A Pointer Network Architecture for Joint Morphological Segmentation and Tagging

Amit Seker
Open University, Ra'anana, Israel
aseker00@gmail.com

Reut Tsarfaty
Bar Ilan University, Israel
reut.tsarfaty@biu.ac.il

Abstract

Morphologically Rich Languages (MRLs) such as Arabic, Hebrew and Turkish often require *Morphological Disambiguation* (MD), i.e., the prediction of the correct morphological decomposition of tokens into morphemes, early in the pipeline. Neural MD may be addressed as a simple pipeline, where segmentation is followed by sequence tagging, or as an end-to-end model, predicting morphemes from raw tokens. Both approaches are suboptimal; the former is heavily prone to error propagation, and the latter does not enjoy explicit access to the basic processing units called *morphemes*. This paper offers an MD architecture that combines the symbolic knowledge of morphemes with the learning capacity of neural end-to-end modeling. We propose a new, general and easy-to-implement Pointer Network model where the input is a morphological lattice and the output is a sequence of indices pointing at a single disambiguated path of morphemes. We demonstrate the efficacy of the model on segmentation and tagging, for Hebrew and Turkish texts, based on their respective Universal Dependencies (UD) treebanks. Our experiments show that with complete lattices, our model outperforms all shared-task results on segmenting and tagging these languages. On the SPMRL treebank, our model outperforms all previously reported results for Hebrew MD in realistic scenarios.

1 Introduction

In *Morphologically Rich Languages* (MRLs) (Tsarfaty et al., 2010), raw tokens are morphologically ambiguous, complex, and consist of sub-token units referred to as morphemes.[1] *Morphological Disambiguation* (MD) is the task of decomposing the tokens into their constituent morphemes, to be used as the basic processing units for NLP tasks down the pipeline (Mueller et al., 2013; More and Tsarfaty, 2016). As opposed to the commonly known scenario of *morphological tagging* (Bohnet et al., 2013), where every input token is assigned a single morphological signature (containing its lemma, part-of-speech tag, and morphological features such as gender, number, person, tense, etc.), in the MD scenario internally-complex input tokens may consist of multiple distinct units, each of which gets assigned its own morphological signature.

Pre-neural statistical approaches for MD (Barhaim et al., 2008; Adler and Elhadad, 2006a; Lee et al., 2011; Habash et al., 2013) typically used weighted finite-state machines to unravel the possible morphological decompositions, and classic machine learning models to select the most likely decomposition. Current neural models, however, take radically different paths.

One neural approach to MD employs pipeline, where a predicted segmentation of words into morphemes is passed on to sequence labeling component that performs tagging of each segment in context. This segmentation-first scenario employs sequence tagging to assign a single morphological tag to each segment similar to POS tagging in English, where each token in the input sequence is assigned a single label by the tagger. This method might be expected to work for MRLs just as well as standard NLP models do for English tagging, however, in actuality, such pipeline architectures are prone to error propagation, which undermines the accuracy of almost any task down the NLP pipeline (tagging, parsing, named entity recognition, relation extraction, etc.) (Tsarfaty et al., 2020; Klein and Tsarfaty, 2020; Bareket and Tsarfaty, 2020).

A second conceivable approach is an end-to-end sequence-to-sequence model that consumes a sequence of tokens (or characters) and produces a

[1] In Universal Dependencies terms, these are called *syntactic words*, to be distinguished from *raw input tokens*.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4368–4378

Hebrew Token	Morphological Analysis	English Translation
bbit	b/ADP bit/NOUN	in a house
	b/ADP h/DET bit/NOUN	in the house
hlbn	h/DET lbn/NOUN	the buttermilk
	h/DET lbn/ADJ	the white
	hlbn/VERB	whitened

Table 1: Partial list of *Morphological Analyses* for the Hebrew tokens: *bbit hlbn*. Each analysis is expressed as a list of morphological properties. In this example we only list the Segment/Tag properties.

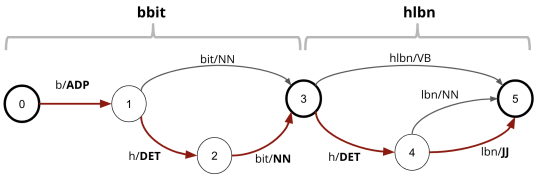

Figure 1: Lattice of the Hebrew tokens *'bbit hlbn'* corresponding to the example in Table 1. Edges are morphemes. Nodes are segmentation boundaries. Bold nodes are token boundaries. Every path through the lattice represents a single morphological analysis.

sequence of morphological signatures. Notably, the number of morphological signatures may vastly exceed the number of input tokens, (e.g., with an average of 1.4 tags per word in Hebrew). The drawback of this approach is that the model has no access to morphological information in the input, and is expected to extract all morphological information directly from the raw text. Tokens in MRLs are lexically and syntactically ambiguous, and carry many possible interpretations, so it is unclear if the surface signal is in fact sufficient. This fact is exacerbated by the fact that some MRLs are low resourced and even with pre-trained word embeddings, many forms are lacking when operating on internally-complex tokens.

