitly increments i, instead of implicitly in MD_s. So, in equation (4) we set $j = i$ and apply:

$$ET : (L, n, i, M) \rightarrow (L, n, i + 1, M) \tag{6}$$

ET is required to occur exactly once at the end of the word-lattice, when n is the top of some word-lattice in L. Set aside from other transitions, ET has its own set of features. Other than incrementing i, ET has no effect on configurations, but it does cause a re-ordering of candidates in the beam during decoding, at each token boundary. Note that ET kicks in only for variable length lattices. On same-length lattices, ET is skipped and equation (4) remains as is — the process essentially falls back on the standard, same-length decoding.

An MD transition sequence thus becomes the union of disjoint sets of configurations $y = y_{md} \cup y_{et}$, and changes $Score$ in Equation (2), where $|y_{et}|$ is the # of tokens in L with variable length paths. :

$$\sum_{i=1}^{d} \omega_i^{md} \phi_i^{md}(y_{md}) + \sum_{j=1}^{d} \omega_j^{et} \phi_j^{et}(y_{et}) = \sum_{c_k \in y_{md}} \sum_{i=1}^{d} \omega_i^{md} \phi_i^{md}(c_k) + \sum_{c_l \in y_{et}} \sum_{j=1}^{d'} \omega_j^{et} \phi_j^{et}(c_l) \tag{7}$$

While the number of morphemes, and therefore $|y_{md}|$, can vary, $|y_{et}|$ is deterministic per lattice. Using this anchor, the features of the ET transition provide a counter-balance to the effects of varied-length sequences by scoring fully disambiguated paths of each word-lattice individually, occuring a fixed amount of times for all paths.

ENDTOKEN vs. IDLE transitions Variable-length sequences in beam search also exist in the structured prediction of constituency trees. Zhu et al. (2013) introduced an IDLE transition (also adopted in Honnibal and Johnson (2014) and Zhang et al. (2014)) that, like ET, has no effect on configuration, but unlike ET, occurs only at the end of the parsing sequence, an arbitrary number of times, until all parsing sequences are complete.

While IDLE transitions make sense when applied after a complete hierarchical structure is predicted — where they may learn to rerank candidates based on features that are visible at the top of the structure (the root) — it is futile to use last-seen features that arbitrarily exist at the end of a morphological disambiguation (linear) sequence, to rerank candidates again and again. This is because at the end of the sequence, we can no longer save candidates that were lost earlier on due to length discrepancies .

To mitigate this, one might try to create IDLE padding with global features spanning the entire disambiguated path. Even then, the learned model parameters would not generalize well, since these features will be applied an arbitrary number of times — the maximal length of an occasional word lattice we are at — which has no linguistic significance, and may arbitrarily inflate certain scores.

ET transitions, in contrast, occur right when they are needed — at the boundary of a word-lattice. This position enables the reordering of candidates right after a length discrepancy may have been introduced. Moreover, ET scores are counted against the global score a fixed number of times per lattice, for all, any length, candidates. This enables a fair comparison of all paths per lattice.

4 Empirical Evaluation

We empirically evaluate the proposed models, and investigate their strengths, weaknesses, and bounds. We start with a detailed investigation of MA&D in the Semitic language Modern Hebrew, which is known for its rich morphology and significant ambiguity. We then verify the cross-linguistic coverage of the models on the set of UD treebanks (Nivre et al., 2016), to validate their efficacy.

We implement *yap* (*yet another parser*), a general-purpose transition-based framework for structured prediction, based on beam search and the generalized perceptron. We extend it with the models described herein. We implement the WB, MB variants, ET transitions, and evaluate different feature settings.

(a)

	Word-Based				
	unigram	+bigram	+trigram	+next unigram	+next with bigram
	85.73 (86.72)	86.9 (87.88)	86.7 (87.66)	**92.19 (92.76)**	91.98 (92.59)
+ET	89.09 (89.81)	89.93 (90.71)	90 (90.85)	**93.39 (93.94)**	92.84 (93.46)

(b)

	Morpheme-Based				
	unigram	+bigram	+trigram	+next unigram	+next with bigram
	90.67 (91.41)	91.12 (91.88)	90.09 (91.82)	93.56 (94.16)	**93.89 (94.49)**
+ET	92.68 (93.36)	92.74 (93.55)	92.64 (93.47)	94.27 (94.92)	**94.33 (94.9)**

Table 1: Dev. set results for Word-Based (a) and Morpheme-based (b) MD: F_1 for full morphological disambiguation (form, part of speech, morphological properties). (n parenthesis: F_1 for form and POS only. The +ET lines indicate a variant that employs the ENDTOKEN transition at token boundaries.

