Transition-Based Morphological Disambiguation

Amir More
School of Computer Science
Interdisciplinary Center (IDC) Herzliya, Israel
habeanf@gmail.com

Reut Tsarfaty
Mathematics and Computer Science
Open University of Israel
reutts@openu.ac.il

Abstract

In Morphologically Rich Languages (MRLs), sentences are composed of ambiguous space-delimited tokens that ought to be disambiguated with respect to their constituent morphemes. Previous work on Morphological Disambiguation (MD) of MRLs has had variable success, with Semitic languages having sub-par results for downstream applications. Here we propose novel MD transition-based systems, both word-based and morpheme-based, and tackle the challenge introduced by the variable length of hypothesized morpheme sequences. Our experiments show that transition-based morpheme-based MD consistently outperforms the word-based variant, while providing new state of the art results on Hebrew MD.

1 Problem Statement

In MRLs, each input space-delimited token may have multiple different morphological analyses, where only one is relevant in context. This *morphological ambiguity* of a token is represented by a word-lattice describing the various morpheme sequences that may combine to form it, with only one sequence (or, path) that is suited in the context of the sentence. *Morphological Disambiguation* (MD) in Semitic languages is particularly difficult, due significant morphological richness. (Adler, 2007; Bar-haim et al., 2008; Shacham and Wintner, 2007; Pasha et al., 2014; Habash and Rambow, 2005). Figure 1, for example, shows the morphologically ambiguous lattices representing the Modern Hebrew phrase בצלם הנעים (literally: in-shadow-of-them the-pleasant, meaning: in their pleasant shadow). In the context of בצלם הנעים, the correct path of the first token is ב־צל־(של)־הם - "in-shadow-(of)-them". Note that in the context of another sentence, the token בצלם may be "Betzelem" — the name of a famous organization.

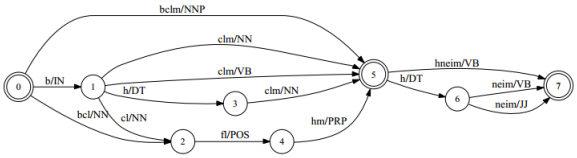

Figure 1: An example of morphological ambiguity lattice in transliterated Hebrew. Edges mark morphemes, and double circles mark word boundaries. Note the variable length of paths.

Morphological disambiguation is more subtle than choosing a segmentation of agglutinated morphemes. Semitic MRLs are fusional; morphemes may be fused to the host, as can be seen in ב־צל־(של)־הם, where the possessive של is fused into the pronoun הם. This results in an ambiguity of the number of morphemes, impacting downstream applications. Downstream applications such as syntactic parsing require a correct disambiguation in order to correctly predict their outputs.

In this paper we present new, novel systems for morphological disambiguation in MRLs. Our approach to disambiguation extends the transition-based framework for structure prediction of Zhang and Clark (2011a). We define two different systems, word-based and morpheme-based, and show that morpheme-based variant consistently outperforms the word-based one, while improving state-of-the-art results on full-fledge, fine-grained, morphological disambiguation of Hebrew.

Our motivation and ultimate goal is *joint* morphological and syntactic disambiguation, an approach advocated for constituency parsing by Tsarfaty (2006; Goldberg and Tsarfaty (2008; Cohen and Smith (2007; Green and Manning (2010), in *dependency-based* frameworks. To this end, we hereby offer an MD transition system compatible with and complementing the various state-of-

the-art frameworks available for dependency parsing nowadays (Nivre and Hall, 2005; Bohnet and Nivre, 2012a; Zhang and Nivre, 2011; Zhang and Clark, 2011b).

2 Formal Preliminaries

Transition-Based Parsing. A transition system is an abstract machine consisting of a set of configurations and transitions between configurations (Kübler et al., 2009). Formally, a 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 starts with an initial configuration given by c_s for any sentence x, and ends with a configuration in C_t. In a data-driven approach, a parametric model is defined as a means of predicting which transition to apply at each step.