In this paper we propose an alternative approach, that enjoys the power of end-to-end neural modeling while maintaining access to morphemes. We frame the problem as a *Morphological Analysis and Disambiguation* (MA&D) task, in which every raw token in the input sequence first goes through *Morphological Analysis* (MA) that exposes *all* of its possible morphological decompositions as a lattice (see Figure 1). This morphological lattice is then passed to the MD component, based on a *Pointer Network*, which selects a sequence of most likely arcs in the context of the sentence being processed. Since every lattice arc contains rich information that is made available by the MA — namely, segmentation boundaries, lemma, Part-of-Speech tag and a set of morphological features — this MA&D framework can jointly predict rich morphological layers while avoiding the pipeline pitfall.

Based on this architecture, we design a neural model for joint *segmentation and tagging* and apply it to two MRLs, Hebrew and Turkish. In *realistic* circumstances, the lexical *coverage* of the lattice may be partial, and we report MD results in both *ideal* and *realistic* scenarios. Our results on the Hebrew and Turkish UD treebanks show state-of-the-art performance for complete morphological lattices, and on the Hebrew SPMRL treebank we outperform all previous results in *realistic* scenarios. Our MA&D solution is generic and can be applied to any language, e.g., assuming MA components as provided in More et al. (2018). In addition, our proposed architecture is suitable for any other task that encodes information in a lattice towards further disambiguation.

2 Linguistic Data and Task Setup

Input tokens in MRLs are internally complex, and bear multiple units of meaning. *Morphological Analysis* (MA) is aimed to convert each of the tokens to a set of all possible morphological decompositions licensed by the rules of the language. A single decomposition represents a possible interpretation of the token being analyzed. Consider the Hebrew phrase *bbit hlbn*.[2] A partial list of analyses is presented in Table 1. A lattice representation of the analyses is illustrated in Figure 1.

Morphological Disambiguation (MD) is the task of selecting a single most-likely analysis for each token in the context of the sentence. The resulting morpheme sequence may then serve as the input processing units for downstream tasks (similarly to space-delimited words in English). Our above example, *bbit hlbn* is likely to be disambiguated as:

(1) *b*/ADP+*h*/DET+*bit*/NOUN+*h*/DET+*lbn*/ADJ
literally: in+the+house+the+white
translated: "in the white house".

The ambiguous MA output is stored in a *lattice* data structure. A Lattice is *Directed Acyclic Graph* (DAG) often used to encode ambiguity in NLP. In a morphological lattice, every node represents a segment boundary, and every edge represents a morpheme. Every path through the lattice represents

[2]We assume the transliteration of (Sima'any et al., 2001).

a single possible analysis of the entire sentence. Notably, not all segmental forms in the lattice are *overt* in the input stream. Some are implicit, due to intricate morpho-phonological and orthographic processes. For example, the analysis of the token *bbit* contains three morphological segments *b, h, bit* in the chosen path, yet the *h* segment is not visible in the input token *bbit* (Figure 1).

3 Proposed Method

The Task The *input* to our MA&D framework is a sequence of tokens and the *output* is a sequence of disambiguated morphological analyses, one per token. We assume a symbolic MA that generates ambiguous lattices containing all possible morphological analyses per token, based on a broad-coverage lexicon and/or symbolic rules of the language.

Given an input lattice, we frame MD as a lattice disambiguation task. Sperber et al. (2019) approached this task by constructing a specific architecture that captures the lattice representation. We, in contrast, choose to modify the *lattice representation* and feed it to an existing network architecture.

The key idea, in a nutshell, is to linearize the lattice into a sequence of partially-ordered analyses, and feed this partial order to a pointer network. For each token, the network will then learn to *point to* (select) the most likely analysis, preserving the linear constraints captured in the lattice structure.

Pointer Network (PtrNet) *Pointer networks* (Vinyals et al., 2015) are designed as a special case of *Sequence-to-Sequence* (Seq2Seq) networks. Seq2Seq models take an input sequence and produce an output sequence which may differ in length and vocabulary. PtrNet in addition can handle output vocabulary depending on the input sequence which can be variable in length.

Seq2Seq is composed of an encoder and a decoder. The encoder consumes and encodes the entire (embedded) input sequence. Then, the decoder is fed the entire encoded input representation and step by step produces discrete outputs which are fed back as input to the next decoding step.

PtrNets have an additional *Copy* Attention layer. The attention layer focuses on specific elements of the encoded input sequence at each decoding step (Luong et al., 2015). *Copy* Attention is a special case where the attention weights determine which input element the decoder's state is most aligned with, which can then be copied to the output.

Pointer Networks for MD (PtrNetMD) The PtrNet architecture is designed to learn the conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in an input sequence (Vinyals et al., 2015). Our goal is then to encode the morphological lattice as a sequence, and then feed it to the PtrNet so that the individual analyses in the lattice can be *pointed*, selected and copied into the output sequence, while respecting the lattice ordering constraints.

