Neural Modeling for Named Entities and Morphology (NEMO^2)

Dan Bareket[1,2] and **Reut Tsarfaty**[1]
[1]Bar Ilan University, Ramat-Gan, Israel
[2]Open Media and Information Lab (OMILab), The Open University of Israel, Israel
dbareket@gmail.com, reut.tsarfaty@biu.ac.il

Abstract

Named Entity Recognition (NER) is a fundamental NLP task, commonly formulated as classification over a sequence of tokens. Morphologically rich languages (MRLs) pose a challenge to this basic formulation, as the boundaries of named entities do not necessarily coincide with *token* boundaries, rather, they respect *morphological* boundaries. To address NER in MRLs we then need to answer two fundamental questions, namely, what are the basic *units* to be labeled, and how can these units be detected and classified in realistic settings (i.e., where no gold morphology is available). We empirically investigate these questions on a novel NER benchmark, with *parallel* token-level and morpheme-level NER annotations, which we develop for Modern Hebrew, a morphologically rich-and-ambiguous language. Our results show that explicitly modeling morphological boundaries leads to improved NER performance, and that a novel *hybrid* architecture, in which NER precedes and prunes morphological decomposition, greatly outperforms the standard *pipeline*, where morphological decomposition strictly precedes NER, setting a new performance bar for *both* Hebrew NER and Hebrew morphological decomposition tasks.

1 Introduction

Named Entity Recognition (NER) is a fundamental task in the area of *Information Extraction (IE)*, in which mentions of Named Entities (NE) are *extracted* and *classified* in naturally occurring texts. This task is most commonly formulated as a *sequence labeling* task, where *extraction* takes the form of assigning each input token with a label that marks the *boundaries* of the NE (e.g., B, I, O), and *classification* takes the form of assigning labels to indicate entity *type* (Per, Org, Loc, etc.).

Despite a common initial impression from latest NER performance, brought about by neural models on the main English NER benchmarks—CoNLL 2003 (Tjong Kim Sang, 2003) and OntoNotes (Weischedel et al., 2013)—the NER task in real-world settings is far from solved. Specifically, NER performance is shown to greatly diminish when moving to other domains (Luan et al., 2018; Song et al., 2018), when addressing the long tail of rare, unseen, and new user-generated entities (Derczynski et al., 2017), and when handling languages with fundamentally different structure than English. In particular, there is no readily available and empirically verified neural modeling strategy for neural NER in those languages with complex word-internal structure, also known as *morphologically rich languages*.

Morphologically rich languages (MRL) (Tsarfaty et al., 2010; Seddah et al., 2013; Tsarfaty et al., 2020) are languages in which substantial information concerning the arrangement of words into phrases and the relations between them is expressed at the word level, rather than in a fixed word-order or a rigid structure. The extended amount of information expressed at word-level and the morpho-phonological processes creating these words result in high token-internal complexity, which poses serious challenges to the basic formulation of NER as classification of raw, space-delimited, tokens. Specifically, while NER in English is formulated as the sequence labeling of *space-delimited tokens*, in MRLs a single token may include multiple meaning-bearing units, henceforth *morphemes*, only some of which are relevant for the entity mention at hand.

In this paper we formulate two questions concerning neural modeling strategies for NER in MRLs, namely: (i) What should be the granularity of the units to be labeled? Space-delimited tokens or finer-grain morphological segments? and, (ii) How can we effectively encode, and accurately detect, the morphological segments that are relevant to NER, and specifically in *realistic* settings, when gold morphological boundaries are not available?

To empirically investigate these questions we develop a novel *parallel* benchmark, containing

Transactions of the Association for Computational Linguistics, vol. 9, pp. 909–928, 2021. https://doi.org/10.1162/tacl_a_00404
Action Editor: Richard Sproat. Submission batch: 1/2021; Revision batch: 3/2021; Published 9/2021.

parallel token-level and morpheme-level NER annotations for texts in Modern Hebrew—a morphologically rich and morphologically ambiguous language, which is known to be notoriously hard to parse (More et al., 2019; Tsarfaty et al., 2019).

Our results show that morpheme-based NER is superior to token-based NER, which encourages a *segmentation-first* pipeline. At the same time, we demonstrate that token-based NER improves morphological segmentation in *realistic* scenarios, encouraging a *NER-first* pipeline. While these two findings may appear contradictory, we aim here to offer a resolution; a *hybrid* architecture where the token-based NER predictions *precede* and *prune* the space of morphological decomposition options, while the actual morpheme-based NER takes place only *after* the morphological decomposition. We empirically show that the *hybrid* architecture we propose outperforms all token-based and morpheme-based model variants of Hebrew NER on our benchmark, and it further outperforms all previously reported results on Hebrew NER and morphological decomposition. Our error analysis further demonstrates that morpheme-based models better *generalize*, that is, they contribute to recognizing the long tail of entities unseen during training (*out-of-vocabulary*, OOV), in particular those unseen entities that turn out to be composed of previously *seen* morphemes.

The contribution of this paper is thus manifold. First, we define key architectural questions for Neural NER modeling in MRLs and chart the space of modeling options. Second, we deliver a new and novel parallel benchmark that allows one to empirically *compare* and *contrast* the morpheme vs. token modeling strategies. Third, we show consistent advantages for *morpheme-based* NER, demonstrating the importance of morphologically aware modeling. Next we present a novel *hybrid* architecture which demonstrates an even further improved performance on both NER and morphological decomposition tasks. Our results for Hebrew present a new bar on these tasks, outperforming the reported state-of-the-art results on various benchmarks.[1]

2 Research Questions: NER for MRLs

In MRLs, words are internally complex, and word *boundaries* do not generally coincide with the boundaries of more basic meaning-bearing units. This fact has critical ramifications for sequence labeling tasks in MRLs in general, and for NER in MRLs in particular. Consider, for instance, the three-token Hebrew phrase in (1):[2]

(1) טסנו מתאילנד לסין

tasnu	*mithailand*	*lesin*
flew.1PL	from-Thailand	to-China

‘we flew from Thailand to China’

It is clear that תאילנד/thailand (*Thailand*) and סין/sin (*China*) are NEs, and in English, each NE is its own token. In the Hebrew phrase though, neither NE constitutes a single token. In either case, the NE occupies only one of two morphemes in the token, the other being a case-assigning preposition. This simple example demonstrates an extremely frequent phenomenon in MRLs such as Hebrew, Arabic, or Turkish, that the adequate boundaries for NEs do not coincide with token boundaries, and tokens must be segmented in order to obtain accurate NE boundaries.[3]

The segmentation of tokens and the identification of adequate NE boundaries is, however, far from trivial, due to complex morpho-phonological and orthographic processes in some MRLs (Vania et al., 2018; Klein and Tsarfaty, 2020). This means that the morphemes that compose NEs are not necessarily transparent in the character sequence of the raw tokens. Consider for example phrase (2):

(2) המרוץ לבית הלבן

hamerotz	*labayit*	*halavan*
the-race	to-house.DEF	the-white

‘the race to the White House’

Here, the full form of the NE הבית הלבן / *habayit halavan* (*the White House*), is not present in the utterances, only the sub-string בית הלבן / *bayit halavan* (*(the) White House*) is present in (2)—due to phonetic and orthographic processes suppressing the definite article ה/*ha* in certain environments. In this and many other cases, it is not only that NE boundaries do not coincide with

[1]Data & code: `https://github.com/OnlpLab/NEMO`.

[2]Glossing conventions are in accord with the Leipzig Glossing Rules (Comrie et al., 2008).

[3]We use the term *morphological segmentation* (or *segmentation*) to refer to splitting raw tokens into morphological segments, each carrying a single Part-Of-Speech tag. That is, we segment away prepositions, determiners, subordination markers and multiple kinds of pronominal clitics, that attach to their hosts via complex morpho-phonological processes. Throughout this work, we use the terms *morphological segment*, *morpheme*, or *segment* interchangeably.

token boundaries, they do not coincide with *characters* or *sub-strings* of the token either. This calls for accessing the more basic meaning-bearing units of the token, that is, to decompose the tokens into *morphemes*.