The Objective Function. Zhang and Clark (2011a) describe a statistical framework for structure prediction based on beam search decoding complemented with supervised statistical learning based on the generalized perceptron. The framework defines the following objective function, where x is an input to be decoded and $GEN(x)$ denotes the set of possible transition sequences for input 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 set of d feature functions $\{\phi_i\}$. Given this vector, $Score(y)$ is computed by multiplying $\Phi(y)$ with a weights vector $\vec{\omega} \in R^d$. The weights vector defines the model parameters.

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

Learning The weights of the feature vector $\vec{\omega} \in R^d$ are learned by a supervised learning algorithm called the generalized perceptron[1], using a set of sentences paired with corresponding correct analyses (a corpus), the algorithm iterates through the corpus parsing the sentences one by one; each sentence is first parsed (decoded) with the last known weights, and the result is compared to the manually parsed tree (gold standard). If the parsed result differs from the gold standard, the weights are updated. This process is usually stopped when overfitting begins to occur. The way Φ is defined can determine the performance of the parser, since the feature model captures the linguistic information used to compute $Score$.

Decoding Decoding in our framework is based on the beam search algorithm. In beam search, a number of possible parsing sequences are evaluated concurrently to mitigate irrecoverable prediction errors. The beam search algorithm maintains a list of candidates. At each step, each candidate is evaluated by the transition system, which reports the valid applicable transitions, passed on to the prediction model and scored. The B highest scoring transitions for all candidates are maintained in the candidate list and passed on to the next step.

3 Our Proposed Systems

3.1 Design Principles

The Transition System There are two conceivable ways to make morphological disambiguation decisions, in a word-based, and in a morpheme-based, fashion (Tsarfaty and Goldberg, 2008). In word-based models, the disambiguation decision determines a complete path of morphemes between word-boundaries. In the lattice, this refers to selecting a path between two word-boundary nodes. In morpheme-based disambiguation, decisions occur at the morpheme level. In the lattice, at any node with more than one outgoing edge, a transition is the decision of choosing a specific morpheme given a set of possible outgoing edges at a given node in the lattice.

Parametrized Transitions We define a morpheme $m = (s, e, f, t, g)$ in the lattice as a 5-tuple with start and end nodes s and e respectively, form f, part-of-speech tag t, and set g of morphological properties. A transition system is required to distinguish between all possible decisions it can make; meaning that every two paths, or two nodes, must be distinguishable from one another. At the same time, the model should be able to generalize from seen decision sequences to unseen ones, and efficiently learn to handle open-class elements and out-of-vocabulary items.

To address both matters, we define the *parametrized projection* of a morpheme. If the

[1] We use the early-update averaged perceptron variant (Collins and Roark, 2004).

POS tag t belongs to an open-class category, the morpheme is parameterized as $(\_, t, g)$. if t is a close class category, the morpheme is lexicalized as (f, t, g). Applying this projection to all morphemes allows the transition system to distinguish between any two morphemes (or paths of morphemes) with the same starting node, while providing an opportunity to generalize the in-context behavior of forms with different morphemes.

3.2 Word-Based Modeling

The Transition System The configuration of a word-based transition system is an ordered pair (b, M): b - a buffer of morphologically ambiguous tokens, represented by word-lattices, and a set of morphemes M. The initial configuration c_s function sets the buffer to all tokens in the sentence, and $M = \emptyset$. The transition system is a set of parametrized transitions $MD_x : ([b|l], M) \rightarrow (b, M \cup F_p(l, x))$, where $F_p(l, x)$ is a sequence s of morphemes in word-lattice l such that $s = m_0 m_1 ... m_i \in l$, $x = p(m_0)p(m_1)...p(m_i)$, s is a path in word-lattice l, and p is the parametrized morpheme projection function. The resulting set of sequences for a terminal configuration, $s_0, s_1, ... \in M$, together form a contiguous path through all word-lattices in the input sentence.