We report the F_1 metric comparing the MSRs of *predicted* vs. *gold* lattice arcs, for full morphological disambiguation (segmentation, POS tags, and all morphological properties), and for segmentation and POS tags only. For comparison with previous work, we also report token-level accuracy (while F_1 awards partial success on word-lattices, token-level accuracy requires exact match on a whole path per token).

4.1 The Case for Modern Hebrew

Setup We evaluate MA&D performance on the Modern Hebrew section of the SPMRL 2014 Shared Task (Seddah et al., 2014), which has been derived from the Unified-SD treebank of Tsarfaty (2013). We updated the treebank to provide consistent theories for the treebank annotation and lexicographic resources (Itai and Wintner, 2008), a consistency that we found lacking in the SPMRL 2014 Hebrew section. We use the standard split, and train on the standard train set (5k sentences). Here we provide results and in-depth analysis on *dev* and confirm our findings on *test*.

A pre-condition for the execution of our MD models is an $MA(x)$ function that generates word-lattices for x (§2). We start off with a morphological analyzer that we implemented, called HEBLEX, which relies on the Ben-Gurion Hebrew Lexicon used by Adler and Elhadad (2006). The lexicon contains full analyses for 567,483 words and 102 prefixes. HEBLEX uses the lexicon to determine the various combinations of prefixes and words that form valid tokens. This process is far from trivial due to morphological *fusion*, as some morphemes are implicit (§1).

Although the lexicon is quite large, there are still tokens which are *out-of-vocabulary (OOV)*. OOV tokens may be of two types: it may be that an entire string is out of the lexicon (mostly proper nouns) or that the affixes and the open class items are seen, but their combination has not yet been encountered. We address OOV by assigning proper noun analyses to entire tokens, as well as to all arcs combined with seen affixes. This adds ambiguity to the lattices, but gives the MD the chance to select a correct path.

In our experiments we aim to quantify exactly the effect of lexical coverage of the MA on MD accuracy. To this end, we add an option to *infuse* missing gold analyses into the MA lattices provided by HEBLEX and present two sets of results: once disambiguating lattices with infused gold analyses (ideal MA), and once without infusing gold analyses (realistic MA).

Results Tables (1a), (1b) present our investigation of the WB and MB models on the dev test, respectively, with different feature templates. Our results show that the MB disambiguation consistently outperforms our WB variant, in various feature template settings. Moreover, the ET transition consistently improves performance, with best results for Hebrew at F_1 scores of 94.3 (94.9) for full MD (seg/POS only). The token-level accuracy for our best results are 93.07 (93.9) for full MD (seg/POS).

These results were obtained on *infused* lattices, that include the gold path as one of the alternatives. In order to gauge the effect of incomplete lexical coverage, we disable infusion of the gold analyses into the HEBLEX lattices. We then observe a drop to F_1 scores of 89.62 (92.06) and token accuracy of 87.72(90.85). To set our results in context, we applied our best model in "English-like" settings for tagging, with gold pre-segmented text. F_1 then increases to 96.82 (97.44). That is, in "English-like" settings, our tagging accuracy (97.44) is as high as state-of-the-art English tagging (Manning, 2011).

MarMoT (Müller et al., 2013), a state of the art CRF tagger, obtains F_1 93.38 on full MD on this set.

Next we aim to verify that ET transitions indeed act as intended. We classify a sequence length error as either an overshoot (predicted morphological disambiguation sequence is longer than gold) or undershoot (predicted shorter than gold). Without ET, in the infused setting, 36.8% of sentences have incorrect length and the overshot:undershoot ratio is 6.6:1. Adding ET transitions results in 31.8% length errors, correcting 20.62% of the overshoot errors, resulting in ratio of 4:1. In the un-infused setting, 41.4% of the sentences have incorrect length with a ratio of 6.39:1. Adding ET results in 36% length errors, correcting 20.67% of the overshoot errors, resulting in ratio of 3.7:1.