Given a lattice we serialize it by going over each token and listing all of its analyses. The linearization function maps a sequence of *n* tokens into a sequence of *m* analyses while preserving the partial order of the tokens, and where *m* is the sum of all token analyses. That is, for input tokens $t_1, ...t_n$, let $a_j{}^i$ denote the *i*'th analysis of the *j*'th token. Then the following holds, such that $\sum_{i=1}^{n} k_i = m$.

(2) $linearize(t_1, t_2, t_3, ..., t_n) =$
$a_1^1, ..., a_1^{k_1}, a_2^1, ..., a_2^{k_2}, ..., a_n^1, ..., a_n^{k_n}$

An analysis a_j^i is expressed as a list of morphemes where each morpheme is represented as a tuple of morphological properties. Both the SPMRL and UD scheme specify four properties *Form, Lemma, POS Tag, Morphological Features*. For example, (3) is an analysis composed of three morphemes:

(3) $a_j{}^i := [(form_1, lemma_1, tag_1, features_1),$
$(form_2, lemma_2, tag_2, features_2),$
$(form_3, lemma_3, tag_3, features_3)]$

We design a Morphological Embedding layer which acts as an interface between the *symbolic MA* and the *neural MD*. Figure 2 describes the encoding of a single morphological analysis into an embedded vector: Each property is embedded and averaged across all the morphemes in a single analysis, and all of the averaged embedded properties are concatenated to form a single embedded vector of a fixed size. The entire MA&D process is depicted in Figure 3.

4 Experimental Setup

The Data The PtrNetMD architecture we propose does not depend on any specific definition of morphological signature. To showcase this, we experiment with data from two different languages and two different annotation schemes. We use the Universal Dependencies v2.2 dataset (Nivre et al.,

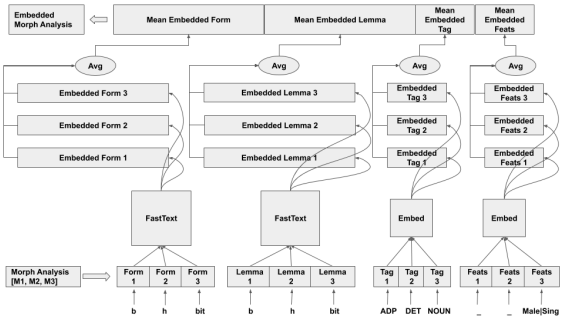

Figure 2: Morphological Embedding Layer Architecture. An analysis composed of 3 morphemes is transformed into a single embedded vector.

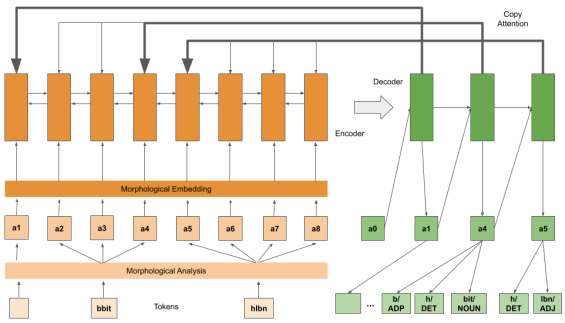

Figure 3: Our Proposed MA&D Architecture. A sequence of tokens is transformed into a sequence of analyses while preserving the token order. The sequence of analyses is embedded and fed into an encoder. Then at each decoding step the entire encoded representation along with the current decoded state are used as input to an attention layer, and the attention weights are used to choose an element from the input sequence.

2016) from the CoNLL18 UD Shared task.[3] In addition we download the corresponding lattice files of each treebank from the CoNLL-UL project.[4] Since our approach is sensitive to the lexical coverage of the MA lattices, we focus on the Hebrew (he_htb) and Turkish (tr_imst) treebanks. Unlike the other languages in the shared task, Hebrew and Turkish provided lattice files generated by broad-coverage analyzers (HEBLEX and TRMorph2).[5] For comparability with previous work on Modern Hebrew, we also train and test our model on the Hebrew SPMRL treebank standard split.[6]

[3]The UD treebanks from the shared task are available at lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2837

[4]https://conllul.github.io/

[5]The Arabic (ar_padt) Calima-Star lattice files exhibited a number of incompatibilities with the corresponding gold UD annotations and therefore cannot be considered

[6]The treebank is publicly available as open source at https://github.com/OnlpLab/HebrewResources/tree/master/HebrewTreebank

Lattice Embedding We use pre-trained FastText models to embed the forms and lemmas. FastText models generate vectors for any word using character ngrams, thus handling Out-of-Vocabulary forms and lemmas (Bojanowski et al., 2017). For POS tags and features we instantiate and train from scratch two embedding modules. Together, these 4 embedded properties are combined to produce a single morphological analysis vector.

