Joint Morphological and Syntactic Disambiguation*

Shay B. Cohen and **Noah A. Smith**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{scohen,nasmith}@cs.cmu.edu

Abstract

In morphologically rich languages, should morphological and syntactic disambiguation be treated sequentially or as a single problem? We describe several efficient, probabilistically-interpretable ways to apply **joint inference** to morphological and syntactic disambiguation using **lattice parsing**. Joint inference is shown to compare favorably to pipeline parsing methods across a variety of component models. State-of-the-art performance on Hebrew Treebank parsing is demonstrated using the new method. The benefits of joint inference are modest with the current component models, but appear to increase as components themselves improve.

1 Introduction

As the field of statistical NLP expands to handle more languages and domains, models appropriate for standard benchmark tasks do not always work well in new situations. Take, for example, parsing the *Wall Street Journal* Penn Treebank, a long-standing task for which highly accurate context-free models stabilized by the year 2000 (Collins, 1999; Charniak, 2000). On this task, the Collins model achieves 90% F_1-accuracy. Extended for new languages by Bikel (2004), it achieves only 75% on Arabic and 72% on Hebrew.[1]

It should come as no surprise that Semitic parsing lags behind English. The Collins model was carefully designed and tuned for *WSJ* English. Many of the features in the model depend on English syntax or Penn Treebank annotation conventions. Inherent in its crafting is the assumption that a million words of training text are available. Finally, for English, it need not handle morphological ambiguity.

Indeed, the figures cited above for Arabic and Hebrew are achieved using *gold-standard* morphological disambiguation and part-of-speech tagging. Given only surface *words*, Arabic performance drops by 1.5 F_1 points. The Hebrew Treebank (unlike Arabic) is built over *morphemes*, a convention we view as sensible, though it complicates parsing.

This paper considers parsing for *morphologically rich* languages, with Hebrew as a test case. Morphology and syntax are two levels of linguistic description that interact. This interaction, we argue, can affect disambiguation, so we explore here the matter of joint disambiguation. This involves the comparison of a **pipeline** (where morphology is inferred first and syntactic parsing follows) with **joint inference**. We present a generalization of the two, and show new ways to do joint inference for this task that does not involve a computational blow-up.

The paper is organized as follows. §2 describes the state of the art in NLP for Hebrew and some phenomena it exhibits that motivate joint inference for morphology and syntax. §3 describes our approach to joint inference using **lattice parsing**, and gives three variants of weighted lattice parsing with their probabilistic interpretations. The different factor models and their stand-alone performance are given in §4. §5 presents experiments on Hebrew parsing and explores the benefits of joint inference.

2 Background

In this section we discuss prior work on statistical morphological and syntactic processing of Hebrew and motivate the joint approach.

2.1 NLP for Modern Hebrew

Wintner (2004) reviews work in Hebrew NLP, emphasizing that the challenges stem from the writing system, rich morphology, unique word formation process of roots and patterns, and relative lack of annotated corpora.

We know of no publicly available statistical parser designed specifically for Hebrew. Sima'an et al.

* The authors acknowledge helpful feedback from the anonymous reviewers, Sharon Goldwater, Rebecca Hwa, Alon Lavie, and Shuly Wintner.

[1] Compared to the Penn Treebank, the Arabic Treebank (Maamouri et al., 2004) has 60% as many word tokens, and the Hebrew Treebank (Sima'an et al., 2001) has 6%.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 208–217, Prague, June 2007. ©2007 Association for Computational Linguistics

a. הרועה | באחו | הירוק | הגדול | והמרוחק | שרועה | שם | יפה

b. ה+ רועה ב+ה+ אחו ה+ ירוק ה+ גדול ו+ה+ מרוחק ש+ רועה שם יפה

c. is-beautiful there shepherds that distant the and big the green the meadow the in shepherd the
ADJ+Masc VB+Masc +Masc

d. the shepherd in the big green distant meadow who shepherds there is beautiful

e. ה+ רועה ב+ה+ אחו ה+ ירוק ה+ גדול ו+ה+ מרוחק שרועה שם יפה

f. nicely there is-lying distant the and big the green the meadow the in shepherd the
ADV VB+Fem +Fem

g. the shepherdess in the big green distant meadow is lying there nicely

h.

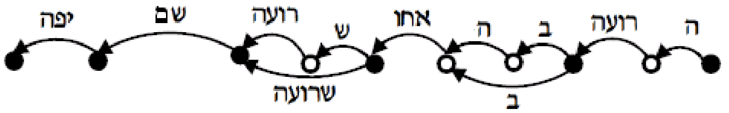

Figure 1: (a.) A sentence in Hebrew (to be read right to left), with (b.) one morphological analysis, (c.) English glosses, and (d.) natural translation; and (e.) a different morphological analysis, (f.) English glosses, and (g.) less natural translation. (h.) shows a morphological "sausage" lattice that encodes the morpheme-sequence analyses $L(\vec{x})$ possible for a shortened sentence (unmodified "meadow"). Shaded states are word boundaries, white states are intra-word morpheme boundaries; in practice we add POS tags to the arcs, to permit disambiguation. According to both native speakers we polled, both interpretations are grammatical—note the long-distance agreement required for grammaticality.

(2001) built a Hebrew Treebank of 88,747 words (4,783 sentences) and parsed it using a probabilistic model. However, they assumed that the input to the parser was already (perfectly) morphologically disambiguated. This assumption is very common in multilingual parsing (see, for example, Cowan et al., 2005, and Buchholz et al., 2006).