Hebrew previous results are non-trivial to compare to due to significant changes of the treebank along the way and unavailable code of previous work. The most relevant results to ours are by Adler (2007), who reports state-of-the-art results for Modern Hebrew in realistic (non-infused) setting, with self-reported token accuracy of 90% (93%) on a different evaluation set. For his prediction on our dev set, F_1 evaluation yields 85.74 (87.95), much lower than ours. Segmentation F_1 for Adler is 96.35, while ours is 97.6. We confirm our findings on the test set, for which Adler F_1 yields 82.91 (85.56). Our best model now yields 86.23 (88.85) and 92.96 (93.73) in realistic and infused settings, respectively.

4.2 The Case for Universal Dependencies

Setup We evaluate the cross-linguistic coverage of our MD models on the UD set. We parse 48 UD treebanks from the UD1.3 release (Nivre et al., 2016), training on the *train* set and evaluating on *test*.[2]

We implement a *universal* data-driven morphological analyzer, that can be trained out-of-the-box along with our MD models for any treebank in the CoNLL-U format. We generate a dictionary for each language from its train set by collecting all seen analyses of each token in the training data, where an analysis is composed of MSRs that contain a lemma, POS, and the full set of morphological features. The dictionary maps each token to a set of MSR sequences, which then compose their ambiguous MA lattices. For out-of-vocabulary (OOV) tokens, the MA pre-computes the cardinality of each coarse POS — the number of unique tokens per coarse POS — and consider the top 5 POS as "open-class". For these top 5 POS, the MA computes the 50 highest-frequency MSRs (POS + morph. properties) to be used as the OOV lattice of an OOV token. When applying MA to the training set, we add the OOV lattice to tokens whose known analysis contains an open-class POS. The model thus encounters during training a larger space of states than the observed one, and learns to accurately apply transitions in OOV circumstances at test time.

Results Table 2 reports F_1 accuracy for full MD and seg/POS for UD languages that do not require segmentation. For most large train sets ($> 200k$ tokens), we observe 2-3 points absolute drop from infused (ideal) to uninfused (realistic) setting. This suggests that when large train sets exist, our data-driven MA is a viable economic alternative to costly hand-crafted monolingual lexical resources. To scrutinize our realistic results we also report non-OOV-only F_1 for 5k limited uninfused setting. Here we see that for ~80% of languages, our results are on a par with a state-of-the-art tagger, MarMot, retrained on these data, within 0.035 (or less) points gap. This demonstrates that our disambiguation capacity is on-par with MarMot, where performance gaps come mostly from our OOV strategy (which is intentionally restrained, to allow handling of MRL segmentation that is handled by MarMot). Table 3 shows results for UD MRLs that require segmentation, contrasting results on gold pre-segmented input and un-segmented raw data. For raw data we see a minor, ~0.02, drop in F_1, compared to gold-segmented settings. That is, our model still retains the competitive MA&D performance, in a *single, universal, trainable* model — we attribute this to our *joint* segmentation and tagging strategy, which overcomes error propagation.[3]

[2] We do not present results for 6 languages: cs,kk,es_ancora,en_esl,pt_br,ja_ktc, as some or all form fields are empty.

[3] Shortly before submitting this article, UDPipe (Straka et al., 2016), a tool for tokenization, morphological analysis, tagging and parsing, had been released. As its name suggests, UDPipe is a pipeline implementation packaging together separate tools for different tasks. Our approaches and ultimate goals are rather different. We present *joint* morphological segmentation and tagging, as opposed to a pipeline. Moreover, our framework can be extended into a *single* transition-based system performing all tasks jointly, overcoming overheads and error propagation, as we intend to address next.