Unfortunately though, the morphological decomposition of surface tokens may be very challenging due to extreme morphological ambiguity. The sequence of morphemes composing a token is not always directly recoverable from its character sequence, and is not known in advance.[4] This means that for every raw space-delimited token, there are many conceivable readings which impose different segmentations, yielding different sets of potential NE boundaries. Consider for example the token לבני (*lbny*) in different contexts:

(3) (a) השרה לבני

hasara	*livni*
the-minister	$[\text{Livni}]_{PER}$

'Minister $[\text{Livni}]_{PER}$'

(b) לבני גנץ

le-beny	*gantz*
for-[Benny	$\text{Gantz}]_{PER}$

'for $[\text{Benny Gantz}]_{PER}$'

(c) לבני היקר

li-bni	*hayakar*
for-son.POSS.1SG	the-dear

'for my dear son'

(d) לבני חימר

livney	*kheymar*
brick.CS	clay

'clay bricks'

In (3a) the token לבני is completely consumed as a labeled NE. In (3b) לבני is only partly consumed by an NE, and in (3c) and (3d) the token is entirely out of an NE context. In (3c) the token is composed of several morphemes, and in (3d) it consists of a single morpheme. These are only some of the possible decompositions of this surface token, other alternatives may still be available. As shown by Goldberg and Tsarfaty (2008), Green and Manning (2010), Seeker and Çetinoğlu (2015), Habash and Rambow (2005), More et al., (2019), and others, the correct morphological decomposition becomes apparent only in the larger (syntactic or semantic) context. The challenge, in a nutshell, is as follows: In order to detect accurately NE boundaries, we need to segment the raw token first, however, in order to segment tokens *correctly*, we need to know the greater semantic content, including, for example, the participating entities. How can we break out of this apparent loop?

[4]This ambiguity gets magnified by the fact that Semitic languages that use abjads, like Hebrew and Arabic, lack capitalization altogether and suppress all vowels (diacritics).

Finally, MRLs are often characterized by an extremely sparse lexicon, consisting of a long-tail of OOV entities unseen during training (Czarnowska et al., 2019). Even in cases where all morphemes are present in the training data, morphological compositions of seen morphemes may yield tokens and entities which were unseen during training. Take for example the utterance in (4), which the reader may inspect as familiar:

(4) טסנו מסין לתאילנד

tasnu	*misin*	*lethailand*
flew.1PL	from-China	to-Thailand

'we flew from China to Thailand'

Example (4) is in fact example (1) with a switched flight direction. This subtle change creates two new surface tokens מסין, לתאילנד which might not have been seen during training, even if example (1) had been observed. Morphological compositions of an entity with prepositions, conjunctions, definite markers, possessive clitics and more, cause mentions of seen entities to have unfamiliar surface forms, which often fail to be accurately detected and analyzed.

Given the aforementioned complexities, in order to solve NER for MRLs we ought to answer the following fundamental modeling questions:

Q1. Units: What are the discrete units upon which we need to set NE boundaries in MRLs? Are they tokens? characters? morphemes? a representation containing multiple levels of granularity?

Q2. Architecture: When employing morphemes in NER, the classical approach is ''segmentation-first''. However, segmentation errors are detrimental and downstream NER cannot recover from them. How is it best to set up the pipeline so that segmentation and NER could interact?

Q3. Generalization: How do the different modeling choices affect NER generalization in MRLs? How can we address the long tail of OOV NEs in MRLs? Which modeling strategy best handles *pseudo*-OOV entities that result from a *previously unseen* composition of *already seen* morphemes?

3 Formalizing NER for MRLs

To answer the aforementioned questions, we chart and formalize the space of modeling options for neural NER in MRLs. We cast NER as a sequence labeling task and formalize it as $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $x \in \mathcal{X}$ is a sequence $x_1, \ldots, x_n$ of n discrete strings from some vocabulary $x_i \in \Sigma$, and $y \in \mathcal{Y}$ is a sequence $y_1, .., y_n$ of the same length, where $y_i \in Labels$, and $Labels$ is a finite set of labels composed of the BIOSE tags (i.e., BIOLU as described in Ratinov and Roth, 2009). Every non-O label is also enriched with an entity type label. Our list of types is presented in Table 2.

3.1 Token-Based or Morpheme-Based?

Our first modeling question concerns the discrete units upon which to set the NE boundaries. That is, what is the formal definition of the input vocabulary Σ for the sequence labeling task?

The simplest scenario, adopted in most NER studies, assumes token-based input, where each token admits a single label—hence *token-single*:

$$NER_{\textit{token-single}} : \mathcal{W} \rightarrow \mathcal{L}$$

Here, $\mathcal{W} = \{w^* | w \in \Sigma\}$ is the set of all possible token sequences in the language and $\mathcal{L} = \{l^* | l \in Labels\}$ is the set of all possible label sequences over the label set defined above. Each token gets assigned a single label, so the input and output sequences are of the same length. The drawback of this scenario is that since the input for *token-single* incorporates *no* morphological boundaries, the exact boundaries of the NEs remain underspecified. This case is exemplified at the top row of Table 1.

There is another conceivable scenario, where the input is again the sequence of space-delimited tokens, and the output consists of complex labels (henceforth *multi-labels*) reflecting, for each token, the labels of its constituent morphemes; henceforth, a *token-multi* scenario:

$$NER_{token-multi} : \mathcal{W} \rightarrow \mathcal{L}^*$$

Here, $\mathcal{W} = \{w^* | w \in \Sigma\}$ is the set of sequences of tokens as in *token-single*. Each *token* is assigned a multi-label, that is, a sequence ($l^* \in \mathcal{L}$) which indicates the labels of the token's morphemes in order. The output is a sequence of such multi-labels, one multi-label per token. This variant incorporates morphological information concerning the number and order of labeled morphemes, but lacks the precise morphological boundaries. This is illustrated at the middle of Table 1. A downstream application may require (possibly noisy) heuristics to determine the precise NE boundaries of each individual label in the multi-label for an input token.

Nickname	Input	Lit	Output
token-single	המרוץ	the-race	O
	לבית	to-house.DEF	B_ORG
	הלבן	the-white	E_ORG
token-multi	המרוץ	the-race	O + O
	לבית	to-house.DEF	O + B_ORG + I_ORG
	הלבן	the-white	I_ORG + E_ORG
morpheme	ה	the	O
	מרוץ	race	O
	ל	to	O
	ה	the	B_ORG
	בית	house	I_ORG
	ה	the	I_ORG
	לבן	white	E-ORG

Table 1: Input/output for *token-single*, *token-multi*, and *morpheme* models for example (2) in Section 2.

Another possible scenario is a *morpheme*-based scenario, assigning a label $l \in L$ for each segment:

$$NER_{morph} : \mathcal{M} \rightarrow \mathcal{L}$$

Here, $\mathcal{M} = \{m^* | m \in \textit{Morphemes}\}$ is the set of sequences of morphological segments in the language, and $\mathcal{L} = \{l^* | l \in \textit{Labels}\}$ is the set of label sequences as defined above. The upshot of this scenario is that NE boundaries are precise. An example is given in the bottom row of Table 1. But, since each token may contain many meaningful morphological segments, the length of the token sequence is not the same as the length of morphological segments to be labeled, and the model assumes prior morphological segmentation—which in realistic scenarios is not necessarily available.

3.2 Realistic Morphological Decomposition

A major caveat with *morpheme-based* modeling strategies is that they often assume an ideal scenario of *gold* morphological decomposition of the space-delimited tokens into morphological segments (cf. Nivre et al., 2007; Pradhan et al., 2012). But in reality, *gold morphological decomposition* is *not* known in advance, it has to be predicted automatically, and prediction errors may propagate to contaminate the downstream task.

Our second modeling question therefore concerns the interaction between the morphological decomposition and the NER tasks: How would it be best to set up the pipeline so that the prediction of the two tasks can interact?

To answer this, we define *morphological decomposition* as consisting of two subtasks: *morphological analysis* (MA) and *morphological disambiguation* (MD). We view sentence-based MA as:

$$MA : \mathcal{W} \rightarrow \mathcal{P}(\mathcal{M})$$

Here $\mathcal{W} = \{w^* | w \in \Sigma\}$ is the set of possible token sequences as before, $\mathcal{M} = \{m^* | m \in Morphemes\}$ is the set of possible morpheme sequences, and $\mathcal{P}(\mathcal{M})$ is the set of subsets of $\mathcal{M}$. The role of MA is then to assign a token sequence $w \in \mathcal{W}$ with all of its *possible* morphological decomposition options. We represent this set of alternatives in a dense structure that we call a *lattice* (exemplified in Figure 1). MD is the task of picking the single correct morphological path $M \in \mathcal{M}$ through the MA lattice of a given sentence:

$$MD : \mathcal{P}(\mathcal{M}) \rightarrow \mathcal{M}$$

Now, assume $x \in \mathcal{W}$ is a surface sentence in the language, with its morphological decomposition initially unknown and underspecified. In a *Standard* pipeline, MA strictly precedes MD:

$$MD_{\textit{Standard}} : M = MD(MA(x))$$

The main problem here is that MD errors may propagate to contaminate the NER output.

We propose a novel *Hybrid* alternative, in which we *inject* a task-specific signal, in this case NER,[5] to *constrain* the search for M through the lattice:

$$MD_{\textit{Hybrid}} : M = MD(MA(x) \restriction \textit{NER}_{\textit{token}}(x))$$

Here, the restriction $MA(x) \restriction NER(x)$ indicates *pruning* the lattice structure $MA(x)$ to contain *only* MD options that are *compatible* with the token-based NER predictions, and only then apply MD to the pruned lattice.