Lattice Encoding The above-mentioned morphological embedding layer turns the input analysis sequence into an embedded sequence. The partially ordered sequence of embedded analyses is fed to an encoder layer thus encoding the entire lattice. Next a step-by-step decoding process begins in which a decoder is using an *Attention* mechanism in order to score the alignment between each of the relevant encoded analyses and the token currently being decoded. Our Copy Attention module is the *global dot-product* of Luong et al. (2015) using *masking* mechanism to make sure each decoding step is focused *only* on the corresponding input token analyses (in figure 3 the masks are represented by the grouped arrows pointing from the decoder back to the encoded sequence). The decoder chooses the highest scoring analysis. The full output sequence contains a list of indices, one per token, pointing to the selected analyses from the input lattice (Fig. 2).

4.1 Baseline Models

MD may be considered a special case of POS tagging, performed on the morpheme sequence. To compare our PtrNetMD architecture to existing modeling solutions we consider three baseline variations of POS tagging-based MD models implemented end-to-end, defined as follows.

Pipeline Straka and Straková (2017) approach the MD problem as a two-phased pipeline, first performing segmentation of the input tokens followed by sequence tagging on the morpheme sequence. This approach mimics the way English POS tagging is performed, with the exception that the tagging is done on the morphological forms as opposed to directly on input tokens. While it is straight forward to design, POS tagging accuracy suffers from error propagation from the earlier segmentation. We compare the tagging accuracy provided by gold (oracle) segments as opposed to realistically predicted segments, for Turkish, Hebrew, Arabic and English, to gauge the drop in the accuracy in English in comparison to MRLs.

Token sequence multi-tagging In order to avoid error propagation and train our neural model end-to-end, we implement a baseline model predicting a complex analysis, referred to as *multi-tag*, for each token. That is, we assign a single complex label composed of multiple POS tags to each raw token. We define a multi-tag as a concatenated list of basic tags, one per segment. In training, a word such as *bbit*, which is gold-segmented into the basic tag sequence *b/IN, h/DET, bit/NOUN*, is assigned a single multi-tag *bbit/IN-DET-NOUN*. Similar to the form and lemma embedding in the PtrNetMD we use FastText for embedding the input token sequence. In addition, in order to inform the model about sub-token information, we combined each embedded token with a vector encoding the sequence of characters in the token, as suggested by Ling et al. (2015). A notable disadvantage of this model compared to the pipeline, and the proposed PtrNet model, is that it does *not* provide any information concerning segmentation boundaries.

Sequence-to-sequence tagging Our multi-tagging model has the drawback of operating on a large and non-compositional output-labels space. So, it cannot assign previously unseen tag compositions to previously unseen tokens. To overcome this, we implement a sequence-to-sequence model in which the input again consists of raw input tokens but the output is a tag sequence, of a possibly different length, predicted (decoded) one by one. Here again we use the combined token and character embedding layer as described in the previous paragraph. This model too, does not provide explicit segmentation boundaries.

4.2 Evaluation

Aligned Segment The CoNLL18 UD Shared task evaluation campaign[7] reports scores for segmentation and POS tagging[8] for all participating languages. The shared task provides an evaluation script producing various levels of F1 scores, based on aligned token-level segments. Since the focus of the shared task was to reflect word segmentation and relations between content words, the script discards unmatched word segments, so in effect the POS tagging scores are in fact *joint segmentation-and-tagging*. We run this script to compare tagging scores between oracle (gold) segmentation and realistic (predicted) segmentation in a pipeline model. In addition, since our PtrNetMD jointly predicts both segments and tags, we can compare our PtrNetMD against the shared task leaders for Hebrew and Turkish.

Aligned Multi-Set In addition to the shared task scores, we compute F1 scores similar to the aforementioned with a slight but important difference. Token counts are based on multi-set intersections of the gold and predicted labels. A multi-set (mset) is a modification of the set concept, allowing multiple instances of its items. In our case we use a multi-set to count intersection of morphological signatures in each token. To illustrate the difference between *aligned segment* and *aligned mset*, let us take for example the gold segmented tag sequence: *b/IN, h/DET, bit/NOUN* and the predicted segmented tag sequence *b/IN, bit/NOUN*. According to *aligned segment*, the first segment (*b/IN*) is aligned and counted as a true positive, the second segment however is considered as a false positive (*bit/NOUN*) and false negative (*h/DET*) while the third gold segment is also counted as a false negative (*bit/NOUN*). The *aligned mset* on the other hand is based on set difference. In this case both *b/IN* and *bit/NOUN* exist in the gold and predicted sets and counted as true positives, while *h/DET* is mismatched and counted as a false negative. In both cases the total counts across the entire datasets are then incremented accordingly and finally used for computing Precision, Recall and F1.