Learning We define three types of word-lattice properties: t - the token itself, a - the projected lattice (all morphemes projected by the parameter function), and i - 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 t and a, and i based features, which predict the next disambiguation decision based on the previous one.

3.3 Morpheme-Based Modeling

The Transition System The configuration of a morpheme-based transition system is a triple (b, M, n), extending the word-based system, where b and M are the same but with n indicating a node in a word-lattice. The initial configuration c_s function sets the buffer to all tokens in the sentence, $n = bottom(l_0)$, where l_0 is the head of b, and $M = \emptyset$. The transition system is an open class of transition $MD_x : ([b, l], M, n) \rightarrow (b', M \cup m, o)$ where $x = p(m)$, m is a morpheme $(n, o, f, t, g) \in l$, and p is the parameterized morpheme projection function. If o is the top of the word-lattice l at the head of b, then $b' = b$, otherwise $b' = [b|l]$.

Learning When making at morpheme-based transition we can access more relevant and specific information. We define the morpheme properties m, p and f, corresponding to morphemes' form, part-of-speech and morphological properties. We use these properties in various unigram, bigram, and trigram combinations, in parallel with the word-based model. As in word-based disambiguation, we use the property i as a a path of previously a disambiguated word-lattices. Similarly, we define the property n to be the set of projected morphemes of the current node, like the property a of word-lattices at morpheme granularity. We use unigram, bigram, and trigram combinations of these two properties as well.[2]

Decoding Since the number of morphemes in lattices' paths may vary, so do the number of transitions of a morpheme-based transition systems. 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. First, 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. Secondly, and more importantly, long sequences have more features, therefore their score may be arbitrarily higher than shorter ones, even though the shorter ones may be correct.

To solve these problems we introduce a special transition, POP, occurring at word-lattice boundaries. Set aside from other transitions, POP has its own set of features. POP has no effect on configurations, but results in a re-ordering of candidates in the beam during parsing, for equal-length candidates in terms of the number of tokens. While the number of morphemes, and therefore transitions, can vary wildly, the number of POPs is equal to the number of word-lattices. Using this anchor, the features of the POP transition provide a counter-balance to the effects of varied-length sequences by scoring fully disambiguated paths of each word-lattice individually.

[2]The supplementary material contains our complete list of features templates, for all models.

	Word-Based					
	unigram	+bigram	+trigram	+next unigram	+next with bigram	+ next with trigram
	86.43 (87.73)	86.79 (87.62)	87.51 (87.07)	91.88 (**91.53**)	**91.99** (91.42)	91.98 (91.41)
+POP	88.67 (88.52)	89.41 (89.3)	89.01 (89.85)	**92.45 (92.64)**	91.98 (91.64)	91.05 (91.64)

	Morpheme-Based					
	unigram	+bigram	+trigram	+next unigram	+next with bigram	+next with trigram
	90.20 (91.19)	91.27 (91.96)	91.00 (91.74)	92.19 (**92.85**)	**92.21** (92.79)	91.92 (92.52)
+POP	91.95 (92.42)	92.57 (92.94)	92.36 (92.65)	**92.99 (93.37)**	92.49 (92.77)	92.28 (92.50)

Table 1: (a)Word-Based and (b)Morpheme-based MD: F_1 for full morphological disambiguation (form, part of speech, morphological properties). In parenthesis: form and POS only

4 Experiments

Data We use the Hebrew Treebank version from the SPMRL 2013 Shared Task (Seddah et al., 2013), derived from the Hebrew Treebank V2 (Sima'an et al., 2001; Guthmann et al., 2009), with the standard split. The gold training analyses of the original Hebrew Treebank uses morphological analysis theories different from those used to generate the morphological lattices. Therefore, we train our structure perceptron using morphological analyses previously predicted by (Adler, 2007) according to the same theory. We discard 1011 sentences in the training data due to bugs and errors in morphological lattices. To evaluate our results, we adapt the dev gold data to the relevant morphological analysis theory.