Until recently, the NLP literature on morphological processing was dominated by the largely non-probabilistic application of finite-state transducers (Kaplan and Kay, 1981; Koskenniemi, 1983; Beesley and Karttunen, 2003) and the largely unsupervised discovery of morphological patterns in text (Goldsmith, 2001; Wicentowski, 2002); Hebrew morphology receives special attention in Levinger et al. (1995), Daya et al. (2004), and Adler and Elhadad (2006). Lately a few supervised *disambiguation* methods have come about, including hidden Markov models (Hakkani-Tür et al., 2000; Hajič et al., 2001), conditional random fields (Kudo et al., 2004; Smith et al., 2005b), and local support vector machines (Habash and Rambow, 2005). There are also morphological disambiguators designed specifically for Hebrew (Segal, 2000; Bar-Haim et al., 2005).

2.2 Why Joint Inference?

In NLP, the separation of syntax and morphology is understandable when the latter is impoverished, as in English. When both involve high levels of ambiguity, this separation becomes harder to justify, as argued by Tsarfaty (2006). To our knowledge, that is the only study to move toward joint inference of syntax and morphology, presenting joint models and testing approximation of these models with two parsers: one a pipeline (segmentation → tagging → parsing), the other involved joint inference of segmentation and tagging, with the result piped to the parser. The latter was slightly more accurate. Tsarfaty discussed but did not carry out joint inference.

In a morphologically rich language, the different morphemes that make up a word can play a variety of different syntactic roles. A reasonable linguistic analysis might not make such morphemes immediate sisters in the tree. Indeed, the convention of the Hebrew Treebank is to place *morphemes* (rather than words) at the leaves of the parse tree, allowing morphemes of a word to attach to different nonterminal parents.[2]

Generating parse trees over morphemes requires the availability of morphological information when parsing. Because this analysis is not in general reducible to sequence labeling (tagging), the problem is different from POS tagging. Figure 1 gives an

[2] The Arabic Treebank, by contrast, annotates words morphologically but keeps the morphemes together as a single node tagged with a POS sequence. In Bikel's Arabic parser, complex POS tags are projected to a small atomic set; it is unclear how much information is lost.

example from Hebrew that illustrates the interaction between morphology and syntax. In this example, we show two interpretations of the surface text, with the first being a more common natural analysis for the sentence. The first and third-to-last words' analyses depend on each other if the resulting analysis is to be the more natural one: for this analysis the first seven words have to be a noun phrase, while for the less common analysis ("lying there nicely") only the first six words compose a noun phrase, with the last two words composing a verb phrase. Consistency depends on a long-distance dependency that a finite-state morphology model cannot capture, but a model that involves syntactic information can. Disambiguating the syntax aids in disambiguating the morphology, suggesting that a joint model will perform both more accurately.

In sum, joint inference of morphology and syntax is expected to allow decisions of both kinds to influence each other, enforce adherence to constraints at both levels, and to diminish the propagation of errors inherent in pipelines.

3 Joint Inference of Morphology and Syntax

We now formalize the problem and supply the necessary framework for performing joint morphological disambiguation and syntactic parsing.

3.1 Notation and Morphological Sausages

Let $\mathcal{X}$ be the language's word vocabulary and $\mathcal{M}$ be its morpheme inventory. The set of valid analyses for a surface word is defined using a morphological lexicon L, which defines $L(x) \subseteq \mathcal{M}^+$. $L(\vec{x}) \subseteq (\mathcal{M}^+)^+$ (sequence of sequences) is the set of whole-sentence analyses for sentence $\vec{x} = \langle x_1, x_2, ..., x_n \rangle$, produced by concatenating elements of $L(x_i)$ in order. $L(\vec{x})$ can be represented as an acyclic lattice with a "sausage" shape familiar from speech recognition (Mangu et al., 1999) and machine translation (Lavie et al., 2004). Fig. 1h shows a sausage lattice for a sentence in Hebrew. We use $\vec{m}$ to denote an element of $L(\vec{x})$ and $\vec{m}_i$ to denote an element of $L(x_i)$; in general, $\vec{m} = \langle \vec{m}_1, \vec{m}_2, ..., \vec{m}_n \rangle$.

We are interested in a function $f : \mathcal{X}^+ \rightarrow (\mathcal{M}^+)^+ \times \mathcal{T}$, where $\mathcal{T}$ is the set of syntactic trees for the language. f can be viewed as a structured classifier. We use $D_G(\vec{m}) \subseteq \mathcal{T}$ to denote the set of valid trees under a grammar G (here, a PCFG with terminal alphabet $\mathcal{M}$) for morpheme sequence $\vec{m}$. To be precise, $f(\vec{x})$ selects a mutually consistent morphological and syntactic analysis from

$$\text{GEN}(\vec{x}) = \{\langle \vec{m}, \tau \rangle \mid \vec{m} \in L(\vec{x}), \tau \in D_G(\vec{m})\}$$

3.2 Product of Experts

Our mapping $f(\vec{x})$ is based on a joint probability model $p(\tau, \vec{m} \mid \vec{x})$ which combines two probability models $p_G(\tau, \vec{m})$ (a PCFG built on the grammar G) and $p_L(\vec{m} \mid \vec{x})$ (a morphological disambiguation model built on the lexicon L). Factoring the joint model into sub-models simplifies **training**, since we can train each model separately, and **inference** (parsing), as we will see later in this section. Factored estimation has been quite popular in NLP of late (Klein and Manning, 2003b; Smith and Smith, 2004; Smith et al., 2005a, *inter alia*).

The most obvious joint parser uses p_G as a conditional model over trees given morphemes and maximizes the joint likelihood:

$$\begin{aligned} & f_{\text{lik}}(\vec{x}) \\ & = \operatorname*{argmax}_{\langle \vec{m}, \tau \rangle \in \text{GEN}(\vec{x})} p_G(\tau \mid \vec{m}) \cdot p_L(\vec{m} \mid \vec{x}) \qquad (1) \\ & = \operatorname*{argmax}_{\langle \vec{m}, \tau \rangle \in \text{GEN}(\vec{x})} \frac{p_G(\tau, \vec{m})}{\sum_{\tau'} p_G(\tau', \vec{m})} \cdot \frac{p_L(\vec{m}, \vec{x})}{\sum_{\vec{m}'} p_L(\vec{m}', \vec{x})} \end{aligned}$$

This is not straightforward, because it involves summing up the trees for each $\vec{m}$ to compute $p_G(\vec{m})$, which calls for the $O(|\vec{m}|^3)$-Inside algorithm to be called on each $\vec{m}$. Instead, we use the joint, $p_G(\tau, \vec{m})$, which, strictly speaking, makes the model deficient ("leaky"), but permits a dynamic programming solution.