Formally, *aligned mset* F1 metric is calculated as follows: For each token we first create a multi-set based on the morphological signatures (morphological signature is defined by the properties of interest: Segments only, POS tag only, joint segment and tag, etc.) for both the predicted (Pred) and gold (Gold) morphemes:

(4) $Pred_{\text{token}} = \uplus(p_1, p_2, ..., p_k)$
$Gold_{\text{token}} = \uplus(g_1, g_2, ..., g_l)$
$\uplus$: multi-set addition operator

We then calculate the token level true and false positives (TP, FP) as well as false negatives (FN):

(5) $TP_{\text{token}} = Pred_{\text{token}} \cap Gold_{\text{token}}$
$FP_{\text{token}} = Pred_{\text{token}} - Gold_{\text{token}}$
$FN_{\text{token}} = Gold_{\text{token}} - Pred_{\text{token}}$

Finally we add up the token counts over the entire dataset to produce the F1 metric:

[7] https://universaldependencies.org/conll18/results.html

[8] respectively referred to as 'Segmented Words' and 'UPOS' in the CoNLL18 evaluation script

(6) $TP_{\text{total}} = \sum |TP_{\text{token}}|$
$FP_{\text{total}} = \sum |FP_{\text{token}}|$
$FN_{\text{total}} = \sum |FN_{\text{token}}|$
$Precision = TP_{\text{total}}/(TP_{\text{total}} + FP_{\text{total}})$
$Recall = TP_{\text{total}}/(TP_{\text{total}} + FN_{\text{total}})$
$F1 = \frac{2 \times Precision \times Recall)}{Precision + Recall}$

Having morphemes available even if out of order or partially, has merit to downstream tasks that consume and further process them. *Aligned mset* accounts for this quality. Furthermore, both our *multi-tagging* and *sequence-to-sequence tagging* baseline models produce a tag sequence *without* segmentation boundaries, and *aligned mset* can be used to compare them against our PtrNetMD model. Finally since this computation was also used by More et al. (2019) we are able to compare our results to their non-neural MA&D framework applied to the Hebrew SPRML treebank, which is so far considered the current state-of-the-art for Hebrew segmentation and tagging.

Ideal vs Realistic Analysis Scenarios Following More et al. (2019) we distinguish between two evaluation scenarios. An **Infused** scenario is an idealised scenario in which the input lattice to our model has complete lexical coverage, and is guaranteed to include the correct analysis as one of its many internal paths. An **Uninfused** scenario is a realistic case in which the lexical coverage might be partial, and might lack certain gold analyses. [9]

5 Results

CoNLL18 UD Shared Task Table 2 shows *aligned segment* F1 scores for joint segmentation-and-tagging on four languages that exhibit different degrees of morphological richness. The top two models are variants of the UDPipe pipeline system — UDPipe *Oracle* scores were obtained by running the UDPipe tagger on gold segments, and UDPipe *Predicted* scores were obtained by segmenting the raw text first and then tagging the predicted segments.[10]

The top two rows in Table 2 allow us to gauge the effect of error propagation for different languages, as reflected in the performance difference between tagging gold (Oracle) segments and tagging predicted segments. These results are remarkable — in an *ideal* (gold-oracle) scenario there is no significant difference in the tagging accuracy between English and MRLs, but in the *realistic* scenarios where segmentation precedes tagging, the difference is large.

The bottom three models in Table 2 report the leading scores from the CoNLL18 UD Shared Task as well as our PtrNetMD results. The PtrNetMD achieves state-of-the-art results for joint segmentation-tagging, on both Hebrew and Turkish, in infused settings. Moreover, the PtrNetMD ties the state-of-the-art on the Hebrew treebank even with uninfused (realistic) lattices with partial lexical coverage.

In Table 3 we see *aligned segment* F1 scores for segmentation-only on the same four languages. The results clearly indicate that segmenting Hebrew is harder than segmenting Arabic, which is then harder to segment than Turkish, and English requires essentially no segmentation. As in Table 2, we see similar behavior comparing PtrNetMD to shared task leaders on the segmentation task — PtrNetMD with infused lattices outperforms the shared-task leader on Turkish, and it outperforms the shared-task leader in both infused and uninfused scenarios on Hebrew.

There are two possible explanations for prediction errors in uninfused scenarios. Either the cor-

	English	Turkish	Arabic	Hebrew
UDPipe Oracle	94.62	93.24	95.30	95.13
UDPipe Predicted	93.62	91.64	89.34	80.87
Shared Task Leader	95.94	94.78	93.63	91.36
PtrNetMD Infused		**96.6**		**94.41**
PtrNetMD Uninfused		89.54		91.3

Table 2: Joint Segmentation-and-Tagging F1, Aligned Segment, CoNLL18 UD Shared Task Test Set. Top two rows are pipeline baseline. Bottom three rows are PtrNetMD compared to shared task leaders.

	English	Turkish	Arabic	Hebrew
UDPipe Oracle	100.00	100.00	100.00	100.00
UDPipe Predicted	99.03	97.92	93.71	85.16
Shared Task Leader	99.26	97.92	96.81	93.98
PtrNetMD Infused		**99.41**		**96.36**
PtrNetMD Uninfused		97.78		94.74

Table 3: Segmentation-only F1, Aligned Segment, CoNLL18 UD Shared Task Test Set. Top two rows are pipeline baseline. Bottom three rows are PtrNetMD compared to shared task leaders.

[9] Like More et al. (2019) we refer to the idealized scenario as **infused** since we make sure the gold annotation is present in each token lattice or else we manually infuse it. The realistic scenario is thus referred to as **uninfused**.

[10] The UDPipe Predicted model served as the baseline model for the CoNLL18 UD Shared Task participants.