The Input: Morphological Analysis The input sentences are converted into morphological lattices using the wide-coverage morphological analyzer of MILA (Itai and Wintner, 2008), with a simple heuristic for dealing with unknown tokens. We apply a few transformations to the lattices to fix various bugs in the morphological analysis, and infuse correct paths from the disambiguation so that the correct path exists in the input.

Parser We implement yap[3], a general-purpose parser based on the generalized perceptron and beam search, and extends it with the transition-based MD systems described herein. We implement the word-based and morpheme-based falvours, with and without POP transitions, and compare different feature templates settings.

Metrics We report the F_1 measure for full morphological disambiguation (segmentation, POS tags, and all morphological properties) as our primary measure. We also report the F_1 of segmentation and POS, for comparison with previous work.

Results Tables 1 (a),(b) show the performance of our word-based and morpheme-based models, respectively. Our results show that morpheme-based disambiguation consistently outperforms word-based disambiguation, in various settings. Also, the POP transition consistently improves performance, with best results yielding state-of-the-art for Modern Hebrew — F_1 scores of 92.99 and 93.37 for full MD and form/POS only, respectively. For comparability, using our evaluation and measures for (Adler, 2007)'s HMM-based MD on our data, we observe F_1 of 85.74 and 87.95 for full and form/POS disambiguation, respectively.[4] Note that it is unrealistic to expect the correct disambiguation to always exist in the morphological lattice, since any morphological analyzer will encounter unseen tokens, resulting in incomplete lattices. To evaluate this scenario, we disable infusion of the correct disambiguations into the ambiguous lattices for the dev data set. In this condition, we observe a drop to F_1 scores of 87.43 and 90.42 for full MD and form/POS, respectively.

5 Conclusion

We introduce a general framework for transition-based MD, with both word-based and morpheme-based variants, along with an effective solution to the biases of variable-length analyses. Our best model provides the best results on full-fledge morphological analysis and MD of Hebrew so far. The framework is compatible with transition-based dependency parsing, making it amenable to the implementation of joint, dependency-based mor-

[3] yap is **y**et **a**nother **p**arser `https://github.com/habeanf/yap`

[4] We introduce an additional bulk of previous MD studies on Semitic MD in the supplementary material.

phosyntactic parsers — which we address next.

References

Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based hmm for Hebrew morphological disambiguation. In *Proceedings of COLING-ACL*.

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.

Bernd Bohnet and Joakim Nivre. 2012a. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1455–1465, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bernd Bohnet and Joakim Nivre. 2012b. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1455–1465, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajic. 2013. Joint morphological and syntactic analysis for richly inflected languages. *TACL*, 1:415–428.

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.

Noemie Guthmann, Adi Milea Milea, and Yoad Winter. 2009. Automatic annotation of morpho-syntactic dependencies in a modern Hebrew treebank. Proceedings of TLT.

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.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Number 2 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Joakim Nivre and Johan Hall. 2005. Maltparser: A language-independent system for data-driven dependency parsing. In *In Proc. of the Fourth Workshop on Treebanks and Linguistic Theories*, pages 13–95.

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).

Djame Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, D. Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Villemonte Eric de la Clergerie. 2013. Proceedings of the fourth workshop on statistical parsing of morphologically-rich languages. pages 146–182. Association for Computational Linguistics.

Danny Shacham and Shuly Winter. 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.

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

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.

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

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

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 188–193, Stroudsburg, PA, USA. Association for Computational Linguistics.

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.