Our models will be parametrized using either unnormalized weights (a log-linear model) or multinomial distributions. Either way, both models define *scores* over parts of analyses, and it may be advantageous to give one model relatively greater strength, especially since we often ignore constant normalizing factors. This is known as a **product of experts** (Hinton, 1999), where a new combined distribution over events is defined by multiplying component distributions together and renormalizing. In the present setting, for some value $\alpha \geq 0$,

$$f_{\text{poe},\alpha}(\vec{x}) = \operatorname*{argmax}_{\langle \vec{m},\tau\rangle \in \text{GEN}(\vec{x})} \frac{p_G(\tau, \vec{m}) \cdot p_L(\vec{m} \mid \vec{x})^\alpha}{Z(\vec{x}, \alpha)} \tag{2}$$

where $Z(\vec{x}, \alpha)$ need not be computed (since it is a constant in $\vec{m}$ and τ). α tunes the relative weight of the morphology model with respect to the parsing model. The higher α is, the more we trust the morphology model over the parser to correctly disambiguate the sentence. We might trust one model more than the other for a variety of reasons: it could be more robustly or discriminatively estimated, or it could be known to come from a more appropriate family.

This formulation also generalizes two more naïve parsing methods. If $\alpha = 0$, the morphology is modeled only through the PCFG and p_L is ignored except as a constraint on which analyses $L(\vec{x})$ are allowed (i.e., on the definition of the set $\text{GEN}(\vec{x})$). At the other extreme, as $\alpha \to +\infty$, p_L becomes more important. Because p_L does not predict trees, p_G still "gets to choose" the syntax tree, but in the limit it must find a tree for $\operatorname{argmax}_{\vec{m} \in L(\vec{x})} p_L(\vec{m} \mid \vec{x})$. This is effectively the morphology-first pipeline.[3]

3.3 Parsing Algorithms

To parse, we apply a dynamic programming algorithm in the $\langle$max, +$\rangle$ semiring to solve the $f_{\text{poe},\alpha}$ problem shown in Eq. 4. If p_L is a unigram-factored model, such that for some single-word morphological model υ we have

$$p_L(\vec{m} \mid \vec{x}) = \prod_{i=1}^{n} \upsilon(\vec{m}_i \mid x_i) \tag{3}$$

then we can implement morpho-syntactic parsing by *weighting* the sausage lattice. Let the weight of each arc that starts an analysis $\vec{m}_i \in L(x_i)$ be equal to $\log \upsilon(\vec{m}_i \mid x_i)$, and let other arcs have weight 0. In the parsing algorithm, the weight on an arc is summed in when the arc is first used to build a constituent.

In general, we would like to define a joint model that assigns (unnormalized) probabilities to elements of $\text{GEN}(\vec{x})$. If p_G is a PCFG and p_L can be described as a weighted finite-state transducer, then this joint model is their *weighted composition*, which is a weighted CFG; call the composed grammar I and its (unnormalized) distribution p_I. Compared to G, I will have many more nonterminals if p_L has a Markov order greater than 0 (unigram, as above). Because parsing runtime depends heavily on the grammar constant (at best, quadratic in the number of nonterminals), parsing with p_I is not computationally attractive.[4] $f_{\text{poe},\alpha}$ is not, then, a scalable solution when we wish to use a morphology model p_L that can make interdependent decisions about different words in $\vec{x}$ in context. We propose two new, efficient dynamic programming solutions for joint parsing.

In the first, we approximate the distribution $p_L(\vec{M} \mid \vec{x})$ using a unigram-factored model of the form in Eq. 3:

$$p'_L(\vec{m} \mid \vec{x}) = \prod_{i=1}^{n} \underbrace{p_L(\vec{M}_i = \vec{m}_i \mid \vec{x})}_{\text{posterior, depends on all of } \vec{x}} \tag{7}$$

Similar methods were applied by Matsuzaki et al. (2005) and Petrov and Klein (2007) for parsing under a PCFG with nonterminals with latent annotations. Their approach was variational, approximating the true posterior over coarse parses using a sentence-specific PCFG on the coarse nonterminals, created directly out of the true fine-grained PCFG. In our case, we approximate the full distribution over morphological analyses for the sentence by a simpler, sentence-specific unigram model that assumes each word's analysis is to be chosen independently of the others. Note that our *model* (p_L) does not make such an assumption, only the approximate model p'_L does, and the approximation is per-sentence. The idea resembles a mean-field variational approximation for graphical models. Turning to implementation, we can solve for $p_L(\vec{m}_i \mid \vec{x})$ exactly using the forward-backward algorithm. We will call this method $f_{\text{vari},\alpha}$ (see Eq. 5).

A closely related method, applied by Goodman (1996) is called **minimum-risk** decoding. Goodman called it "maximum expected recall" when applying it to parsing. In the HMM community it

[3] There is a slight difference. If no parse tree exists for the p_L-best morphological analysis, then a less probable $\vec{m}$ may be chosen. So as $\alpha \to +\infty$, we can view $f_{\text{lik},\alpha}$ as finding the best *grammatical* $\vec{m}$ and its best tree—not exactly a pipeline.

