The Role of Context in Neural Morphological Disambiguation

Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, Chris Dyer
Carnegie Mellon University
{qinlans, dclothia, etagtow, plittell, cdyer}@cs.cmu.edu

Abstract

Languages with rich morphology often introduce sparsity in language processing tasks. While morphological analyzers can reduce this sparsity by providing morpheme-level analyses for words, they will often introduce ambiguity by returning multiple analyses for the same surface form. The problem of disambiguating between these morphological parses is further complicated by the fact that a correct parse for a word is not only dependent on the surface form but also on other words in its context. In this paper, we present a language-agnostic approach to morphological disambiguation. We address the problem of using context in morphological disambiguation by presenting several LSTM-based neural architectures that encode long-range surface-level and analysis-level contextual dependencies. We applied our approach to Turkish, Russian, and Arabic to compare effectiveness across languages, matching state-of-the-art in two of the three languages. Our results also demonstrate that while context plays a role in learning how to disambiguate, the type and amount of context needed varies between languages based on their morphological and syntactic properties.

1 Introduction

Morphologically rich languages introduce sparsity in language processing tasks, as different surface variants over the same root are often taken as independent entities. Using a morphological analyzer can decompose inflected words into known tags that encode syntactic and semantic information about the word. However, finding the correct morphological parse is a non-trivial task. Functionally different morphemes may have similar forms, and long strings of potentially ambiguous morphemes compound the problem of ambiguity. As a result, analyzers for morphologically complex languages often return several parses for the same surface word. Table 1, for example, shows the resulting candidate parses for the surface form "alın" returned by Oflazer's (1994) morphological analyzer for Turkish.

```
alın+Noun+A3sg+Pnon+Nom (forehead)
al+Adj^DB+Noun+Zero+A3sg+P2sg+Nom (your red)
al+Adj^DB+Noun+Zero+A3sg+Pnon+Gen (of red)
al+Verb+Pos+Imp+A2pl ((you) take)
al+Verb^DB+Verb+Pass+Pos+Imp+A2sg ((you) be taken)
alın+Verb+Pos+Imp+A2sg ((you) be offended)
```

Table 1: Possible morphological parses for surface form "alın"

The surrounding context of the word being disambiguated plays a major part in determining the role of a word in a sentence and thus its correct morphological parse. For example, the surface form "evi" in Turkish can be interpreted as either accusative or third-person singular possessive, as shown in Table 2 (Yildiz et al., 2016); this determination cannot reliably be made without further context. Moreover, the disambiguation of a word can in turn disambiguate other words; if we have determined that "evi" is

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 181–191, Osaka, Japan, December 11-17 2016.

accusative, we can infer that a transitive verb must follow and that other ambiguous forms in a sentence are not accusative.

Sentence and translation	Analysis of `evi`
Evi bulabildiniz mi? – Did you find the house?	`ev+Noun+3sg +Pnon+Acc`
Evi gerçekten güzelmiş. – His/Her house is really beautiful	`ev+Noun+3sg +P3sg+Nom`

Table 2: Possible interpretations for "`evi`" based on context

The problem of disambiguating over the candidate morphological parses generated by a morphological analyzer has been tackled in many languages and with different strategies. These systems primarily rely on methods for capturing the structure of a target word and its candidate tag sequences (Yuret and Türe, 2006; Habash and Rambow, 2005; Daybelge and Cicekli, 2007; Daoud, 2009) and/or the surrounding context of a target word (Hakkani-Tür et al., 2002; Smith et al., 2005; Sak et al., 2008; Lee et al., 2011) to choose the best candidate analysis. Despite the breadth of work on this problem, there is little work on disambiguation using neural network models. Based on previous work, it is not clear whether neural models are able to disambiguate only using surface forms or how context plays a role in a neural disambugation model. Models that disambiguate jointly over tokens incorporate more information about the surrounding context of a word, but models that make decisions at the word level are often simpler to train.

In this paper, we present a language-agnostic LSTM-based approach for morphological disambiguation that takes into account both the structure of a word, including its candidate tag sequences, and the surrounding context of a target word. We propose a neural architecture for generating vector embeddings of the candidate analyses of a target word using character-based LSTMs to generate representations for the stems and surface forms, as well as an LSTM over the tag sequences to embed analyses.