Both $MD_{Standard}$ and MD_{Hybrid} are disambiguation architectures that result in a morpheme sequence $M \in \mathcal{M}$. The latter benefits from the NER signal, while the former doesn't. The sequence $M \in \mathcal{M}$ can be used in one of two ways. We can use M as *input* to a *morpheme* model to output morpheme labels. Or, we can rely on the *output* of the *token-multi* model and align the token's multi-label with the segments in M.

In what follows, we want to empirically assess the effect of different modeling choices (*token-single*, *token-multi*, *morpheme*) and disambiguation architectures (*Standard, Hybrid*) on the performance of NER in MRLs. To this end, we need a corpus that allows training and evaluating NER at both token and morpheme-level granularity.

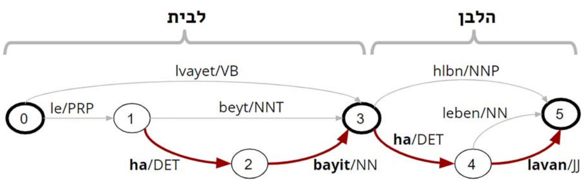

Figure 1: Lattice for a partial list of analyses of the Hebrew tokens לבית הלבן corresponding to Table 1. Bold nodes are token boundaries. Light nodes are segment boundaries. Every path through the lattice is a single morphological analysis. The bold path is a single NE.

4 The Data: A Novel NER Corpus

This work empirically investigates NER modeling strategies in Hebrew, a Semitic language known for its complex and highly ambiguous morphology. Ben-Mordecai (2005), the only previous work on Hebrew NER to date, annotated space-delimited tokens, basing their guidelines on the CoNLL 2003 shared task (Chinchor et al., 1999).

Popular Arabic NER corpora also label space-delimited tokens (ANERcorp [Benajiba et al., 2007], AQMAR [Mohit et al., 2012], TWEETS [Darwish, 2013]), with the exception of the Arabic portion of OntoNotes (Weischedel et al., 2013) and ACE (LDC, 2008) which annotate NER labels on gold morphologically pre-segmented texts. However, these works do not provide a comprehensive analysis on the performance gaps between *morpheme*-based and *token*-based scenarios.

In agglutinative languages as Turkish, token segmentation is *always* performed before NER (Tür et al., 2003; Küçük and Can, 2019, re-enforcing the need to contrast the *token*-based

[5]We can do this for any sequence labeling task in MRLs.

scenario, widely adopted for Semitic languages, with the *morpheme*-based scenarios in other MRLs.

Our first contribution is thus a *parallel* corpus for Hebrew NER; one version consists of gold-labeled tokens and the other consists of gold-labeled morphemes, for the same text. For this, we performed gold NE annotation of the Hebrew Treebank (Sima'an et al., 2001), based on the 6,143 morpho-syntactically analyzed sentences of the HAARETZ corpus, to create both token-level and morpheme-level variants, as illustrated at the topmost and lowest rows of Table 1, respectively.

Annotation Scheme We started off with the guidelines of Ben-Mordecai (2005), from which we deviate in three main ways. First, we label NE boundaries and their types on sequences of *morphemes*, in addition to the space-delimited token annotations.[6] Secondly, we use the finer-grained entity categories list of ACE (LDC, 2008).[7] Finally, we allow *nested* entity mentions, as in Finkel and Manning (2009) and Benikova et al. (2014).[8]

Annotation Cycle As Fort et al. (2009) put it, examples and rules would never cover all possible cases because of the specificity of natural language and the ambiguity of formulation. To address this we employed the cyclic approach of *agile* annotation as offered by Alex et al. (2010). Every cycle consisted of: *annotation*, *evaluation and curation*, *clarification and refinements*. We used WebAnno (Yimam et al., 2013) as our annotation interface.

The Initial Annotation Cycle was a two-stage pilot with 12 participants, divided into 2 teams of 6. The teams received the same guidelines, with the exception of the specifications of entity boundaries. One team was guided to annotate the minimal string that designates the entity. The other was guided to tag the maximal string which can still be considered as the entity. Our agreement analysis showed that the minimal guideline generally led to more consistent annotations. Based on this result (as well as low-level refinements) from the pilot, we devised the full version of the guidelines.[9]

Annotation, Evaluation, and Curation: Every annotation cycle was performed by two annotators (*A*, *B*) and an annotation manager/curator (*C*). We annotated the full corpus in 7 cycles. We evaluated the annotation in two ways, manual curation and automatic evaluation. After each annotation step, the curator manually reviewed every sentence in which disagreements arose, as well as specific points of difficulty pointed out by the annotators. The inter-annotator agreement metric described below was also used to quantitatively gauge the progress and quality of the annotation.

Clarifications and Refinements: In the end of each cycle we held a clarification talk between A, B, and C, in which issues that came up during the cycle were discussed. Following that talk we refined the guidelines and updated the annotators, which went on to the next cycle. In the end we performed a final curation run to make sentences from earlier cycles comply with later refinements.[10]

Inter-Annotator Agreement (IAA) IAA is commonly measured using the κ-statistic. However, Pyysalo et al. (2007) show that it is not suitable for evaluating inter-annotator agreement in NER. Instead, an F_1 metric on entity mentions has in recent years been adopted for this purpose (Zhang, 2013). This metric allows for computing pair-wise IAA using standard F_1 score by treating

	train	**dev**	**test**
Sentences	4,937	500	706
Tokens	93,504	8,531	12,619
Morphemes	127,031	11,301	16,828
All mentions	6,282	499	932
Type: Person (PER)	2,128	193	267
Type: Organization (ORG)	2,043	119	408
Type: Geo-Political (GPE)	1,377	121	195
Type: Location (LOC)	331	28	41
Type: Facility (FAC)	163	12	11
Type: Work-of-Art (WOA)	114	9	6
Type: Event (EVE)	57	12	0
Type: Product (DUC)	36	2	3
Type: Language (ANG)	33	3	1

Table 2: Basic corpus statistics. Standard HTB splits.

[6]A single NE is always continuous. Token-morpheme discrepancies do not lead to discontinuous NEs.

[7]Entity categories are listed in Table 2. We dropped the NORP category, since it introduced complexity concerning the distinction between adjectives and group names. LAW did not appear in our corpus.

[8]Nested labels are are not modeled in this paper, but they are published with the corpus, to allow for further research.

[9]The complete annotation guide is publicly available at `https://github.com/OnlpLab/NEMO-Corpus`.

[10]A, B, and C annotations are published to enable research on learning with disagreements (Plank et al., 2014).

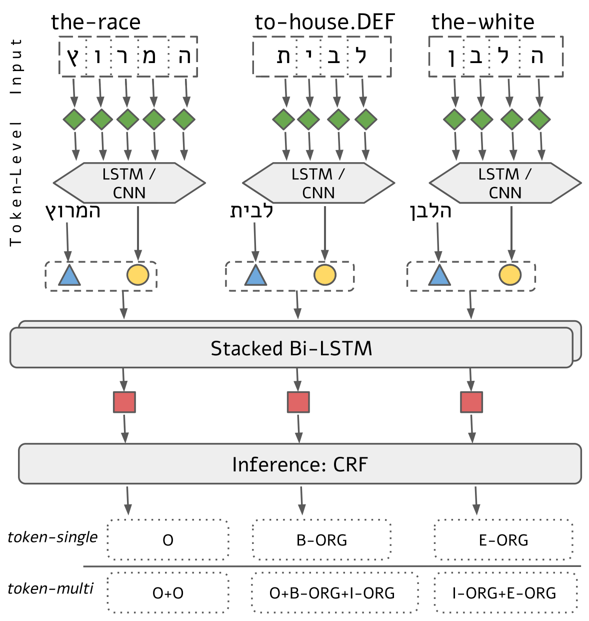

Figure 2: The *token-single* and *token-multi* models. The input and output correspond to rows 1,2 in Table 1. Triangles indicate *string* embeddings. Circles indicate *char-based* encoding.

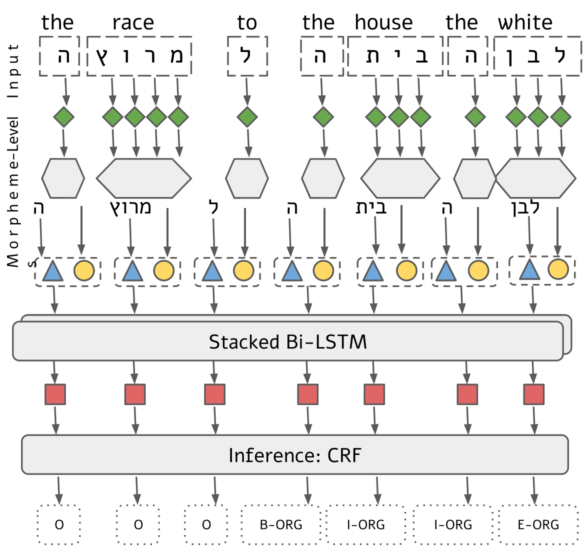

Figure 3: The *morpheme* model. The input and output correspond to row 3 in Table 1. Triangles indicate *string* embeddings. Circles indicate *char-based* encoding.

one annotator as gold and the other as the prediction.