	Turkish	Arabic	Hebrew
Token Multi-Tag	92.57	94.2	93.82
Token Seq-Tag	92.77	95.05	93.75
PtrNetMD infused	**96.76**		**96.40**
PtrNetMD uninfused	90.01		94.02

Table 4: Tagging F1, Aligned MSet, CoNLL18 UD Shared Task Test Set

rect analysis (gold annotation) is part of the lattice but the model makes a wrong selection, or, the correct analysis is not in the lattice. Acknowledging the notable gap in Table 2 between PtrNetMD infused and uninfused scores on Turkish, we compared the number of prediction errors with the number of missing analyses in the uninfused lattices. Out of 1028 wrong predictions, 652 of them were also missing the correct analysis which accounts to 60% of the uninfused errors. Interestingly there is a 60% error reduction when moving to the infused lattices. The missing analyses could account for the difference between infused and uninfused scores. The same holds for Hebrew as well: out of 850 made, 330 do not have the correct analysis in the lattice, which is also very close to the difference between the infused and uninfused scores. Another insight into the coverage difference between the Turkish and Hebrew lattices is revealed by the fact that the average number of analyses per token is 2.6 for Turkish compared to 10 in Hebrew.

Table 4 contains the *aligned mset* scores of our two baselines, as well as the PtrNetMD infused and uninfused settings (since both baselines don't predict segments they are inapplicable for *aligned segment* evaluation). In both Turkish and Hebrew, the infused PtrNetMD performs much better than end-to-end tagging models. The Hebrew PtrNetMD even outperforms both baselines in uninfused circumstances. The high infused scores on both treebanks suggest that the PtrNetMD model is more than capable to select the correct analysis as long as one is present in the lattice. The difference between infused and uninfused scores highlight the importance of generating full coverage lattices by the MA component.

SPMRL Hebrew Treebank To put our results in context, Table 5 compares PtrNetMD on the Hebrew SPMRL treebank with the state of the art results of More et al. (2019), who used the same *aligned mset* scores for performing joint segmentation-and-tagging evaluation. The

	Dev-Inf	Dev-Uninf	Test-Inf	Test-Uninf
MoreMD	94.09	90.83	92.92	87.53
MoreMD-DEP	**95.49**	92.36	**93.92**	89.08
PtrNetMD	95.09	**93.9**	93.51	**90.49**

Table 5: Joint Segmentation-and-Tagging F1, Aligned MSet, Hebrew SPMRL treebank

MoreMD lattice disambiguation approach is similar to our PtrNetMD, albeit non-neural, using feature-based structured perceptron for disambiguation.

As can be seen in the table, the PtrNetMD outperforms the MoreMD model in all settings. The MoreMD-DEP model, jointly performs MD and dependency parsing, taking advantage of additional syntactic information that is predicted jointly with the segmentation and tags. The syntactic information contributes to the MD performance as can be seen in the Infused columns. However, our PtrNetMD handles incomplete morphological information better than MoreMD-DEP, as can be seen in the Uninfused columns.

6 Related Work

Initial work on MD viewed it as a special case of POS tagging and applied generative probabilistic frameworks such as *Hidden Markov Models* (Barhaim et al., 2008) as well as discriminative feature-based models (Sak et al., 2009; Lee et al., 2011; Bohnet et al., 2013; Habash et al., 2013). When used as input to parsing, Goldberg and Elhadad (2010) showed that consuming the predicted MD output of Adler and Elhadad (2006b) as input to *dependency parsing* significantly reduced parsing performance on Hebrew.

To address this error propagation inherent in the pipeline approach, More et al. (2019) and Seeker and Çetinoğlu (2015) proposed joint morpho-syntactic frameworks which enable interaction between the morphological and syntactic layers. While proving to be state-of-the-art for both MD and dependency parsing, on Hebrew and Turkish respectively, these solutions involved massive hand-crafted feature engineering.

MA&D on Arabic was addressed by Habash and Rambow (2005); Roth et al. (2008) using MA output and applying a set of classification and language models to make grammatical and lexical predictions. A ranking component then scored the analyses produced by the MA using a weighted sum of matched predicted features. Zalmout and

Habash (2017) presented a neural version of the above system using LSTM networks in several configurations and embedding levels to model the various morphological features and use them to score and rank the MA analyses. In addition, they incorporated features based on the space of possible analyses from the MA into the MD component. By enriching the input word embedding with these additional morphological features they increased MD accuracy drastically. This ranking technique requires building several models - language models to predict form and lemma and sequence labeling models to predict non-lexical features such as POS, gender, number etc. Our solution on the other hand involves a single model to score the joint analyses and choose the best one. In addition, our neural MD component is language agnostic and doesn't depend on any language-specific properties, and as a result can be easily applied to any language.

Yildiz et al. (2016) proposed a MA&D framework with a neural MD model, however their MD component was implemented as a binary classifier predicting whether or not a current property value is correct, and was trained in a semi-supervised fashion. Such simple topology is focused on predicting POS tags and morphological feature but is inappropriate for the general case that includes segmentation.