Lang.	sents	5k Sent. Trainset			Full Trainset		
	words	inf.	no inf.	MM	inf.	no inf.	MM
bg	8907	0.933	0.914 (0.959)	0.937 (0.964)	0.946	0.926	0.948
	124474	0.969	0.96 (0.983)	0.976 (0.987)	0.979	0.969	0.982
cu	5077	0.901	0.853 (0.911)	0.893 (0.927)	0.908	0.851	0.897
	46025	0.964	0.931 (0.977)	0.959 (0.978)	0.965	0.929	0.962
da	4868	0.932	0.88 (0.944)	0.943 (0.966)	0.934	0.894	0.943
	88979	0.953	0.91 (0.954)	0.961 (0.972)	0.954	0.922	0.961
el	1929	0.912	0.876 (0.918)	0.921 (0.944)	0.914	0.876	0.921
	47449	0.977	0.965 (0.989)	0.979 (0.99)	0.978	0.965	0.979
en	12543	0.904	0.887 (0.927)	0.904 (0.937)	0.927	0.914	0.93
	204586	0.923	0.911 (0.951)	0.923 (0.954)	0.942	0.932	0.945
en_	3650	0.95	0.944 (0.955)	0.96 (0.969)	0.95	0.944	0.96
lines	66374	0.95	0.944 (0.955)	0.96 (0.969)	0.95	0.944	0.96
et	14510	0.866	0.821 (0.911)	0.883 (0.933)	0.913	0.884	0.928
	187814	0.923	0.895 (0.953)	0.933 (0.961)	0.951	0.932	0.959
eu	5396	0.877	0.79 (0.895)	0.874 (0.93)	0.88	0.797	0.877
	72974	0.948	0.904 (0.964)	0.944 (0.971)	0.948	0.908	0.946
fi	12217	0.884	0.756 (0.942)	0.889 (0.961)	0.925	0.864	0.932
	162721	0.915	0.863 (0.96)	0.928 (0.973)	0.95	0.909	0.958
ga	720	0.781	0.743 (0.87)	0.819 (0.909)	0.791	0.747	0.819
	16701	0.896	0.873 (0.946)	0.924 (0.961)	0.9	0.882	0.924
gl	2276	0.971	0.968 (0.985)	0.971 (0.984)	0.971	0.968	0.971
	79329	0.971	0.968 (0.985)	0.971 (0.984)	0.971	0.968	0.971
got	4360	0.874	0.824 (0.873)	0.877 (0.901)	0.871	0.83	0.877
	44722	0.955	0.927 (0.964)	0.961 (0.972)	0.953	0.924	0.961
grc	13185	0.862	0.762 (0.867)	0.844 (0.883)	0.862	0.776	0.859
	196083	0.926	0.853 (0.937)	0.935 (0.946)	0.92	0.861	0.941
grc_	13306	0.829	0.719 (0.89)	0.774 (0.887)	0.909	0.853	0.9
proiel	166061	0.911	0.827 (0.972)	0.893 (0.966)	0.973	0.939	0.971
hi	13304	0.82	0.799 (0.823)	0.867 (0.891)	0.846	0.832	0.891
	281057	0.944	0.934 (0.955)	0.947 (0.964)	0.953	0.948	0.963
hr	3557	0.86	0.812 (0.877)	0.87 (0.913)	0.86	0.813	0.87
	78817	0.951	0.936 (0.974)	0.954 (0.973)	0.951	0.933	0.954
hu	1433	0.763	0.701 (0.831)	0.751 (0.862)	0.758	0.697	0.751
	33016	0.92	0.895 (0.951)	0.945 (0.973)	0.915	0.896	0.945
id	4477	0.932	0.926 (0.934)	0.937 (0.945)	0.932	0.926	0.937
	97531	0.932	0.926 (0.934)	0.937 (0.945)	0.932	0.926	0.937