Our full corpus pair-wise F_1 scores are: IAA(A,B)=89, IAA(B,C)=92, IAA(A,C)=96. Table 2 presents final corpus statistics.

Annotation Costs The annotation took on average about 35 seconds per sentence, and thus a total of 60 hours for all sentences in the corpus for each annotator. Six clarification talks were held between the cycles, which lasted from thirty minutes to an hour. Giving a total of about 130 work hours of expert annotators.[11]

5 Experimental Settings

Goal We set out to empirically evaluate the representation alternatives for the input/output sequences (*token-single*, *token-multi*, *morpheme*) and the effect of different architectures (*Standard*, *Hybrid*) on the performance of NER for Hebrew.

Modeling Variants All experiments use the corpus we just described and employ a standard Bi-LSTM-CRF architecture for implementing the neural sequence labeling task (Huang et al., 2015). Our basic architecture[12] is composed of an embedding layer for the input and a 2-layer Bi-LSTM followed by a CRF inference layer—for which we test three modeling variants.

Figures 2–3 present the variants we employ. Figure 2 shows the token-based variants, *token-single* and *token-multi*. The former outputs a single BIOSE label per token, and the latter outputs a multi-label per token—a concatenation of BIOSE labels of the morphemes composing the token. Figure 3 shows the *morpheme*-based variant for the same input phrase. It has the same basic architecture, but now the input consists of morphological segments instead of tokens. The model outputs a single BIOSE label for each morphological segment in the input.

In all modeling variants, the input may be encoded in two ways: (a) String-level embeddings (token-based or morpheme-based) optionally initialized with pre-trained embeddings. (b) Char-level embeddings, trained simultaneously with the main task (cf. Ma and Hovy, 2016; Chiu and Nichols, 2015; Lample et al., 2016). For char-based encoding (of either tokens or morphemes) we experiment with CharLSTM, CharCNN, or NoChar, that is, no character embedding at all.

We pre-trained all token-based or morpheme-based embeddings on the Hebrew Wikipedia dump of Goldberg (2014). For morpheme-based embeddings, we decompose the input using More et al. (2019), and use the morphological segments as the embedding units.[13] We compare

[11]The corpus is available at `https://github.com/OnlpLab/NEMO-Corpus`.

[12]Using the NCRF++ suite of Yang and Zhang (2018).

[13]Embeddings and Wikipedia corpus also available in: `https://github.com/OnlpLab/NEMO`.

Parameter	Value	Parameter	Value
Optimizer	SGD	*LR (*token-single*)	0.01
*Batch Size	8	*LR (*token-multi*)	0.005
LR decay	0.05	*LR (*morpheme*)	0.01
Epochs	200	Dropout	0.5
Bi-LSTM layers	2	*CharCNN window	7
*Word Emb Dim	300	Char Emb dim	30
Word Hidden Dim	200	*Char Hidden Dim	70

Table 3: Summary of hyper-parameter tuning. The * indicates divergence from the NCRF++ proposed setup and empirical findings (Yang and Zhang, 2018).

GloVe (Pennington et al., 2014) and fastText (Bojanowski et al., 2017). We hypothesize that since FastText uses sub-string information, it will be more useful for analyzing OOVs.

Hyper Parameters Following Reimers and Gurevych (2017) and Yang et al. (2018), we performed hyper-parameter tuning for each of our model variants. We performed hyper-parameter tuning on the dev set in a number of rounds of random search, independently on every input/output and char-embedding architecture. Table 3 shows our selected hyper-parameters.[14] The *Char CNN window size* is particularly interesting as it was not treated as a hyper-parameter in Reimers and Gurevych (2017), Yang et al. (2018). However, given the token-internal complexity in MRLs we conjecture that the window size over characters might make a crucial effect. In our experiments we found that a larger window (7) increased the performance. For MRLs, further research into this hyper-parameter might be of interest.

Evaluation Standard NER studies typically invoke the CoNLL evaluation script that anchors NEs in token *positions* (Tjong Kim Sang, 2003). However, it is inadequate for our purposes because we want to compare entities across token-based vs. morpheme-based settings. To this end, we use a revised evaluation procedure, which anchors the entity in its *form* rather than its *index*. Specifically, we report F_1 scores on strict, exact-match of the surface forms of the entity mentions. That is, the gold and predicted NE spans must exactly match in their form, boundaries, and entity type. In all experiments, we report both token-level F-scores and morpheme-level F-scores, for all models.

- **Token-Level Evaluation.** For the sake of backwards compatibility with previous work on Hebrew NER, we first define *token-level* evaluation. For *token-single* this is a straightforward calculation of F_1 against gold spans. For *token-multi* and *morpheme*, we need to map the predicted label sequence of that token to a single label, and we do so using linguistically informed rules we devise (as elaborated in Appendix A).[15]

- **Morpheme-Level Evaluation.** Our ultimate goal is to obtain *precise* boundaries of the NEs. Thus, our main metric evaluates NEs against the gold *morphological* boundaries. For *morpheme* and *token-single* models, this is a straightforward F_1 calculation against gold spans. Note for *token-single* we are expected to pay a price for boundary mismatch. For *token-multi*, we know the number and order of labels, so we align the labels in the multi-label of the token with the morphemes in its morphological decomposition.[16]

For all experiments and metrics, we report mean and confidence interval (0.95) over ten runs.

Input-Output Scenarios We experiment with two kinds of input settings: *token*-based, where the input consists of the sequence of space-delimited tokens, and *morpheme*-based, where the input consists of morphological segments. For the *morpheme* input, there are three input variants:

(i) *Morph-gold*: where the morphological sequence is produced by an expert (idealistic).

(ii) *Morph-standard*: where the morphological sequence is produced by a standard segmentation-first pipeline (realistic).

[14] A few interesting empirical observations diverging from those of Reimers and Gurevych (2017) and Yang et al. (2018) are worth mentioning. We found that a lower *Learning Rate* than the one recommended by Yang et al. (2018) (0.015), led to better results and less occurrences of divergence. We further found that raising the number of *Epochs* from 100 to 200 did not result in over-fitting, and significantly improved NER results. We used for evaluation the weights from the best epoch.

[15] In the *morpheme* case we might encounter ''illegal'' label sequences in case of a prediction error. We employ similar linguistically informed heuristics to recover from that (see Appendix A).

[16] In case of a misalignment (in the number of morphemes and labels) we match the label-morpheme pairs from the final one backwards, and pad unpaired morphemes with O labels.

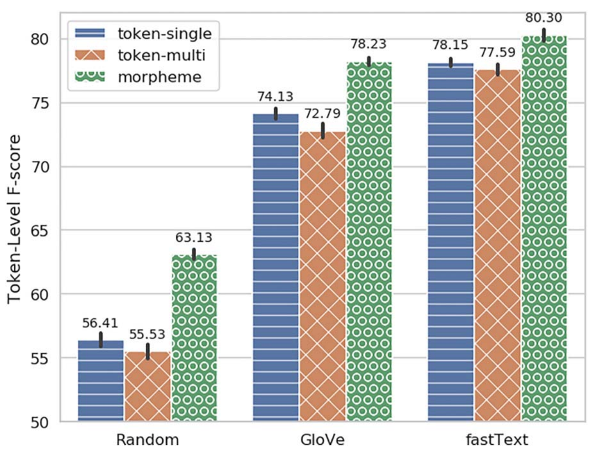

Figure 4: Token-level evaluation on dev w/ gold segmentation. CharCNN for morph, CharLSTM for tok.

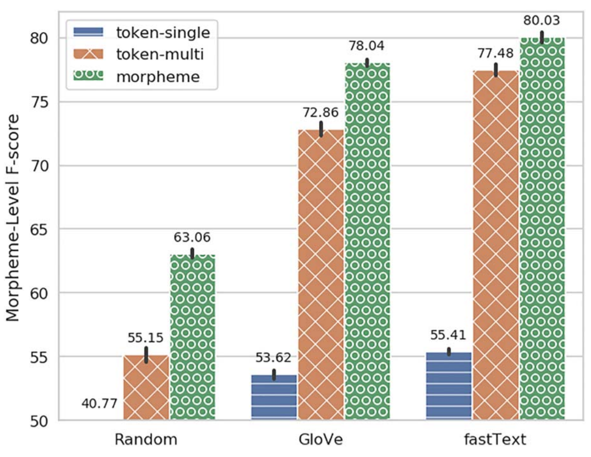

Figure 5: Morph-level evaluation on dev w/ gold segmentation. CharCNN for morph, CharLSTM for tok.