[4] In prior work involving factored syntax models—lexicalized (Klein and Manning, 2003b) and bilingual (Smith and Smith, 2004)—$f_{\text{poe},1}$ was applied, and the asymptotic runtime went to $O(n^5)$ and $O(n^7)$.

$$f_{\mathrm{poe},\alpha}(\vec{x}) = \operatorname*{argmax}_{\langle \vec{m},\tau\rangle \in \mathrm{GEN}(\vec{x})} \log p_G(\tau,\vec{m}) + \alpha \log p_L(\vec{m} \mid \vec{x}) \tag{4}$$

$$f_{\mathrm{vari},\alpha}(\vec{x}) = \operatorname*{argmax}_{\langle \vec{m},\tau\rangle \in \mathrm{GEN}(\vec{x})} \log p_G(\tau,\vec{m}) + \alpha \textstyle\sum_{i=1}^{n} \log p_L(\vec{m}_i \mid \vec{x}) \tag{5}$$

$$f_{\mathrm{risk},\alpha}(\vec{x}) = \operatorname*{argmax}_{\langle \vec{m},\tau\rangle \in \mathrm{GEN}(\vec{x})} \log p_G(\tau,\vec{m}) + \alpha \textstyle\sum_{i=1}^{n} p_L(\vec{m}_i \mid \vec{x}) \tag{6}$$

is sometimes called "posterior decoding." Minimum risk decoding is attributable to Goel and Byrne (2000). Applied to a single model, it factors the parsing decision by penalizable errors, and chooses the solution that minimizes the risk (expected number of errors under the model). This factors into a sum of expectations, one per potential mistake. This method is expensive for parsing models (since it requires the Inside algorithm to compute expected recall mistakes), but entirely reasonable for sequence labeling models. The idea is to score each word-analysis $\vec{m}_i$ in the morphological lattice by the expected value (under p_L) that $\vec{m}_i$ is present in the final analysis $\vec{m}$. This is, of course $p_L(\vec{M}_i = \vec{m}_i \mid \vec{x})$, the same quantity computed for $f_{\mathrm{vari},\alpha}$, except the score of a path in the lattice is now a *sum* of posteriors rather than a *product*. Our second approximate joint parser tries to maximize the *probability* of the parse (as before) and at the same time to minimize the *risk* of the morphological analysis. See $f_{\mathrm{risk},\alpha}$ in Eq. 6; the only difference between $f_{\mathrm{risk},\alpha}$ and $f_{\mathrm{vari},\alpha}$ is whether posteriors are added ($f_{\mathrm{risk},\alpha}$) or multiplied ($f_{\mathrm{vari},\alpha}$).

To summarize this section, $f_{\mathrm{vari},\alpha}$ and $f_{\mathrm{risk},\alpha}$ are two approximations to the expensive-in-general $f_{\mathrm{poe},\alpha}$ that boil down to parsing over weighted lattices. The only difference between them is how the lattice is weighted: using $\alpha \log p_L(\vec{m}_i \mid \vec{x})$ for $f_{\mathrm{vari},\alpha}$ or using $\alpha p_L(\vec{m}_i \mid \vec{x})$ for $f_{\mathrm{risk},\alpha}$.[5] In case of a unigram p_L, $f_{\mathrm{poe},\alpha}$ is equivalent to $f_{\mathrm{vari},\alpha}$; otherwise $f_{\mathrm{poe},\alpha}$ is likely to be too expensive.

3.4 Lattice Parsing

To parse the weighted lattices using $f_{\mathrm{vari},\alpha}$ and $f_{\mathrm{risk},\alpha}$ in the previous section, we use *lattice parsing*. Lattice parsing is a straightforward generalization of string parsing that indexes constituents by states in the lattice rather than word interstices. At parsing time, a ⟨max, +⟩ lattice parser finds the best combined parse tree and path through the lattice. Importantly, the data structures that are used in chart parsing need not change in order to accommodate lattices. The generalization over classic Earley or CKY parsing is simple: keep in the parsing chart constituents created over a pair of start state and end state (instead of start position and end position), and (if desired) factor in weights on lattice arcs; see Hall (2005).

4 Factored Models

A fair comparison of joint and pipeline parsing must make some attempt to control for the component models. We describe here two PCFGs we used for $p_G(\tau, \vec{m})$ and two finite-state morphological models we used for $p_L(\vec{m} \mid \vec{x})$. We show how these models perform in stand-alone evaluations. For all experiments, we used the Hebrew Treebank (Sima'an et al., 2001). After removing traces and removing functional information from the nonterminals, we had 3,770 sentences in the training set, 371 sentences in the development set (used primarily to select the value of α) and 370 sentences in the test set.

4.1 Syntax Model

Our first syntax model is an unbinarized PCFG trained using relative frequencies. Preterminal (POS tag → morpheme) rules are smoothed using backoff to a model that predicts the morpheme length and letter sequence. The PCFG is not binarized. This grammar is remarkably good, given the limited effort that went into it. The rules in the training set had **high coverage** with respect to the development set: an oracle experiment in which we maximized the number of recovered gold-standard constituents (on the development set) gave F_1 accuracy of 93.7%. In fact, its accuracy supersedes

[5] Until now, we have talked about weighting *word analyses*, which may cover several arcs, rather than arcs. In practice we apply the weight to the first arc of a word analysis, and weight the remaining arcs of that analysis with 0 (no cost or benefit), giving the desired effect.

more complex, lexicalized, models: given gold-standard morphology, it achieves 81.2% (compared to 72.0% by Bikel's parser, with head rules specified by a native speaker). This is probably attributable to the dataset's size, which makes training with highly-parameterized lexicalized models precarious and prone to overfitting. With first-order **vertical markovization** (i.e., annotating each nonterminal with its parent as in Johnson, 1998), accuracy is also at 81.2%. Tuning the horizontal markovization of the grammar rules (Klein and Manning, 2003a) had a small, adverse effect on this dataset.

Since the PCFG model was relatively successful compared to lexicalized models, and is faster to run, we decided to use a vanilla PCFG, denoted G_{van}, and a parent-annotated version of that PCFG (Johnson, 1998), denoted $G_{\mathrm{v}=2}$.

4.2 Morphology Model

Both of our morphology models use the same morphological lexicon L, which we describe first.