Lang.	sents	5k Sent. Trainset			Full Trainset		
	words	inf.	no inf.	MM	inf.	no inf.	MM
la	2660	0.797	0.696 (0.814)	0.797 (0.876)	0.787	0.709	0.797
	37819	0.904	0.847 (0.927)	0.935 (0.967)	0.897	0.854	0.935
la_	16258	0.916	0.895 (0.922)	0.922 (0.938)	0.935	0.914	0.945
ittb	276941	0.978	0.965 (0.985)	0.979 (0.986)	0.986	0.977	0.988
la_	11986	0.857	0.773 (0.859)	0.841 (0.886)	0.892	0.843	0.892
proiel	132376	0.949	0.896 (0.959)	0.943 (0.967)	0.969	0.943	0.97
lv	673	0.797	0.745 (0.927)	0.817 (0.931)	0.801	0.739	0.817
	12629	0.876	0.841 (0.971)	0.898 (0.971)	0.88	0.835	0.898
nl	13000	0.836	0.797 (0.889)	0.823 (0.911)	0.88	0.863	0.872
	197134	0.866	0.841 (0.923)	0.861 (0.928)	0.902	0.891	0.897
nl_	6641	0.939	0.934 (0.967)	0.95 (0.971)	0.943	0.94	0.952
lassysmall	88929	0.954	0.95 (0.973)	0.961 (0.974)	0.956	0.953	0.963
no	15696	0.926	0.887 (0.952)	0.93 (0.963)	0.947	0.917	0.952
	244776	0.959	0.944 (0.97)	0.964 (0.976)	0.971	0.958	0.975
pl	6800	0.858	0.772 (0.882)	0.854 (0.916)	0.87	0.778	0.866
	69499	0.955	0.915 (0.976)	0.961 (0.978)	0.961	0.922	0.967
pt	8800	0.913	0.881 (0.919)	0.916 (0.938)	0.927	0.906	0.93
	214812	0.96	0.947 (0.965)	0.967 (0.977)	0.967	0.958	0.972
ro	4759	0.915	0.894 (0.917)	0.927 (0.939)	0.916	0.895	0.927
	108618	0.962	0.946 (0.967)	0.962 (0.973)	0.961	0.947	0.962
ru	4029	0.869	0.791 (0.895)	0.861 (0.925)	0.872	0.791	0.861
	79772	0.957	0.925 (0.976)	0.956 (0.979)	0.956	0.925	0.956
ru_	46750	0.896	0.818 (0.919)	0.89 (0.944)	n/a	n/a	n/a
syntagrus	815485	0.965	0.935 (0.977)	0.968 (0.98)	n/a	n/a	n/a
sl	6471	0.874	0.791 (0.876)	0.884 (0.927)	0.882	0.804	0.892
	112334	0.954	0.923 (0.967)	0.963 (0.976)	0.958	0.927	0.967
sl_	2472	0.83	0.796 (0.86)	0.857 (0.902)	0.832	0.793	0.857
sst	23575	0.896	0.88 (0.92)	0.919 (0.947)	0.895	0.878	0.919
sv	4303	0.928	0.921 (0.951)	0.946 (0.966)	0.931	0.921	0.946
	66645	0.952	0.947 (0.965)	0.966 (0.975)	0.954	0.947	0.966
sv_	3650	0.955	0.952 (0.964)	0.957 (0.966)	0.955	0.952	0.957
lines	63949	0.955	0.952 (0.964)	0.957 (0.966)	0.955	0.952	0.957
zh	3997	0.903	0.889 (0.918)	0.913 (0.932)	0.903	0.889	0.913
	98608	0.914	0.903 (0.933)	0.923 (0.943)	0.914	0.903	0.923

Table 2: MA&D for unsegmented languages : F_1 scores of the languages in UD that do not require morphological segmentation, the upper line indicates full MD, the lower line indicates segmentation and POS only. MM columns are results for MarMoT. Results in parentheses are F_1 scores for non-OOV-only tokens.