We then describe several model architectures that operate over these vector representations, differing according to the window of context used and whether the context tokens have themselves been disambiguated. Because our architecture relies on character- and tag-level embeddings, the models can be adapted into other languages and handle unknown words at test time. Experiments on Turkish, Russian, and Arabic show that different languages benefit from different types and windows of context.

2 Models

The problem of morphological disambiguation involves selecting among a list of possible parses returned by an analyzer. Given the output of a morphological analyzer for tokens in a sentence, we use several LSTM architectures to predict the correct analysis for a word based on its context. These architectures vary based on the amount and type of context taken into account when choosing the best analysis.

Our approach to disambiguation relies on two embeddings: vector representations for each possible analysis for a word (expressed as matrix $\mathbf{R}$), which encode the stem and all of the morpheme tags for each analysis, and a vector embedding $\mathbf{h}$ of the relevant context of the target word. Using these embeddings, we can define a compatibility function between the representation of each candidate analysis and the representation of the context by taking the product of $\mathbf{R}$ and $\mathbf{h}$. Taking the softmax of this compatibility function will give us a probability distribution over possible parses given the relevant context.

$$p(y_t = a|x) = \text{softmax}(\mathbf{R}_{x_t} \times \mathbf{h}_t) \quad (1)$$

We describe a general architecture for embedding the possible parses of a target word, as well as the different architectural and objective variants for different models of context.

2.1 Analysis Embeddings

Given a morphological analysis of the form

$$stem_i + tag_{i,1} + tag_{i,2} + ... + tag_{i,L}$$

where $stem_i = (stem_{i,1}, stem_{i,2}...stem_{i,K})$ is the K character long stem of the i-th parse and each $tag_{i,j}$ is the j-th tag in the i-th parse (containing L tags), we use a bidirectional character-based LSTM to embed the stem, and a separate bidirectional LSTM over tags to embed the morphemes. A bidirectional LSTM creates a representation g_x of an input sequence $x = (x_1, x_2, ..., x_T)$ by computing a forward sequence $\overrightarrow{g}$ and a reverse sequence $\overleftarrow{g}$ over the input sequence, concatenating the results of the two sequences, and applying a rectified linear unit (ReLU) activation

$$\overrightarrow{g}_t = f(x_t, \overrightarrow{g}_{t-1}) \tag{2}$$

$$\overleftarrow{g}_t = f(x_t, \overleftarrow{g}_{t+1}) \tag{3}$$

$$g = \text{ReLU}([\overrightarrow{g}_T, \overleftarrow{g}_0]) \tag{4}$$

where $f(x, y)$ is the output of an LSTM unit with inputs x and y.

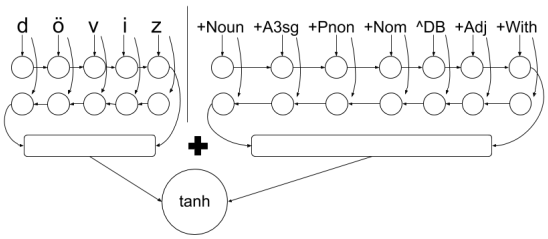

Figure 1: Neural architecture for analysis embedding

Thus, we create a representation of the stem by taking the characters of $(stem_{i,1}, stem_{i,2}, ..., stem_{i,K})$ as the input sequence for a "stem" LSTM and a representation of the tags by taking $(tag_{i,1}, tag_{i,2}, ..., tag_{i,L})$ as the input sequence for a separate "tag" LSTM. We then add these two representations and apply a $\tanh$ nonlinearity to create an embedding, r_i, for the i-th potential analysis a of the word (Figure 1).

$$r_i = \tanh(g_{stem_i} + g_{tag_i}) \tag{5}$$

These vectors of parse options r_i are concatenated to form matrix $\mathbf{R}$ for a word with N analyses, where each row in the matrix corresponds to a possible parse for a given word.

$$\mathbf{R} = [r_1; r_2; ...; r_N] \tag{6}$$

As a baseline model, we use matrix $\mathbf{R}$ without leveraging any of the surrounding context of the target word. To obtain the probability distribution of the possible parses from $\mathbf{R}$, we take the softmax of the product of $\mathbf{R}$ and a learned parameter vector $\mathbf{h}$. We then take the analysis a with the highest probability as the predicted analysis for the word.

$$p(y_t = a|x_t) = \text{softmax}(\mathbf{R}_{x_t} \times \mathbf{h}) \tag{7}$$

2.2 Surface Model

One method of integrating the context around a target word for morphological disambiguation is to leverage the surface forms of the words surrounding the target word. To capture the surface-level context of a target word, we first use another bidirectional character LSTM to embed the surface forms of each word x_i surrounding the current target word, creating a vector representation for each surrounding word.