(iii) Morph-hybrid: where the morphological sequence is produced by the hybrid architecture we propose (realistic).

In the *token-multi* case we can perform *morpheme-based evaluation* by aligning individual labels in the multi-label with the morpheme sequence of the respective token. Again we have three options as to which morphemes to use:

(i) Tok-multi-gold: The multi-label is aligned with morphemes produced by an expert (idealistic).

(ii) Tok-multi-standard: The multi-label is aligned with morphemes produced by a standard pipeline (realistic).

(iii) Tok-multi-hybrid: The multi-label is aligned with morphemes produced by the hybrid architecture we propose (realistic).

Pipeline Scenarios Assume an input sentence x. In the *Standard* pipeline we use YAP,[17] the current state-of-the-art morpho-syntactic parser for Hebrew (More et al., 2019), for the predicted segmentation $M = MD(MA(x))$. In the *Hybrid* pipeline, we use YAP to first generate complete morphological lattices $MA(x)$. Then, to obtain $MA(x) \upharpoonright NER(x)$ we omit lattice paths where the number of morphemes in the token decomposition does not conform with the number of labels in the multi-label of $NER_{\textit{token-multi}}(x)$. Then, we apply YAP to obtain $MD(MA(x) \upharpoonright NER(x))$ on the constrained lattice. In predicted morphology scenarios (either *Standard* or *Hybrid*), we use the same model weights as trained on the gold segments, but feed predicted morphemes as input.[18]

6 Results

6.1 The Units: Tokens vs. Morphemes

Figure 4 shows the token-level evaluation for the different model variants we defined. We see that *morpheme* models perform significantly better than the *token-single* and *token-multi* variants. Interestingly, explicit modeling of morphemes leads to better NER performance even when evaluated against token-level boundaries. As expected, the performance gaps between variants are smaller with fastText than they are with embeddings that are unaware of characters (GloVe) or with no pre-training at all. We further pursue this in Section 6.3.

Figure 5 shows the morpheme-level evaluation for the same model variants as in Figure 4. The most obvious trend here is the drop in the performance of the *token-single* model. This is expected, reflecting the inadequacy of token boundaries for identifying accurate boundaries for NER. Interestingly, *morpheme* and *token-multi* models keep a similar level of performance as in token-level evaluation, only slightly lower. Their performance gap is also maintained, with *morpheme* performing better than *token-multi*. An obvious

[17]For other languages this may be done using models for canonical segmentation as in (Kann et al., 2016).

[18]We do not re-train the *morpheme* models with predicted segmentation, which might achieve better performance (e.g., jackknifing). We leave this for future work.

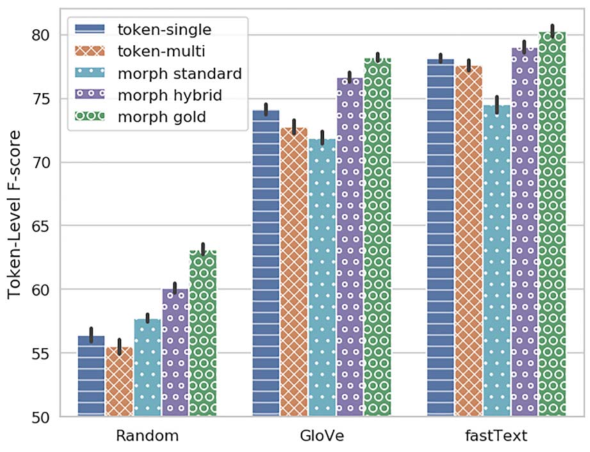

Figure 6: Token-level evaluation in realistic scenarios on dev, comparing *gold*, *standard*, and *hybrid* morphology. CharCNN for morph, CharLSTM for tok. Results for *Gold*, *token-single* and *token-multi* are taken from Figure 4.

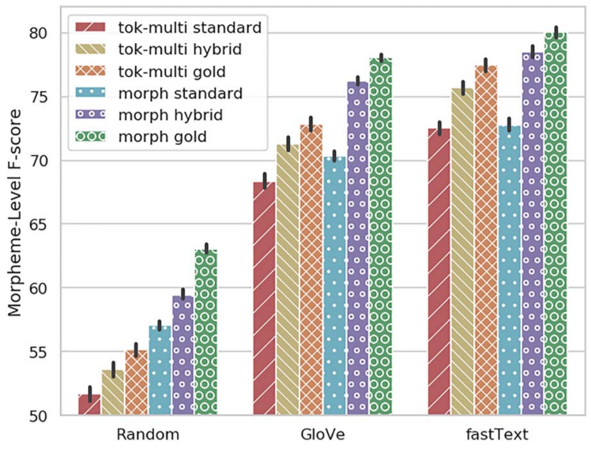

Figure 7: Morph-level evaluation in realistic scenarios on dev, comparing *gold*, *standard*, and *hybrid* morphology. CharCNN for morph, CharLSTM for tok. Results for *Gold*, *token-single* and *token-multi* are taken from Figure 5.

caveat is that these results are obtained with *gold* morphology. What happens in realistic scenarios?

6.2 The Architecture: Pipeline vs. Hybrid

Figure 6 shows the token-level evaluation results in realistic scenarios. We first observe a significant drop for *morpheme* models when *Standard* predicted segmentation is introduced instead of *gold*. This means that MD errors are indeed detrimental for the downstream task, in a non-negligible rate. Second, we observe that much of this performance gap is recovered with the *Hybrid* pipeline. It is noteworthy that while *morph hybrid* lags behind *morph gold*, it is still consistently better than token-based models, *token-single* and *token-multi*.

Figure 7 shows morpheme-level evaluation results for the same scenarios as in Table 6. All trends from the token-level evaluation persist, including a drop for all models with predicted segmentation relative to *gold*, with the *hybrid* variant recovering much of the gap. Again *morph gold* outperforms *token-multi*, but *morph hybrid* shows great advantages over *all tok-multi* variants. This performance gap between *morph* (*gold* or *hybrid*) and *tok-multi* indicates that *explicit* morphological modeling is indeed crucial for accurate NER.

6.3 Morphologically Aware OOV Evaluation

As discussed in Section 2, morphological composition introduces an extremely sparse word-level ‘‘long-tail’’ in MRLs. In order to gauge this phenomenon and its effects on NER performance, we categorize unseen, out-of-training-vocabulary (OOTV) mentions into 3 categories:

- *Lexical*: Unknown mentions caused by an unknown token which consists of a single morpheme. This is a strictly lexical unknown with no morphological composition (most English unknowns are in this category).
- *Compositional*: Unknown mentions caused by an unknown token which consists of multiple *known* morphemes. These are unknowns introduced strictly by morphological composition, with no lexical unknowns.
- *LexComp*: Unknown mentions caused by an unknown token consisting of multiple morphemes, of which (at least) one morpheme was not seen during training. In such cases, both unknown morphological composition *and* lexical unknowns are involved.

We group NEs based on these categories, and evaluate each group separately. We consider mentions that do not fall into any category as *Known*.

Figure 8 shows the distributions of entity mentions in the dev set by entity type and OOTV category. OOTV categories that involve composition (*Comp* and *LexComp*) are spread across all categories but one, and in some they even make up more than half of all mentions.

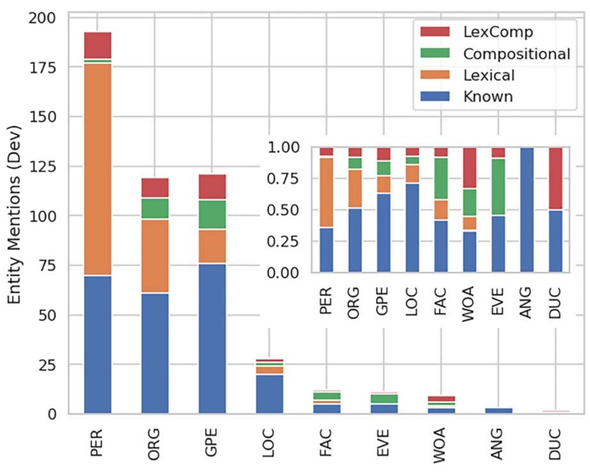

Figure 8: Entity mention counts and ratio by category and OOTV category, for dev set.