Lang.	sents	Gold Segmented 5k Train. Set (non-OOV accuracy)					Gold Segmented Full Trainset					Un-Segmented Full Trainset			
	words	inf.	+ET	no inf.	+ET	MM	inf.	+ET	no inf.	+ET	MM	inf.	+ET	no inf.	+ET
ar	6174	0.887	0.887	0.853	0.853 (0.87)	0.903 (0.924)	0.892	0.892	0.86	0.86	0.907	0.867	0.871	0.8	0.799
	225853	0.956	0.956	0.948	0.948 (0.956)	0.956 (0.972)	0.959	0.959	0.951	0.951	0.959	0.929	0.933	0.882	0.882
ca	13123	0.953	0.953	0.919	0.919 (0.958)	0.957 (0.965)	0.963	0.963	0.939	0.939	0.968	0.961	0.961	0.938	0.937
	429157	0.969	0.969	0.936	0.936 (0.972)	0.973 (0.977)	0.977	0.977	0.954	0.954	0.98	0.975	0.975	0.952	0.951
cs_	23478	0.865	0.857	0.758	0.758 (0.862)	0.86 (0.921)	0.901	0.901	0.831	0.831	0.904	0.897	0.899	0.827	0.827
cac	472608	0.973	0.969	0.93	0.928 (0.984)	0.975 (0.988)	0.983	0.983	0.961	0.961	0.987	0.983	0.983	0.961	0.961
cs_	860	0.845	0.844	0.812	0.801 (0.871)	0.887 (0.922)	0.848	0.847	0.816	0.812	0.887	0.832	0.822	0.804	0.8
cltt	26234	0.956	0.959	0.942	0.938 (0.988)	0.98 (0.991)	0.958	0.957	0.94	0.942	0.98	0.953	0.951	0.937	0.938
de	14118	0.928	0.928	0.921	0.921 (0.936)	0.927 (0.941)	0.929	0.929	0.921	0.921	0.927	0.93	0.928	0.921	0.92
	269626	0.928	0.928	0.921	0.921 (0.936)	0.927 (0.941)	0.929	0.929	0.921	0.921	0.927	0.93	0.928	0.921	0.92
es	14187	0.929	0.928	0.888	0.888 (0.943)	0.927 (0.956)	0.939	0.939	0.908	0.908	0.936	0.93	0.933	0.9	0.903
	382436	0.947	0.947	0.931	0.931 (0.959)	0.945 (0.967)	0.955	0.955	0.943	0.943	0.953	0.948	0.951	0.935	0.938
fa	4798	0.953	0.961	0.958	0.958 (0.972)	0.963 (0.978)	0.954	0.953	0.956	0.957	0.963	0.957	0.956	0.947	0.949
	121020	0.96	0.968	0.964	0.964 (0.974)	0.969 (0.98)	0.96	0.959	0.962	0.963	0.969	0.962	0.962	0.954	0.955
fi_	14981	0.876	0.88	0.794	0.793 (0.921)	0.858 (0.944)	0.856	0.847	0.811	0.809	0.915	0.916	0.929	0.814	0.856
ftb	127602	0.914	0.917	0.856	0.855 (0.953)	0.904 (0.957)	0.883	0.869	0.843	0.844	0.943	0.939	0.95	0.849	0.899
fr	14554	0.931	0.93	0.921	0.918 (0.935)	0.939 (0.954)	0.943	0.951	0.925	0.925	0.949	0.944	0.942	0.92	0.921
	356216	0.949	0.947	0.941	0.939 (0.952)	0.955 (0.967)	0.959	0.965	0.942	0.942	0.964	0.958	0.957	0.937	0.938
he	5241	0.934	0.933	0.888	0.888 (0.907)	0.921 (0.953)	0.934	0.93	0.891	0.886	0.922	0.914	0.917	0.724	0.724
	135496	0.968	0.968	0.937	0.938 (0.947)	0.955 (0.974)	0.967	0.966	0.939	0.937	0.957	0.945	0.947	0.768	0.769
it	11699	0.945	0.946	0.91	0.911 (0.953)	0.943 (0.962)	0.96	0.958	0.945	0.945	0.97	0.953	0.957	0.935	0.937
	249330	0.959	0.96	0.924	0.925 (0.961)	0.958 (0.972)	0.97	0.967	0.956	0.955	0.978	0.961	0.965	0.946	0.947
ta	400	0.761	0.793	0.761	0.732 (0.912)	0.82 (0.929)	0.794	0.797	0.757	0.756	0.82	0.72	0.722	0.659	0.664
	6329	0.835	0.859	0.825	0.813 (0.927)	0.877 (0.938)	0.856	0.861	0.827	0.821	0.877	0.765	0.772	0.714	0.717
tr	3947	0.781	0.873	0.765	0.806 (0.935)	0.855 (0.933)	0.794	0.787	0.768	0.805	0.855	0.84	0.781	0.742	0.78
	40609	0.884	0.938	0.87	0.897 (0.968)	0.934 (0.964)	0.893	0.885	0.879	0.899	0.934	0.907	0.87	0.846	0.871

Table 3: MA&D in segmented languages: F_1 scores of the languages in UD which require morphological segmentation. The upper line indicate full MD, the lower line indicates segmentation and POS only. The left hand side shows results for GOLD segmentation, the right hand side for input lattices. MM columns are results for MarMoT. Results in parentheses are F_1 scores for non-OOV-only tokens.