Most recently, Khalifa et al. (2020) provided further validation of the hypothesis that in low-resource settings, morphological analyzers help boost the performance of the full morphological disambiguation task. We support this claim as well with our results on Hebrew and Turkish, which are considered low-resource languages, at least in terms of the resources the UD treebank collection provides. In the same vein, incorporating symbolic morphological information in MRLs has long shown to improve NLP tasks; see for instance Marton et al. (2013) for the contribution of morphological knowledge on parsing quality on Arabic.

End-to-end neural modeling for word segmentation was addressed by Shao et al. (2018) who modeled segmentation as character-level sequence labeling, and applied it to the UD data collection. While improving the results averaged over the entire UD set, Hebrew and Arabic accuracy remained low. Wang et al. (2016) tackled the segmentation challenge by taking an unsupervised approach for learning segment boundaries, but did not address POS and morphological features assignments.

A pre-requisite for our proposed approach is the availabilty of a morphological analyzer (MA) component. Over the past years several MA resources have been published and are available for MA&D research. The CoNLL-UL project (More et al., 2018) provides static lattice files generated for the CoNLL18 UD shared task (Zeman et al., 2018). Other MA resources are available for specific languages, for example: HEBLEX (Adler and Elhadad, 2006a), TRMorph2 (Çağrı Çöltekin, 2014), and Calima-Star (Taji et al., 2018). To facilitate MA for the UD treebanks, Sagot (2018) produced a collection of multilingual lexicons in the CoNLL-UL format covering many of the UD languages. The Universal Morphology (UniMorph) project contains morphological data annotated in a canonical schema for many languages, which has been shown to improve, e.g., low-resource machine translation (Shearing et al., 2018).

Encoding complete lattices into vector representations was previously achieved by modifying the implementation of the LSTM cells to keep track of the history of multiple node children (Ladhak et al., 2016; Su et al., 2017; Sperber et al., 2017). More recently, Sperber et al. (2019) applied self-attention layers coupled with reachability masks and positional embedding to efficiently handle lattice inputs. All of these lattice-aware networks were applied to speech recognition tasks, where the segmentation of the input stream refers only to *overt* elements, with no *covert* elements as in morphology. In this work, in contrast, we cope with non-concatenative morphological phenomena where not all segments are covert. Finally, our system is simple to apply and easy to comprehend. In contrast with the non-trivial modification to the internals of the neural model, we parse and encode the lattice as a sequence to be fed into (any) existing neural components.

7 Conclusions and Future Work

In this work we addressed the challenge of morphological disambiguation for MRLs. We design a general framework that consumes lattice files and output a sequence of disambiguated morphemes, each containing the segmentation boundary, lemma, part-of-speech tag and morphological features. Our solution is language agnostic and we apply it on two different languages and two different annotation schemes. We show that access to symbolic morphological information aids the neural disam-

biguation model, compared to end-to-end strong baselines that only have access to the raw tokens.

We empirically evaluate our model using two evaluation methods. The CoNLL18 UD Shared Task evaluation, and a multi-set intersection-based evaluation, which is a more informative metric for downstream tasks that operate directly on morpheme sequences. In an ideal scenario, where full lexical coverage is assumed, our model outperformed the shared task leaders in the word segmentation task as well as the joint segmentation-and-tagging task, in both Turkish and Hebrew. Furthermore, we match the leading joint segmentation and tagging scores in realistic scenario with only partial lexical coverage on Hebrew. We further show superior performance of our model compared to previous models on the Hebrew SPMRL treebank.

This work motivates two future research directions. Our infused-vs-uninfused analysis suggests that most errors on uninfused lattices are due to partial MA coverage. Our disambiguation model proves to be very reliable in selecting the correct analysis, when available. It follows that a broad-coverage MA component may improve the overall quality of the disambiguation in realistic (uninfused) scenarios. This motivates learning to induce universal, high-recall, MA which is free to generate large lattices, and rather than focusing on precision, reward high recall. A second research path towards improving realistic partial-coverage (uninfused) lattices is by combining our morphologically-aware *Pointer Network* with an end-to-end model that operates on the raw token sequence.

Finally, we intend to extend this lattice-based architecture for complete *Joint Morpho-Syntactic* and *Morpho-Semantic* tasks. That is, in addition to morphological segmentation and tagging, the pointer network can be trained to predict span labels (as in NER), headedness relations (as in dependency parsing) and possibly more properties for the lattice arcs, so that these multiple layers of information may be *jointly* predicted as part of the lattice-disambiguation task.

Acknowledgements

We thank the BIU-NLP lab members for comments and discussion, and to four anonymous reviewers for their insightful remarks. This research is funded by grants from the Israeli Science Foundation (ISF grant 1739/26) and the European Research Council (ERC grant 677352), for which we are grateful.

References

Meni Adler and Michael Elhadad. 2006a. An unsupervised morpheme-based HMM for Hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 665–672, Sydney, Australia. Association for Computational Linguistics.