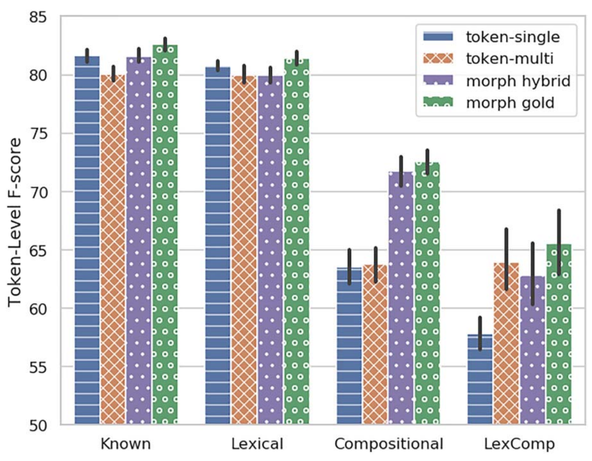

Figure 9: Token-level evaluation on dev by OOTV category. Using fastText and CharLSTM.

Figure 9 shows token-level evaluation[19] with fastText embeddings, grouped by OOTV type. We first observe that indeed unknown NEs that are due to morphological composition (*Comp* and *LexComp*) proved the most challenging for all models. We also find that in strictly *Compositional* OOTV mentions, *morpheme*-based models exhibit their most significant performance advantage, supporting the hypothesis that explicit morphology helps to *generalize*. We finally observe that *token-multi* models perform better than *token-single* models for these NEs (in contrast with the trend for non-compositional NEs). This corroborates the hypothesis that even partial modeling of morphology (as in *token-multi* compared to *token-single*) is better than none, leading to better generalization.

String-level vs. Character-level Embeddings To further understand the generalization capacity of different modeling alternatives in MRLs, we probe into the interplay of string-based and char-based embeddings in treating OOTV NEs.

Figure 10 presents 12 plots, each of which presents the level of performance (y-axes) for all models (x-axes). Token-based models are on the left of each x-axis, morpheme-based are on the right. We plot results with and without character embeddings,[20] in orange and blue, respectively. The plots are organized in a large grid, with the type of NE on the y-axes (*Known*, *Lex*, *Comp*, *LexComp*), and the type of pre-training on the x-axes (No pre-training, GloVe, fastText).

At the top-most row, plotting the accuracy for *Known* NEs, we see a high level of performance for all pre-training methods, with not much differences between the type of pre-training, with or without the character embeddings. Moving further down to the row of *Lexical* unseen NEs, char-based representations lead to significant advantages when we assume no pre-training, but with GloVe pre-training the performance substantially increases, and with fastText the differences in performance with/without char-embeddings almost entirely diminish, indicating the char-based embeddings are somewhat redundant in this case.

The two lower rows in the large grid show the performance for *Comp* and *LexComp* unseen NEs, which are ubiquitous in MRLs. For *Compositional* NEs, pre-training closes only part of the gap between token-based and morpheme-based models. Adding char-based representations indeed helps the token-based models, but crucially does *not* close the gap with the morpheme-based variants.

Finally, for *LexComp* NEs at the lowest row, we again see that adding GloVe pre-training and char-based embeddings does not close the gap with morpheme-based models, indicating that not all morphological information is captured by these vectors. For fastText with char-based embeddings the gap between *token-multi* and *morpheme* greatly diminishes, but is still well above *token-single*. This suggests biasing the model to

[19]This section focuses on token-level evaluation, which is a permissive evaluation metric, allowing us to compare the models on a more level playing field, where all models (including *token-single*) have an equal opportunity to perform.

[20]For brevity we only show char LSTM (vs. no char representation), there was no significant difference with CNN.

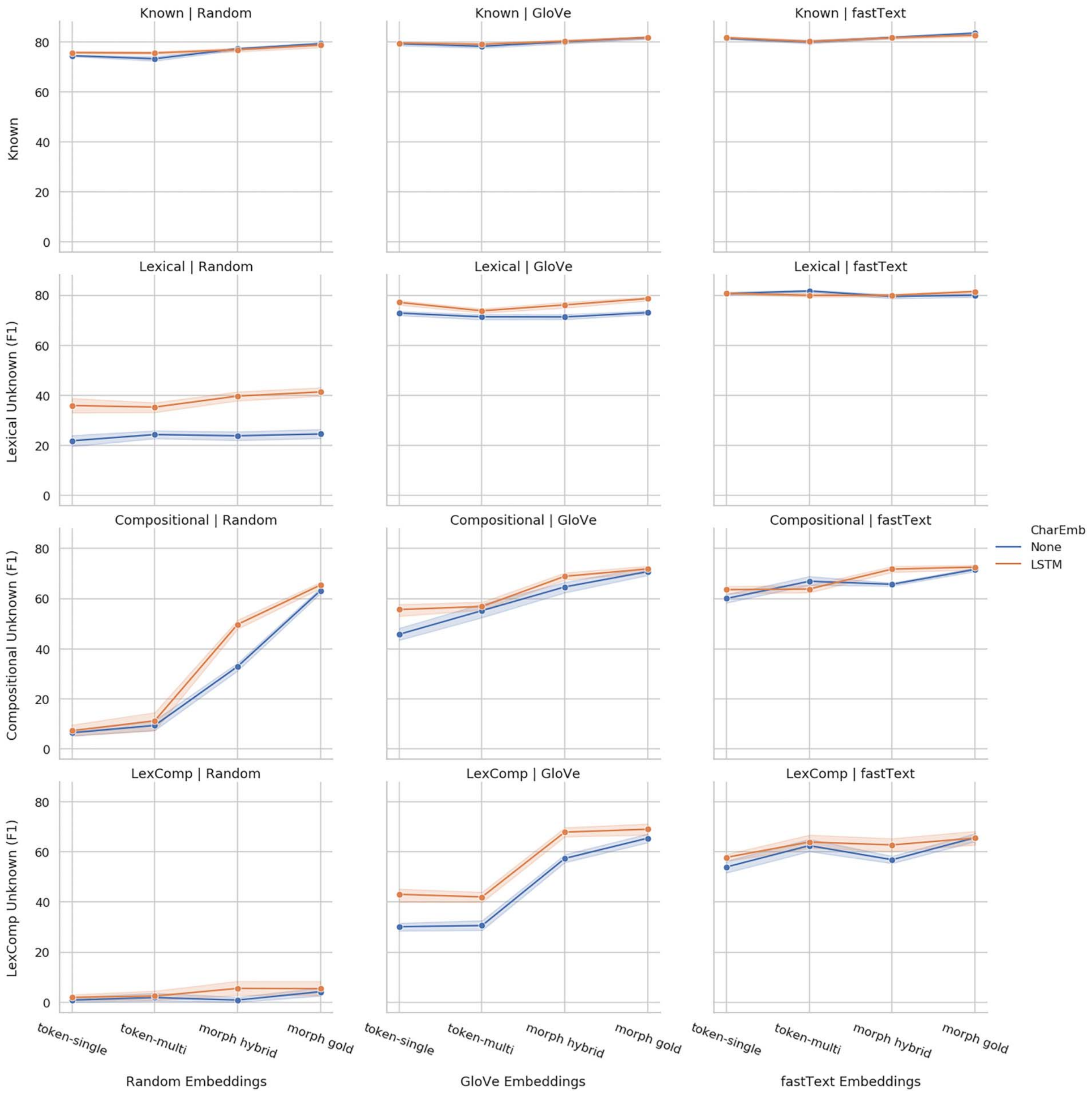

Figure 10: Token-level evaluation on dev for different OOTV types, char- and word-embeddings.

learn about morphology (either via multi-labels or by incorporating morphological boundaries) has advantages for analysing OOTV entities, beyond the contribution of char-based embeddings alone.

All in all, the biggest advantage of morpheme-based models over token-based models is their ability to generalize from observed tokens to composition-related OOTV (*Comp/LexComp*). While character-based embeddings do help token-based models generalize, the contribution of modeling morphology is indispensable, above and beyond the contribution of char-based embeddings.

6.4 Setting in the Greater Context

Test Set Results Table 4 confirms our best results on the Test set.The trends are kept, though results on Test are lower than on Dev. The *morph gold* scenario still provides an upperbound of the performance, but it is not realistic. For the realistic scenarios, *morph hybrid* generally outperforms all other alternatives. The only divergence is that in token-level evaluation, *token-multi* performs on a par with *morph hybrid* on the Test set.

Results on MD Tasks. While the *Hybrid* pipeline achieves superior performance on NER, it also improves the state-of-the-art on other tasks

Eval	Model	dev	test
Morph-Level	*morph gold*	*80.03 ± 0.4*	*79.10 ± 0.6*
	morph hybrid	**78.51** ± 0.5	**77.11** ± 0.7
	morph standard	72.79 ± 0.5	69.52 ± 0.6
	token-multi hybrid	75.70 ± 0.5	74.64 ± 0.3
Token-Level	*morph gold*	*80.30 ± 0.5*	*79.28 ± 0.6*
	morph hybrid	**79.04** ± 0.5	77.64 ± 0.7
	morph standard	74.52 ± 0.7	73.53 ± 0.8
	token-multi	77.59 ± 0.4	**77.75** ± 0.3
	token-single	78.15 ± 0.3	77.15 ± 0.6

Table 4: Test vs. dev: Results with fastText for all Models. *morph-gold* presents an ideal upper-bound.

in the pipeline. Table 5 shows the Seg+POS results of our *Hybrid* pipeline scenario, compared with the *Standard* pipeline which replicates the pipeline of More et al. (2019). We use the metrics defined by More et al. (2019). We show substantial improvements for the *Hybrid* pipeline over the results of More et al. (2019), and also outperforming the Test results of Seker and Tsarfaty (2020).