4.2.1 Morphological Lexicon

In this work, a morphological analysis of a word is a sequence of morphemes, possibly with a tag for each morpheme. There are several available analyzers for Hebrew, including Yona and Wintner (2005) and Segal (2000). We use instead an empirically-constructed generative lexicon that has the advantage of matching the Treebank data and conventions. If the Treebank is enriched, this would then directly benefit the lexicon and our models.

Starting with the training data from the Hebrew Treebank, we first create a set of **prefixes** $\mathcal{M}_{\mathrm{p}} \subset \mathcal{M}$; this set includes any morpheme seen in a non-final position within any word. We also create a set of **stems** $\mathcal{M}_{\mathrm{s}} \subset \mathcal{M}$ that includes any morpheme seen in a final position in a word. This effectively captures the morphological analysis convention in the Hebrew Treebank, where a stem is prefixed by a relatively dominant low-entropy sequence of 0–5 prefix morphemes. For example, *MHKLB* ("from the dog") is analyzed as *M+H+KLB* with prefixes *M* ("from") and *H* ("the") and *KLB* ("dog") is the stem. In practice, $|\mathcal{M}_{\mathrm{p}}| = 124$ (including some conventions for numerals) and $|\mathcal{M}_{\mathrm{s}}| = 13{,}588$. The morphological lexicon is then defined as any analysis given $\mathcal{M}_{\mathrm{p}}$ and $\mathcal{M}_{\mathrm{s}}$:

$$\begin{aligned} L(x) \;=\; & \{m_1^k \in \mathcal{M}_{\mathrm{p}}^* \times \mathcal{M}_{\mathrm{s}} \mid \mathrm{concat}(m_1^k) = x)\} \\ & \cup \{m_1^k \mid \mathrm{count}(m_1^k, x) \geq 1\} \end{aligned} \tag{9}$$

where m_1^k denotes $\langle m_1, ..., m_k \rangle$ and $\mathrm{count}(m_1^k, x)$ denotes the number of occurrences of x disambiguated as m_1^k in the training set. Note that $L(x)$ also includes any analysis of x observed in the training data. This permits the memorization of any observed analysis that is more involved than simple segmentation (4% of word tokens in the training set; e.g., *LXDR* ("to the room") is analyzed as *L+H+XDR*). This will have an effect on evaluation (see §5.1). On the development data, L has 98.6% coverage.

4.2.2 Unigram Baseline

The baseline morphology model, p_L^{uni}, first defines a joint distribution following Eq. 8. The word model factors out when we conditionalize to form $p_L^{\mathrm{uni}}(\langle m_1, ..., m_k \rangle \mid x)$. The prefix sequence model is multinomial estimated by MLE. The stem model (conditioned on the prefix sequence) is smoothed to permit *any* stem that is a sequence of Hebrew characters. On the development data, p_L^{uni} is 88.8% accurate (by word).

4.2.3 Conditional Random Field

The second morphology model, p_L^{crf}, which is based on the same morphological lexicon L, uses a second-order conditional random field (Lafferty et al., 2001) to disambiguate the full sentence by modeling local contexts (Kudo et al., 2004; Smith et al., 2005b). Space does not permit a full description; the model uses all the features of Smith et al. (2005b) except the "lemma" portion of the model, since the Hebrew Treebank does not provide lemmas. The weights are trained to maximize the probability of the correct path through the morphological lattice, conditioned on the lattice. This is therefore a **discriminative** model that defines $p_L(\vec{m} \mid \vec{x})$ directly, though we ignore the normalization factor in parsing.

Until now we have described p_L as a model of morphemes, but this CRF is trained to predict POS tags as well—we can either use the tags (i.e., label the morphological lattice with tag/morpheme pairs,

$$p_L^{\text{uni}}(\langle m_1, m_2, ..., m_k \rangle, x) = \underbrace{p(x \mid \langle m_1, m_2, ..., m_k \rangle)}_{\text{word}} \cdot \underbrace{p(m_k \mid \langle m_1, ..., m_{k-1} \rangle)}_{\text{stem}} \cdot \underbrace{p(\langle m_1, ..., m_{k-1} \rangle)}_{\text{prefix sequence}} \quad (8)$$

so that the lattice parser finds a parse that is consistent under both models), or sum the tags out and let the parser do the tagging. One subtlety is the tagging of words not seen in the training data; for such words an unsegmented hypothesis with tag UNKNOWN is included in the lattice and may therefore be selected by the CRF. On the development data, p_L^{crf} is 89.8% accurate on morphology, with 74.9% fine-grained POS-tagging F_1-accuracy (see §5.1).

Note on generative and discriminative models. The reader may be skeptical of our choice to combine a generative PCFG with a discrimative CRF. We point out that both are used to define conditional distributions over desired "output" structures given "input" sequences. Notwithstanding the fact that the factors can be *estimated* in very different ways, our combination in an exact or approximate product-of-experts is a reasonable and principled approach.

5 Experiments

In this section we evaluate parsing performance, but an evaluation issue is resolved first.

5.1 Evaluation Measures

The "Parseval" measures (Black et al., 1991) are used to evaluate a parser's phrase-structure trees against a gold standard. They compute precision and recall of constituents, each indexed by a label and two endpoints. As pointed out by Tsarfaty (2006), joint parsing of morphology and syntax renders this indexing inappropriate, since it assumes the yields of the trees are identical—that assumption is violated if there are any errors in the hypothesized $\vec{m}$. Tsarfaty (2006) instead indexed by non-whitespace *character* positions, to deal with segmentation mismatches. In general (and in this work) that is still insufficient, since $L(\vec{x})$ may include $\vec{m}$ that are not simply segmentations of $\vec{x}$ (see §4.2.1).