5 Conclusion

We present an MD transition-based system that can effectively cope with extreme morphological ambiguities in MRLs. To the best of our knowledge, this is the first joint framework for MRL segmentation and tagging in a transition-based setup. Moreover, we present the best MA&D results for Modern Hebrew to date, and the first ever set of MA&D results for the most recent release of UD treebanks (UD1.3).[4] Our system provides a first tier for dependency parsing in real-world scenarios, dispensing with the need of external pre-processing. Furthermore, this transition-based model can be extended into a *joint* model for *complete* morphological and syntactic analysis, as has been previously advanced in phrase-based parsing (Tsarfaty, 2006; Goldberg and Tsarfaty, 2008; Cohen and Smith, 2007; Green and Manning, 2010).

[4]Upon publication we will make our source code, executables and trained models available at `https://github.com/habeanf/yap`.

References

Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based hmm for Hebrew morphological disambiguation. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *ACL*. The Association for Computer Linguistics.

Meni Adler. 2007. *Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach*. Ph.D. thesis, Ben-Gurion University of the Negev, Beer-Sheva, Israel.

Roy Bar-Haim, Khalil Simaan, and Yoad Winter. 2005. Part-of-speech tagging for Hebrew and other semitic languages. *Master's thesis, Technion, Haifa, Israel*.

Roy Bar-haim, Khalil Sima'an, and Yoad Winter. 2008. Part-of-speech tagging of Modern Hebrew text. *Natural Language Engineering*, 14(2):223–251.

Anders Bjorkelund, Ozlem Cetinoglu, Richard 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*, pages 135–145, Seattle, Washington, USA, October. Association for Computational Linguistics.

Rens Bod. 1995. *Enriching Linguistics With Statistics*. Ph.D. thesis, University of Amsterdam.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*, pages 149–164.

Eugene Charniak. 1996. Tree-Bank Grammars. In *AAAI/IAAI, Vol. 2*, pages 1031–1036.

S. B. Cohen and N. A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of EMNLP*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoav Goldberg and Reut Tsarfaty. 2008. A single framework for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL*.

Spence Green and Christopher D. Manning. 2010. Better Arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 394–402, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 573–580, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association of Computational Linguistics – Volume 2, Issue 1*, pages 131–142.

Alon Itai and Shuly Wintner. 2008. Language resources for Hebrew. *Language Resources and Evaluation*, 42(1):75–98, March.

Christopher D. Manning, 2011. *Computational Linguistics and Intelligent Text Processing: 12th International Conference, CICLing 2011, Tokyo, Japan, February 20-26, 2011. Proceedings, Part I*, chapter Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?, pages 171–189. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ryan McDonald. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvanya.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order crfs for morphological tagging. *In Proceedings of EMNLP*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, pages 2216–2219.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Çağrı Çöltekin, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gokirmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Normunds Grūzītis, Bruno Guillaume, Jan Hajič, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Simon Krek, Veronika Laippala, Lucia Lam, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Loganathan Ramasamy, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulīte, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jing Xian Wang, Jonathan North Washington, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2016. Universal dependencies 1.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. pages 103–109.

Danny Shacham and Shuly Wintner. 2007. Morphological disambiguation of Hebrew: A case study in classifier combination. In *Proceedings of ACL*.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and N. Nativ. 2001. Building a tree-bank of Modern Hebrew text. *Traitment Automatique des Langues*, 42(2).

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24. Association for Computational Linguistics.

Milan Straka, Jan Hajic, and Jana Straková. 2016. Udpipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Reut Tsarfaty and Yoav Goldberg. 2008. Word-based or morpheme-based? annotation strategies for modern Hebrew clitics. In *Proceedings of LREC*.

Reut Tsarfaty, Djame Seddah, Yoav Goldberg, Sandra Kuebler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing for morphologically rich language (spmrl): What, how and whither. In *Proceedings of the first workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL) at NA-ACL*.

Reut Tsarfaty. 2006. Integrated morphological and syntactic disambiguation for modern Hebrew. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, COLING ACL '06, pages 49–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Reut Tsarfaty. 2013. A unified morphosyntactic scheme for stanford dependencies. In *Proceedings of ACL*.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Comput. Linguist.*, 37(1):105–151, March.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1326–1336, Baltimore, Maryland, June. Association for Computational Linguistics.

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 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 434–443. The Association for Computer Linguistics.