Comparison with Prior Art. Table 6 presents our results on the Hebrew NER corpus of Ben-Mordecai (2005) compared to their model, which uses a hand-crafted feature-engineered MEMM with regular-expression rule-based enhancements and an entity lexicon. Like Ben-Mordecai (2005) we performed three 75%-25% random train/test splits, and used the same seven NE categories (PER,LOC,ORG,TIME,DATE,PERCENT,MONEY). We trained a *token-single* model on the original space-delimited tokens and a *morpheme* model on automatically segmented morphemes we obtained using our best segmentation model (*Hybrid* MD on our trained *token-multi* model, as in Table 5). Since their annotation includes only token-level boundaries, all of the results we report conform with token-level evaluation.

Table 6 presents the results of these experiments. Both models significantly outperform the previous state-of-the-art by Ben-Mordecai (2005), setting a new performance bar on this earlier benchmark. Moreover, we again observe an empirical advantage when explicitly modeling morphemes, *even* with the automatic noisy segmentation that is used for the morpheme-based training.

		Seg+POS
dev	*Standard* (More et al., 2019)	92.36
	Ptr-Network (Seker and Tsarfaty, 2020)	**93.90**
	Hybrid (This work)	93.12
test	*Standard* (More et al., 2019)	89.08
	Ptr-Network (Seker and Tsarfaty, 2020)	90.49
	Hybrid (This work)	**90.89**

Table 5: Morphological segmentation & POS scores.

	Precision	Recall	F1
Ben-Mordecai (2005) MEMM+HMM+REGEX	84.54	74.31	79.10
This work *token-single*+FT+CharLSTM	86.84 ±0.5	82.6 ±0.9	84.71 ±0.5
This work *morph-Hybrid*+FT+CharLSTM	**86.93** ±0.6	**83.59** ±0.8	**85.22** ±0.5

Table 6: NER comparison with Ben-Mordecai (2005).

7 Discussion: Joint Modeling Alternatives and Future Work

The present study provides the motivation and the necessary foundations for comparing morpheme-based and token-based modeling for NER. While our findings clearly demonstrate the advantages of morpheme-based modeling for NER in a morphologically rich language, it is clear that our proposed *Hybrid* architecture is not the only modeling alternative for linking NER and morphology.

For example, a previous study by Güngör et al. (2018) addresses *joint* neural modeling of morphological segmentation and NER labeling, proposing a multi-task learning approach for joint MD and NER in Turkish. They employ separate Bi-LSTM networks for the MD and NER tasks, with a shared loss to allow for joint learning. Their results indicate improved NER performance, with no improvement in the MD results. Contrary to our proposal, they view MD and NER as distinct tasks, assuming a single NER label per token, and not providing disambiguated morpheme-level boundaries for the NER task. More generally, they test only *token-based* NER labeling and do not attend to the question of input/output granularity in their models.

A different approach for joint NER and morphology is jointly predicting the segmentation and labels for each token in the input stream. This is the approach taken, for instance, by the lattice-based Pointer-Network of Seker and Tsarfaty (2020). As shown in Table 5, their results for morphological segmentation and POS tagging are on a par with our reported results and, at least in principle, it should be possible to extend the Seker and Tsarfaty (2020) approach to yield also NER predictions.

However, our preliminary experiments with a lattice-based Pointer-network for token segmentation and NER labeling shows that this is not a straightforward task. Contrary to POS tags, which are constrained by the MA, every NER label can potentially go with any segment, and this leads to a combinatorial explosion of the search space represented by the lattice. As a result, the NER predictions are brittle to learn, and the complexity of the resulting model is computationally prohibitive.

A different approach to joint sequence segmentation and labeling can be applying the neural model directly on the character-sequence of the input stream. Such an approach is for instance the char-based *labeling as segmentation* setup proposed by Shao et al. (2017). Shao et al. use a character-based Bi-RNN-CRF to output a single label-per-char which indicates both word boundary (using BIES sequence labels) and the POS tags. This method is also used in their universal segmentation paper, (Shao et al., 2018). However, as seen in the results of Shao et al. (2018), char-based labeling for segmenting Semitic languages lags far behind all other languages, precisely because morphological boundaries are not explicit in the character sequences.

Additional proposals are those of Kong et al. (2015); Kemos et al. (2019). First, Kong et al. (2015) proposed to solve, for example, Chinese segmentation and POS tagging using dynamic programming with neural encoding, by using a Bi-LSTM to encode the character input, and then feeding it to a semi-Markov CRF to obtain probabilities for the different segmentation options. Kemos et al. (2019) propose an approach similar to Kong et al. (2015) for joint segmentation and tagging but add convolution layers on top of the Bi-LSTM encodings to obtain segment features hierarchically and then feed them to the semi-Markov CRF.

Preliminary experiments we conducted confirm that char-based joint segmentation and NER labeling for Hebrew, either using char-based labeling or a seq2seq architecture, still lags behind our reported results. We conjecture that this is due to the complex morpho-phonological and orthographic processed in Semitic languages. Going into char-based modeling nuances and offering a sound joint solution for a language like Hebrew is an important matter that merits its own investigation. Such work is feasible now given the new corpus, however, it is out of the scope of the current study.

All in all, the design of sophisticated joint modeling strategies for morpheme-based NER poses fascinating questions—for which our work provides a solid foundation (data, protocols, metrics, strong baselines). More work is needed for investigating joint modeling of NER and morphology, in the directions portrayed in this section, yet it is beyond the scope of this paper, and we leave this investigation for future work.

Finally, while the joint approach is appealing, we argue that the elegance of our *Hybrid* solution is precisely in providing a clear and well-defined interface between MD and NER through which the two tasks can interact, while still keeping the distinct models simple, robust, and efficiently trainable. It also has the advantage of allowing us to seamlessly integrate sequence labelling with any lattice-based MA, in a plug-and-play language-agnostic fashion, towards obtaining further advantages on both of these tasks.

8 Conclusion

This work addresses the modeling challenges of neural NER in MRLs. We deliver a parallel *token-vs-morpheme* NER corpus for Modern Hebrew, that allows one to assess NER modeling strategies in morphologically rich-and-ambiguous environments. Our experiments show that while NER benefits from morphological decomposition, downstream results are sensitive to segmentation errors. We thus propose a *Hybrid* architecture in which NER *precedes* and *prunes* the morphological decomposition. This approach greatly outperforms a *Standard* pipeline in realistic (non-gold) scenarios. Our analysis further shows that morpheme-based models better recognize OOVs that result from morphological composition. All in all we deliver new state-of-the-art results for Hebrew NER and MD, along with a novel benchmark, to encourage further investigation into the interaction between NER and morphology.

Acknowledgments

We are grateful to the BIU-NLP lab members as well as 6 anonymous reviewers for their insightful remarks. We further thank Daphna Amit and Zef Segal for their meticulous annotation and profound discussions. This research is funded by an ISF Individual Grant (1739/26) and an ERC Starting Grant (677352), for which we are grateful.

References

Bea Alex, Claire Grover, Rongzhou Shen, and Mijail Kabadjov. 2010. Agile corpus annotation in practice: An overview of manual and automatic annotation of CVs. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 29–37, Uppsala, Sweden. Association for Computational Linguistics.

Naama Ben-Mordecai. 2005. Hebrew Named Entity Recognition. Master's thesis, Department of Computer Science, Ben-Gurion University. `https://doi.org/10.1007/978-3-540-70939-8_13`

Yassine Benajiba, Paolo Rosso, and José Miguel BenedíRuiz. 2007. Anersys: An Arabic named entity recognition system based on maximum entropy. In *Computational Linguistics and Intelligent Text Processing*, pages 143–153, Berlin, Heidelberg. Springer Berlin Heidelberg.

Darina Benikova, Chris Biemann, and Marc Reznicek. 2014. NoSta-D named entity annotation for German: Guidelines and dataset. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 2524–2531, Reykjavik, Iceland. European Languages Resources Association (ELRA).

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. `https://doi.org/10.1162/tacl_a_00051`

N. Chinchor, E. Brown, L. Ferro, and P. Robinson. 1999. Named entity recognition task definition. The MITRE Corporation and SAIC.