Roark et al. (2006) propose an evaluation metric for comparing a parse tree over a sentence generated by a speech recognizer to a gold-standard parse. As in our case, the hypothesized tree could have a different yield than the original gold-standard parse tree, because of errors made by the speech recognizer. The metric is based on an alignment between the hypothesized sentence and the gold-standard sentence. We used a similar evaluation metric, which takes into account the information about parallel word boundaries as well, a piece of information that does not appear naturally in speech recognition. Given the correct $\vec{m}^*$ and the hypothesis $\hat{\vec{m}}$, we use dynamic programming to find an optimal many-to-many monotonic alignment between the atomic morphemes in the two sequences. The algorithm penalizes each violation (by a morpheme) of a one-to-one correspondence,[6] and each character edit required to transform one side of a correspondence into the other (without whitespace). Word boundaries are (here) known and included as index positions. In the case where $\hat{\vec{m}} = \vec{m}^*$ (or equal up to whitespace) the method is identical to Parseval (and also to Tsarfaty, 2006). POS tag accuracy is evaluated the same way, for the same reasons; we report F_1-accuracy for tagging and parsing.

5.2 Experimental Comparison

In our experiment we vary four settings:

- Decoding algorithm: $f_{\text{poe},\alpha}$, $f_{\text{risk},\alpha}$, or $f_{\text{vari},\alpha}$ (§3.3).
- Syntax model: G_{van} or $G_{\text{v=2}}$ (§4.1).
- Morphology model: p_L^{uni} or p_L^{crf} (§4.2). In the latter case, we can use the scores over morpheme sequences only (summing out tags before lattice parsing; denoted m.-p_L^{crf}) or the full model over morphemes and tags, denoted t.-p_L^{crf}.[7]
- α, the relative strength given to the morphology model (see §3). We tested values of α in $\{0, +\infty\} \cup \{10^q \mid q \in \{0, 1, ..., 16\}\}$. Recall that $\alpha = 0$ ignores the morphology model probabilities altogether (using an unweighted lattice),

[6] That is, in a correspondence of a morphemes in one string with b in the other, the penalty is $a + b - 2$, since the morpheme on each side is not in violation.

[7] One subtlety is that any arc with the UNKNOWN POS tag can be relabeled—to any other tag—by the syntax model, whose preterminal rules are smoothed. This was crucial for $\alpha = +\infty$ (pipeline) parsing with t.-p_L^{crf} as the morphology model, since the parser does not recognize UNKNOWN as a tag.

			tuned α				pipeline ($\alpha \rightarrow +\infty$)			
parser	morph. model	syntax model	seg. acc	fine POS F_1	coarse POS F_1	parse F_1	seg. acc	fine POS F_1	coarse POS F_1	parse F_1
$f_{\text{poe},\alpha}$	p_L^{uni}	$p_{G_{\text{van}}}$	88.0	70.6	75.5	59.5	88.5	71.5	76.1	59.8
		$p_{G_{\text{v=2}}}$	88.0	70.7	75.8	60.4	88.6	70.8	75.7	59.9
	m.-p_L^{crf}	$p_{G_{\text{van}}}$	$*$	$*$	$*$	$*$	90.9	75.6	80.2	63.7
		$p_{G_{\text{v=2}}}$	$*$	$*$	$*$	$*$	90.9	75.3	80.2	64.2
	t.-p_L^{crf}	$p_{G_{\text{van}}}$	$*$	$*$	$*$	$*$	90.9	77.2	†81.5	63.0
		$p_{G_{\text{v=2}}}$	$*$	$*$	$*$	$*$	90.9	77.2	†81.5	64.0
$f_{\text{risk},\alpha}$	p_L^{uni}	$p_{G_{\text{van}}}$	87.9	70.9	75.3	58.9	**88.5**	71.5	76.1	59.8
		$p_{G_{\text{v=2}}}$	87.8	**70.9**	75.6	59.5	**88.6**	70.8	75.6	**59.9**
	m.-p_L^{crf}	$p_{G_{\text{van}}}$	89.8	74.5	78.9	62.5	89.8	74.5	78.9	62.4
		$p_{G_{\text{v=2}}}$	89.8	74.3	79.1	63.0	89.8	74.3	79.1	63.0
	t.-p_L^{crf}	$p_{G_{\text{van}}}$	**90.2**	76.6	80.5	62.4	89.9	76.4	80.4	61.6
		$p_{G_{\text{v=2}}}$	**90.2**	76.6	80.5	**63.1**	89.9	76.4	80.4	62.2
$f_{\text{vari},\alpha}$	p_L^{uni}	$p_{G_{\text{van}}}$	88.0	70.6	75.5	59.5	88.5	71.5	76.1	59.8
		$p_{G_{\text{v=2}}}$	88.0	70.7	75.8	60.4	88.6	70.8	75.7	59.9
	m.-p_L^{crf}	$p_{G_{\text{van}}}$	†**91.1**	**75.6**	**80.4**	**64.0**	90.9	74.8	79.3	62.9
		$p_{G_{\text{v=2}}}$	**90.9**	**75.4**	**80.5**	†**64.4**	90.1	74.6	79.5	63.2
	t.-p_L^{crf}	$p_{G_{\text{van}}}$	†**91.3**	†**77.7**	†81.7	63.0	90.9	77.0	†81.3	62.6
		$p_{G_{\text{v=2}}}$	†**91.3**	†77.6	†81.6	63.6	90.9	77.0	†81.3	63.6

Table 1: Results of experiments on Hebrew (test data, max. length 40). This table shows the performance of **joint parsing** (finite α; left) and a **pipeline** ($\alpha \rightarrow +\infty$; right). Joint parsing with a non-unigram morphology model is too expensive (marked $*$). Morphological analysis accuracy (by word), fine-grained (full tags) and coarse-grained (only parts of speech) POS tagging accuracy (F_1), and generalized constituent accuracy (F_1) are reported; α was tuned for each of these separately. **Boldface** denotes that figures were significantly better than their counterparts in the same row, under a binomial sign test ($p < 0.05$). † marks the best overall accuracy and figures that are not significantly worse (binomial sign test, $p < 0.05$).

and as $\alpha \rightarrow +\infty$ a morphology-first pipeline is approached.