Transition-Based Morphological Disambiguation
Supplementary Material

Appendix 1: Feature Properties

Morpheme-based properties, addressed by M_k, where k is the k-th disambiguated morpheme.
m - morpheme's form
f - morpheme's morphological properties
p - morpheme's part-of-speech tag
n - the set of projected ambiguous morphemes of the current node
Lattice-based properties, addressed by L_k, where k is the k-th word-lattice[5] in the lattice buffer b.
t - lattice's token
a - lattice's set of all possible paths/spellouts, with projected morphemes
i - sequence of a lattice's spellout with projected morphemes - exists only for disambiguated lattices

Appendix 2: Word-Based Features

Lattice Unigram:L_0a, L_0t
Lattice Bigram: $L_0tL_{-1}t, L_0tL_{-1}a, L_0aL_{-1}t, L_0aL_{-1}a$
Lattice Trigram:
$L_0tL_1tL_{-1}t, L_0tL_1aL_{-1}t, L_0aL_1tL_{-1}t, L_0aL_1aL_{-1}t,$
$L_0tL_1tL_{-1}a, L_0tL_1aL_{-1}a, L_0aL_1tL_{-1}a, L_0aL_1aL_{-1}a$
Previously Disambiguated Lattice Unigram: $L_{-1}i$
Previously Disambiguated Lattice Bigram:
$L_{-1}iL_0a, L_{-1}iL_0t, L_{-1}iaL_0a, L_{-1}iaL_0t, L_{-1}itL_0a, L_{-1}itL_0t$
Previously Disambiguated Lattice Trigram:
$L_{-2}iL_{-1}iL_0a, L_{-2}iL_{-1}iL_0t, L_{-2}iL_{-1}iaL_0a, L_{-2}iL_{-1}iaL_0t, L_{-2}iL_{-1}itL_0a, L_{-2}iL_{-1}itL_0t$
$L_{-2}iaL_{-1}iL_0a, L_{-2}iaL_{-1}iL_0t, L_{-2}iaL_{-1}iaL_0a, L_{-2}iaL_{-1}iaL_0t, L_{-2}iaL_{-1}itL_0a, L_{-2}iaL_{-1}itL_0t$
$L_{-2}itL_{-1}iL_0a, L_{-2}itL_{-1}iL_0t, L_{-2}itL_{-1}iaL_0a, L_{-2}itL_{-1}iaL_0t, L_{-2}itL_{-1}itL_0a, L_{-2}itL_{-1}itL_0t$
POP[6]:$L_{-1}i, L_{-1}it, L_{-1}ia$

Appendix 3: Morpheme-Based Features

Disambiguated Morphemes Unigram: $M_0m, M_0p, M_0mp, M_0f, M_0mf, M_0pf, M_0mpf$
Disambiguated Morphemes Bigram:
$M_0mM_1m, M_0mfM_1m, M_0mpM_1m, M_0mpfM_1m$
$M_0mM_1mp, M_0mfM_1mp, M_0mpM_1mp, M_0mpfM_1mp$
$M_0pM_1p, M_0pfM_1pf, M_0pfM_1p, M_0fM_1p, M_0fM_1p$
Disambiguated Morphemes Trigram:
$M_0mM_1mM_2m, M_0pM_1pM_2p, M_0mpM_1mpM_2mp$
$M_0mpfM_1mpM_2mp, M_0mpfM_1mpfM_2mpf, M_0fM_1pM_2p$
Next Ambiguous Morphemes Unigram: L_0n, L_0na, L_0nt
Next Ambiguous Morphemes with Previous Lattice Disambiguation Bigram:
$L_{-1}iL_0n, L_{-1}iL_0na, L_{-1}iL_0nt, L_{-1}iaL_0n, L_{-1}iaL_0na, L_{-1}iaL_0nt, L_{-1}itL_0n, L_{-1}itL_0na, L_{-1}itL_0nt$
Next Ambiguous Morphemes with Previous Lattice Disambiguation Trigram:
$L_{-2}iL_{-1}iL_0n, L_{-2}iL_{-1}iL_0na, L_{-2}iL_{-1}iL_0nt, L_{-2}iL_{-1}iaL_0n$
$L_{-2}iL_{-1}iaL_0na, L_{-2}iL_{-1}iaL_0nt, L_{-2}iL_{-1}itL_0n, L_{-2}iL_{-1}itL_0na$
$L_{-2}iL_{-1}itL_0nt, L_{-2}iaL_{-1}iL_0n, L_{-2}iaL_{-1}iL_0na, L_{-2}iaL_{-1}iL_0nt$
$L_{-2}iaL_{-1}iaL_0n, L_{-2}iaL_{-1}iaL_0na, L_{-2}iaL_{-1}iaL_0nt, L_{-2}iaL_{-1}itL_0n$
$L_{-2}iaL_{-1}itL_0na, L_{-2}iaL_{-1}itL_0nt, L_{-2}itL_{-1}iL_0n, L_{-2}itL_{-1}iL_0na$
$L_{-2}itL_{-1}iL_0nt, L_{-2}itL_{-1}iaL_0n, L_{-2}itL_{-1}iaL_0na, L_{-2}itL_{-1}iaL_0nt$