Meni Adler and Michael Elhadad. 2006b. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 665–672, Stroudsburg, PA, USA. Association for Computational Linguistics.

Roy Bar-haim, Khalil Sima'an, and Yoad Winter. 2008. Part-of-speech tagging of modern hebrew text. *Nat. Lang. Eng.*, 14(2):223–251.

Dan Bareket and Reut Tsarfaty. 2020. Neural modeling for named entities and morphology (nemo^2).

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. 2013. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Los Angeles, California. 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 of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan. Association for Computational Linguistics.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological analysis and disambiguation for dialectal Arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 426–432, Atlanta, Georgia. Association for Computational Linguistics.

Salam Khalifa, Nasser Zalmout, and Nizar Habash. 2020. Morphological analysis and disambiguation for Gulf Arabic: The interplay between resources

and methods. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3895–3904, Marseille, France. European Language Resources Association.

Stav Klein and Reut Tsarfaty. 2020. Getting the ##life out of living: How adequate are word-pieces for modelling complex morphology? In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, SIGMORPHON 2020, Online, July 10, 2020*, pages 204–209.

Faisal Ladhak, Ankur Gandhe, Markus Dreyer, Lambert Mathias, Ariya Rastrow, and Björn Hoffmeister. 2016. Latticernn: Recurrent neural networks over lattices. In *INTERSPEECH*.

John Lee, Jason Naradowsky, and David A. Smith. 2011. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 885–894, Portland, Oregon, USA. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Yuval Marton, Nizar Habash, and Owen Rambow. 2013. Dependency parsing of modern standard Arabic with lexical and inflectional features. *Computational Linguistics*, 39(1):161–194.

Amir More, Özlem Çetinoğlu, Çağrı Çöltekin, Nizar Habash, Benoît Sagot, Djamé Seddah, Dima Taji, and Reut Tsarfaty. 2018. CoNLL-UL: Universal morphological lattices for Universal Dependency parsing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC'18)*.

Amir More, Amit Seker, Victoria Basmova, and Reut Tsarfaty. 2019. Joint transition-based models for morpho-syntactic parsing: Parsing strategies for MRLs and a case study from modern Hebrew. *Transactions of the Association for Computational Linguistics*, 7:33–48.

Amir More and Reut Tsarfaty. 2016. Data-driven morphological analysis and disambiguation for morphologically rich languages and universal dependencies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 337–348, Osaka, Japan. The COLING 2016 Organizing Committee.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *Proceedings of ACL-08: HLT, Short Papers*, pages 117–120, Columbus, Ohio. Association for Computational Linguistics.

Benoît Sagot. 2018. A multilingual collection of CoNLL-u-compatible morphological lexicons. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2009. Morphological disambiguation of turkish text with perceptron algorithm. In *Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing '07, page 107–118, Berlin, Heidelberg. Springer-Verlag.

Wolfgang Seeker and Özlem Çetinoğlu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *Transactions of the Association for Computational Linguistics*, 3:359–373.

Yan Shao, Christian Hardmeier, and Joakim Nivre. 2018. Universal word segmentation: Implementation and interpretation. *Transactions of the Association for Computational Linguistics*, 6:421–435.

Steven Shearing, Christo Kirov, Huda Khayrallah, and David Yarowsky. 2018. Improving low resource machine translation using morphological glosses (non-archival extended abstract). In *Proceedings of the 13th Conference of the Association for Machine*

Translation in the Americas (Volume 1: Research Papers), pages 132–139, Boston, MA. Association for Machine Translation in the Americas.

K. Sima'any, A. Itaiz, Y. Winterz, A. Altmanz, and N. Nativx. 2001. Building a tree-bank of modern hebrew text. volume 42, pages 347–380. Traitement automatique des langues.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1380–1389, Copenhagen, Denmark. Association for Computational Linguistics.

Matthias Sperber, Graham Neubig, Ngoc-Quan Pham, and Alex Waibel. 2019. Self-attentional models for lattice inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1185–1197, Florence, Italy. Association for Computational Linguistics.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 3302–3308. AAAI Press.

Dima Taji, Salam Khalifa, Ossama Obeid, Fadhl Eryani, and Nizar Habash. 2018. An Arabic morphological analyzer and generator with copious features. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 140–150, Brussels, Belgium. Association for Computational Linguistics.

Reut Tsarfaty, Dan Bareket, Stav Klein, and Amit Seker. 2020. From SPMRL to NMRL: what did we learn (and unlearn) in a decade of parsing morphologically-rich languages (mrls)? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7396–7408.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kuebler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (SPMRL) what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12, Los Angeles, CA, USA. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window lstm neural networks. In *AAAI*.

Eray Yildiz, Caglar Tirkaz, H. Sahin, Mustafa Eren, and Omer Sonmez. 2016. A morphology-aware network for morphological disambiguation. In *AAAI Conference on Artificial Intelligence*.

Nasser Zalmout and Nizar Habash. 2017. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 704–713, Copenhagen, Denmark. Association for Computational Linguistics.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

Çağrı Çöltekin. 2014. A set of open source tools for turkish natural language processing. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).