Jason P. C. Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional LSTM-CNNS. *CoRR*, abs/1511.08308.

Bernard Comrie, Martin Haspelmath, and Balthasar Bickel. 2008. The Leipzig glossing rules: Conventions for interlinear morpheme-by-morpheme glosses. *Department of Linguistics of the Max Planck Institute for Evolutionary Anthropology & the Department of Linguistics of the University of Leipzig*.

Paula Czarnowska, Sebastian Ruder, Edouard Grave, Ryan Cotterell, and Ann Copestake. 2019. Don't forget the long tail! A comprehensive analysis of morphological generalization in bilingual lexicon induction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 974–983, Hong Kong, China. Association for Computational Linguistics. `https://doi.org/10.18653/v1/D19-1090`

Kareem Darwish. 2013. Named entity recognition using cross-lingual resources: Arabic as an example. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1558–1567, Sofia, Bulgaria. Association for Computational Linguistics.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics. `https://doi.org/10.18653/v1/W17-4418`

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 141–150, Stroudsburg, PA, USA. Association for Computational Linguistics. `https://doi.org/10.3115/1699510.1699529`

Karën Fort, Maud Ehrmann, and Adeline Nazarenko. 2009. Towards a methodology for named entities annotation. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 142–145, Suntec, Singapore. Association for Computational Linguistics. https://doi.org/10.3115/1698381.1698406

Yoav Goldberg. 2014. Hebrew Wikipedia dependency parsed corpus, v.1.0.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL-08: HLT*, pages 371–379, Columbus, Ohio. Association for Computational Linguistics.

Spence Green and Christopher D. Manning. 2010. Better Arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 394–402, Beijing, China. Coling 2010 Organizing Committee.

Onur Güngör, Suzan Üsküdarli, and Tunga Güngör. 2018. Improving named entity recognition by jointly learning to disambiguate morphological tags. *CoRR*, abs/1807.06683.

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. https://doi.org/10.3115/1219840.1219911

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 961–967, Austin, Texas. Association for Computational Linguistics.

Apostolos Kemos, Heike Adel, and Hinrich Schütze. 2019. Neural semi-Markov conditional random fields for robust character-based part-of-speech tagging. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2736–2743, Minneapolis, Minnesota. Association for Computational Linguistics.

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*, pages 204–209, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.sigmorphon-1.24

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2015. Segmental recurrent neural networks. *arXiv preprint arXiv:1511.06018*.

Dilek Küçük and Fazli Can. 2019. A tweet dataset annotated for named entity recognition and stance detection. *CoRR*, abs/1901.04787.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

LDC. 2008. ACE (automatic content extraction) english annotation guidelines for entities version 6.6.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1360

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNS-CRF. *CoRR*, abs/1603.01354.

Behrang Mohit, Nathan Schneider, Rishav Bhowmick, Kemal Oflazer, and Noah A. Smith. 2012. Recall-oriented learning of named entities in Arabic Wikipedia. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 162–173, Avignon, France. Association for Computational Linguistics.

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. https://doi.org/10.1162/tacl_a_00253

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 915–932, Prague, Czech Republic. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 532–1543, Doha, Qatar. Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014. Learning part-of-speech taggers with inter-annotator agreement loss. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 742–751, Gothenburg, Sweden. Association for Computational Linguistics. https://doi.org/10.3115/v1/E14-1078

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007. Bioinfer: A corpus for information extraction in the biomedical domain. *BMC bioinformatics*, 8(1):50. https://doi.org/10.1186/1471-2105-8-50

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics. https://doi.org/10.3115/1596374.1596399

Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. *CoRR*, abs/1707.06799.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA. Association for Computational Linguistics.

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. https://doi.org/10.1162/tacl_a_00144

Amit Seker and Reut Tsarfaty. 2020. A pointer network architecture for joint morphological segmentation and tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4368–4378, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.391

Yan Shao, Christian Hardmeier, and Joakim Nivre. 2018. Universal word segmentation: Implementation and interpretation. *Transactions of the Association for Computational Linguistics*, 6:421–435. https://doi.org/10.1162/tacl_a_00033

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. Character-based joint segmentation and POS tagging for Chinese using bidirectional RNN-CRF. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 173–183, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. 2001. Building a treebank of modern Hebrew text. *Traitement Automatique des Langues*, 42(2):347–380.

Hye-Jeong Song, Byeong-Cheol Jo, Chan-Young Park, Jong-Dae Kim, and Yu-Seop Kim. 2018. Comparison of named entity recognition methodologies in biomedical documents. *Biomedical Engineering Online*, 17(2):158. https://doi.org/10.1186/s12938-018-0573-6

Erik F. Tjong Kim Sang. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics. https://doi.org/10.3115/1119176.1119195

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*, pages 7396–7408, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.660

Reut Tsarfaty, Shoval Sadde, Stav Klein, and Amit Seker. 2019. What's wrong with Hebrew NLP? And how to make it right. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 259–264, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-3044

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.

Gökhan Tür, Dilek Hakkani-tür, and Kemal Oflazer. 2003. A statistical information extraction system for turkish. *Natural Language Engineering*, 9(2):181–210. https://doi.org/10.1017/S135132490200284X

Clara Vania, Andreas Grivas, and Adam Lopez. 2018. What do character-level models learn about morphology? The case of dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2573–2583, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1278

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. Ontonotes release 5.0. *Linguistic Data Consortium*, Philadelphia, PA.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*. https://doi.org/10.18653/v1/P18-4013

Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013.

WebAnno: A flexible, Web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria. Association for Computational Linguistics.

Ziqi Zhang. 2013. *Named Entity Recognition: Challenges in Document Annotation, Gazetteer Construction and Disambiguation*. Ph.D. thesis, University of Sheffield.

A Alignment Heuristics

Aligning Multi-labels to Single Labels. In order to evaluate morpheme-based labels (morph or token-multi) in token-based settings, we introduce a deterministic procedure to extend the morphological labels to token boundaries. Specifically, we use regular expressions to map the multiple sequence labels to a single label by choosing the first non-O entity category (BIES) as the single category. In case the sequence of labels is not valid (e.g., B comes after E, or there is an O between two I labels), we use a relaxed mapping that does not take the order of the labels into consideration: if there is an S or *both* B and E in the sequence, return an S. Otherwise, if there is an E, return an E; if there is a B, return a B; if there is an I return an I (Figure 11).

Aligning Multi-labels to Morphemes. In order to obtain morpheme boundary labels from *token-multi*, we introduce a deterministic procedure to align the token's predicted multi-label with the list of morphemes predicted for it by the MD. Specifically, we align the multi-labels to morphemes in the order that they are both provided. In case of a mismatch between the number of labels and morphemes predicted for the token, we match label-morpheme pairs from the final one backwards. If the number of morphemes exceeds the number of labels, we pad unpaired morphemes with O labels. If the number of labels exceeds the morphemes, we drop unmatched labels (Figure 12).

```
def extend_morph_to_tok(token_labels):
    # gets list of morpheme NER labels for a token
    # returns single NER label
    biose_labels = ''
    categories = list()
    for label in token_labels:
        if label=='O':
            biose_labels += 'O'
            categories += None
        else:
            bio, cat = label.split('-')
            biose_labels += bio
            categories += cat
    single_biose = get_single_biose(biose_labels)
    single_cat = get_single_token_categ(categories)
    if single_biose == 'O':
        return 'O'
    else: return single_biose + '-' +single_cat

def get_single_token_categ(categ_labels):
    # gets list of NER categories for a token
    # returns the first category which is not None
    [...]

def get_single_biose(biose_labels):
    # gets string of 'BIOSE' for a token
    # returns single character from 'BIOSE'
    # regex matches valid labels for: O|B|S|I|E|S
    valid_seq = 'O+|O*BI*|O*BI*EO*|I+|I*EO*|O*SO*'
    if valid_seq.match(biose_labels):
        # split valid_seq by '|' and return
        # label based on the corresponding re match
        [...]
    else: # treat as set
        if 'S' in biose_labels:
            return 'S'
        elif 'B' and 'E' in biose_labels:
            return 'S'
        elif 'E' in biose_labels:
            return 'E'
        elif 'B' in biose_labels:
            return 'B'
        elif 'I' in biose_labels:
            return 'I'
        return 'O'
```

Figure 11: Multi-label to single label alignment.

```
def align_multi_to_morph(labels, forms):
    # gets list of labels and list of morph forms
    # returns aligned list of tuples (morph, label)
    if len(labels)==len(forms):
        # if same length, return zipped by order
        return zip(forms, labels)
    elif len(labels)>len(forms):
        # if extra labels, trim labels from the beginning
        # return zipped by order
        [ ... ]
    elif len(labels)<len(forms):
        # if extra forms, pad beginning with 'O'
        # return zipped by order
        [ ... ]
```

Figure 12: Multi-label to morpheme alignment.