We measured four outcome values: segmentation accuracy (fraction of word tokens segmented correctly), fine- and coarse-grained tagging accuracy,[8] and parsing accuracy. For tagging and parsing, F_1-measures are given, according to the generalized evaluation measure described in §5.1.

5.3 Results

Tab. 1 compares parsing with tuned α values to the pipeline.

The best results were achieved using $f_{\text{vari},\alpha}$, using the CRF and joint disambiguation. Without the CRF (using p_L^{uni}), the difference between the decoding algorithms is less apparent, suggesting an interaction between the sophistication of the components and the best way to decode with them. These results suggest that $f_{\text{vari},\alpha}$, which permits p_L to "veto" any structure involving a morphological analysis for any word that is *a posteriori* unlikely (note that $\log p_L(\vec{m}_i \mid \vec{x})$ can be an arbitrarily large negative number), is beneficial as a "filter" on parses.[9] $f_{\text{risk},\alpha}$, on the other hand, is only allowed to give "bonuses" of up to α to each morphological analysis that p_L believes in; its influence is therefore weaker. This result is consistent with the findings of Petrov et al. (2007) for another approximate parsing task.

The advantage of the parent-annotated PCFG is also more apparent when the CRF is used for morphology, and when α is tuned. All other things equal, then, p_L^{crf} led to higher accuracy all around. Letting the CRF help predict the POS tags helped tagging accuracy but not parsing accuracy.

While the gains over the pipeline are modest, the segmentation, fine POS, and parsing accuracy scores achieved by joint disambiguation with $f_{\text{vari},\alpha}$ with the CRF are significantly better than any of the pipeline conditions.

Interestingly, if we had not tested with the CRF, we might have reached a very different conclusion about the usefulness of tuning α as opposed to a pipeline. With the unigram morphology model, joint parsing frequently *underperforms* the pipeline, sometimes even signficantly. The explanation, we

[8] Although the Hebrew Treebank is small, the size of its POS tagset is large (four times larger than the Penn Treebank), because the tags encode morphological features (gender, person, and number). These features have either been ignored in prior work or encoded differently. In order for our POS-tagging figures to be reasonably comparable to previous work, we include accuracy for coarse-grained tags (only the core part of speech) tags as well as the detailed Hebrew Treebank tags.

[9] Another way to describe this combination is to call it a product of $|\vec{x}|+1$ experts: one for the morphological analysis of each word, plus the grammar. The morphology experts (softly) veto any analysis that is dubious based on surface criteria, and the grammar (softly) vetoes less-grammatical parses.

parser	morph. model	syntax model	seg. acc	fine POS F_1	coarse POS F_1	parse F_1
$f_{\text{risk},\alpha}$	p_L^{uni}	$p_{G_{\text{van}}}$	90.7	73.4	78.5	64.3
		$p_{G_{\text{v=2}}}$	90.2	73.0	78.5	64.9
	m.-p_L^{crf}	$p_{G_{\text{van}}}$	90.7	75.4	80.0	65.2
		$p_{G_{\text{v=2}}}$	90.8	75.1	80.2	65.4
	t.-p_L^{crf}	$p_{G_{\text{van}}}$	91.2	78.1	82.4	65.7
		$p_{G_{\text{v=2}}}$	91.1	78.0	82.2	66.2
$f_{\text{vari},\alpha}$	p_L^{uni}	$p_{G_{\text{van}}}$	90.6	73.2	78.3	63.5
		$p_{G_{\text{v=2}}}$	90.2	72.8	78.4	64.4
	m.-p_L^{crf}	$p_{G_{\text{van}}}$	92.0	76.6	81.5	66.9
		$p_{G_{\text{v=2}}}$	91.9	76.2	81.6	66.9
	t.-p_L^{crf}	$p_{G_{\text{van}}}$	91.8	79.1	83.2	66.5
		$p_{G_{\text{v=2}}}$	91.7	78.7	83.0	67.4

Table 2: **Oracle** results of experiments on Hebrew (test data, max. length 40). This table shows the performance of morphological segmentation, part-of-speech tagging, coarse part-of-speech tagging and parsing when using an oracle to select the best α for each sentence. The notation and interpretation of the numbers are the same as in Tab. 1.

believe, has to do with the ability of the unigram model to estimate a good *distribution* over analyses. While the unigram model is nearly as good as the CRF at picking the right segmentation for a word, joint parsing demands much more. In case the best segmentation does not lead to a grammatical morpheme sequence (under the syntax model), the morphology model needs to be able to give relative strengths to the alternatives. The unigram model is less able to do this, because it ignores the context of the word, and so the benefit of joint parsing is lost.

Most commonly the tuned value of α is around 10 (not shown, to preserve clarity). Because of ignored normalization constants, this does *not* mean that morphology is "10× more important than syntax," but it *does* mean that, for a particular p_L and p_G, tuning their relative importance in decoding can improve accuracy. In Tab. 2 we show how performance would improve if the oracle value of α was selected for each test-set *sentence*; this further highlights the potential impact of perfecting the tradeoff between models. Of course, selecting α automatically at test-time, per sentence, is an open problem.

To our knowledge, the parsers we have described represent the state-of-the-art in Modern Hebrew parsing. The closest result is Tsarfaty (2006), which we have not directly replicated. Tsarfaty's model is essentially a pipeline application of $f_{\text{poe},\infty}$ with a grammar like $p_{G_{\text{van}}}$. Her work focused more on the interplay between the segmentation and POS tagging models and the amount of information passed to the parser. Some key differences preclude direct comparison: we modeled fine-grained tags (though we report both kinds of tagging accurcy), we employed a richer morphological lexicon (permitting analyses that are not just segmentation), and a different training/test split and length filter (we used longer sentences). Nonetheless, our conclusions support the argument in Tsarfaty (2006) for more integrated parsing methods.