[5]Negative values of k address previous word-lattices, relative to the head of the lattice buffer b, so L_{-1} addresses the word-lattice occurring in the input sentence, before the word-lattice at the head of b.

[6]POP feature templates are restricted to fire when predicting a POP transition, all other feature templates are not applied.

$L_{-2}itL_{-1}itL_0n, L_{-2}itL_{-1}itL_0na, L_{-2}itL_{-1}itL_0nt$
POP[7]:$L_{-1}i, L_{-1}it, L_{-1}ia$

Related Work and Future Work

Previous work tackled matters of MD in MRLs and in Semitic languages in particular, both in isolation and in the context of joint processing. Both Adler ((Adler and Elhadad, 2006), (Adler, 2007)) and . (Bar-Haim et al., 2005) used generative models based on HMM. Adler used HMMs for morphological disambiguation in semi supervised settings, where morphological properties are combined into a so-called dense morpheme model allowing for learning both segmentation and tagging together. (Bar-Haim et al., 2005) uses HMM for separate segmentation and POS tagging only (without disambiguation morphological properties). (Shacham and Winter, 2007) use a hierarchy of classifiers to obtain fine-grained MD in Hebrew. For Modern Standard Arabic (MSA), MADAMIRA and its predecessor MADA use SVM to predict morphological properties (Pasha et al., 2014). For generic morphological segmentation, Morfessor (Smit et al., 2014) uses a max-likelihood in a semisupervised setting. Niether of these frameworks are adequate for joint transition-based morphosyntactic processing.

Goldberg and Tsarfaty (2008) have shown that in a generative setting, joint morphological segmentation and syntactic parsing of Modern Hebrew improves accuracy for both tasks, yielding state-of-the-art results. Bohnet et al. propose joint morphological and syntactic analysis in (Bohnet et al., 2013) and (Bohnet and Nivre, 2012b) of richly inflected languages, yielding results supporting the joint hypothesis, however the languages they process do not have variable length segmentations as a result of morphological disambiguation.

Variable-length sequences in beam search also exist in the structured prediction of constituency trees. Zhu et al (Zhu et al., 2013) introduced an IDLE transition (adopted in (Honnibal and Johnson, 2014) and (Zhang et al., 2014)) that, like POP, has no effect, but unlike POP occurs only at the end of parsing and has a non-deterministic number of occurrences. The IDLE transition likely solves the bias of only slightly varying length sequences, but would not be applicable to wildly varying lengths, such as is the case in morphological disambiguation of MRLs.

In the future, we intend to embed our morpheme-based MD in a joint framework for morphological and syntactic disambiguation, that is amendable to deal with variable length morpheme sequence – as is the case in Semitic languages. In the context of joint processing, a morpheme-based transition system is more suitable because it allows the dependency parser earlier access to disambiguated morphemes. In turn, the dependency parser can attempt to attach the morphemes to existing syntactic structures. Moreover, none of our definition is language-specific, and hence, our model is potentially suitable for cross-linguistic MD, which we are going to instantiate and evaluate in future work.

[7]POP feature templates are restricted to fire only when predicting a POP transition, all other feature templates are not applied