We conclude that tuning the relative importance of the two models—rather than pipelining to give one infinitely more importance—can provide an improvement on segmentation, tagging, and parsing accuracy. This suggests that future parsing efforts for languages with rich morphology might continue to assume separately-trained (and separately-improved) morphology and syntax components, which would stand to gain from joint decoding. In our experiments, better morphological disambiguation was crucial to getting any benefit from joint decoding. Our result also suggests that exploring new, fully-integrated models (and training methods for them) may be advantageous.

6 Conclusion

We showed that joint morpho-syntactic parsing can improve the accuracy of both kinds of disambiguation. Several efficient parsing methods were presented, using factored state-of-the-art morphology and syntax models for the language under consideration. We demonstrated state-of-the-art performance on and consistent improvements across many settings for Modern Hebrew, a morphologically-rich language with a relatively small treebank.

References

M. Adler and M. Elhadad. 2006. An unsupervised morpheme-based HMM for Hebrew morphological disambiguation. In *Proc. of COLING-ACL*.

R. Bar-Haim, K. Sima'an, and Y. Winter. 2005. Choosing an optimal architecture for segmentation and POS-tagging of Modern Hebrew. In *Proc. of ACL Workshop on Computational Approaches to Semitic Languages*.

K. R. Beesley and L. Karttunen. 2003. *Finite State Morphology*. CSLI.

D. Bikel. 2004. Multilingual statistical parsing engine. `http://www.cis.upenn.edu/~dbikel/software.html#stat-parser`.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proc. of DARPA Workshop on Speech and Natural Language*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of NAACL*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, U. Penn.

B. Cowan and M. Collins. 2005. Morphology and reranking for the statistical parsing of Spanish. In *Proc. of HLT-EMNLP*.

E. Daya, D. Roth, and S. Wintner. 2004. Learning Hebrew roots: Machine learning with linguistic constraints. In *Proc. of EMNLP*.

V. Goel and W. Byrne. 2000. Minimum Bayes risk automatic speech recognition. *Computer Speech and Language*, 14(2):115–135.

J. Goldsmith. 2001. Unsupervised learning of the morphology of natural language. *Comp. Ling.*, 27(2):153–198.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proc. of ACL*.

N. Habash and O. Rambow. 2005. Arabic tokenization, part-of-speech tagging, and morphological disambiguation in one fell swoop. In *Proc. of ACL*.

J. Hajič, P. Krbec, P. Květoň, K. Oliva, and V. Petkevič. 2001. Serial combination of rules and statistics: A case study in Czech tagging. In *Proc. of ACL*.

D. Z. Hakkani-Tür, K. Oflazer, and G. Tür. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proc. of COLING*.

K. Hall. 2005. *Best-first Word-lattice Parsing: Techniques for Integrated Syntactic Language Modeling*. Ph.D. thesis, Brown University.

G. E. Hinton. 1999. Products of experts. In *Proc. of ICANN*.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Comp. Ling.*, 24(4):613–632.

R. M. Kaplan and M. Kay. 1981. Phonological rules and finite-state transducers. Presented at LSA.

D. Klein and C. D. Manning. 2003a. Accurate unlexicalized parsing. In *Proc. of ACL*, pages 423–430.

D. Klein and C. D. Manning. 2003b. Fast exact inference with a factored model for natural language parsing. In *Advances in NIPS 15*.

K. Koskenniemi. 1983. A general computational model of word-form recognition and production. Technical Report 11, University of Helsinki.

T. Kudo, K. Yamamoto, and Y. Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proc. of EMNLP*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

A. Lavie, S. Wintner, Y. Eytani, E. Peterson, and K. Probst. 2004. Rapid prototyping of a transfer-based Hebrew-to-English machine translation system. In *Proc. of TMI*.

M. Levinger, U. Ornan, and A. Itai. 1995. Learning morpholexical probabilities from an untagged corpus with an application to Hebrew. *Comp. Ling.*, 21:383–404.

M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki. 2004. The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. In *Proc. of NEMLAR*.

L. Mangu, E. Brill, and A. Stolcke. 1999. Finding consensus among words: Lattice-based word error minimization. In *Proc. of ECSCT*.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc. of ACL*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of HLT-NAACL*.

B. Roark, M. Harper, E. Charniak, B. Dorr, M. Johnson, J. Kahn, Y. Liu, M. Ostendorf, J. Hale, A. Krasnyanskaya, M. Lease, I. Shafran, M. Snover, R. Stewart, and Lisa Yung. 2006. Sparseval: Evaluation metrics for parsing speech. In *Proc. of LREC*.

E. Segal. 2000. A probabilistic morphological analyzer for Hebrew undotted texts. Master's thesis, Technion.

K. Sima'an, A. Itai, Y. Winter, A. Altman, and N. Nativ. 2001. Building a treebank of modern Hebrew text. *Journal Traitement Automatique des Langues*. Available at `http://mila.cs.technion.ac.il`.

D. A. Smith and N. A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proc. of EMNLP*, pages 49–56.

A. Smith, T. Cohn, and M. Osborne. 2005a. Logarithmic opinion pools for conditional random fields. In *Proc. of ACL*.

N. A. Smith, D. A. Smith, and R. W. Tromble. 2005b. Context-based morphological disambiguation with random fields. In *Proc. of HLT-EMNLP*.

R. Tsarfaty. 2006. Integrated morphological and syntactic disambiguation for Modern Hebrew. In *Proc. of COLING-ACL Student Research Workshop*.

R. Wicentowski. 2002. *Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework*. Ph.D. thesis, Johns Hopkins U.

S. Wintner. 2004. Hebrew computational linguistics: Past and future. *Art. Int. Rev.*, 21(2):113–138.

S. Yona and S. Wintner. 2005. A finite-state morphological grammar of Hebrew. In *Proc. of ACL Workshop on Computational Approaches to Semitic Languages*.