We then use the embeddings for the relevant context words to the left of the target word as input to a left-to-right LSTM and the embeddings of the relevant context words to the right as input to a right-to-left LSTM over words to create vectors representing the left ($\overrightarrow{c}_t$) and right ($\overleftarrow{c}_t$) contexts of the target word at position t.

$$\overrightarrow{c}_t = f(x_t, \overrightarrow{c}_{t-1}) \tag{8}$$

$$\overleftarrow{c}_t = f(x_t, \overleftarrow{c}_{t+1}) \tag{9}$$

We add the left and right context vectors, then apply a tanh non-linearity to get $\mathbf{h}_t$ representing the surrounding surface context of the target word at position t.

$$\mathbf{h}_t = \tanh(\overrightarrow{c}_t + \overleftarrow{c}_t) \tag{10}$$

To combine this surface context representation with the possible parse embeddings matrix $\mathbf{R}$, we take a softmax over the product of $\mathbf{R}$ and $\mathbf{h}_t$ to return a probability distribution over possible parses given surface-level context.

$$p(y_t = a|x) = \text{softmax}(\mathbf{R}_{x_t} \times \mathbf{h}_t) \tag{11}$$

We compare two models that leverage the surface context of a word. The full context model uses all the words to the left of the target word and all the words to the right of the target word to build context vector $\mathbf{h}_t$ (Figure 2). The local context model uses a one word window to the left and right of its target word to build context vector $\mathbf{h}_t$.

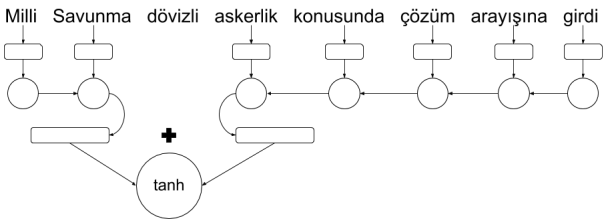

Figure 2: Neural architecture for full surface context embedding

2.3 Left-To-Right Analysis

A further question regarding the use of context in morphological disambiguation is whether the contextual tokens themselves need to be disambiguated, or whether their surface forms alone suffice to guide further disambiguation. For example, in a language like Russian with subject-verb agreement, having disambiguated a word as being third-person singular in the nominative case can help us predict that the verb should have third-person agreement.

To explore this question, we also consider models that take previously-disambiguated tokens as context. A simple way of leveraging information about the parses of surrounding words is to disambiguate the words in the sentence sequentially and use the previously selected parses to inform our decision at the current position. We use an LSTM over the selected parses of previously disambiguated words to create m_t, a representation of the decisions that were made up until position t. To build m_t, we take the representation of chosen parse $\tilde{r}^t$ from $\mathbf{R}_{x_t}$, and feed it into the LSTM encoding the sequence of previously chosen parses.

$$m_t = f(\tilde{r}_i^t, m_{t-1}) \tag{12}$$

Then, when choosing the next parse at position $t+1$, we also add m_t to the stem and morpheme representations $g_{stem_i}^{t+1}$ and $g_{tag_i}^{t+1}$ and apply a tanh nonlinearity when creating parse embeddings r_i^{t+1}.

$$r_i^{t+1} = \tanh(g_{stem_i}^{t+1} + g_{tag_i}^{t+1} + m_t) \tag{13}$$

$$\mathbf{R}_{x_{t+1}} = [r_1^{t+1}; r_2^{t+1}; ...; r_N^{t+1}] \tag{14}$$

Intuitively, the goal of this operation is to learn interactions between the previous parses and each candidate parse at the current position. We can then calculate a probability distribution over the candidates given both surface context and the previous parses in a manner similar to the process in Section 2.2.

$$p(y_t = a|x, y_1, y_2, ..., y_{t-1}) = \text{softmax}(\mathbf{R}_{x_t} \times \mathbf{h}_t) \tag{15}$$

When decoding, we use a greedy approach to select the parse to add to the previous parse LSTM. Thus, our objective is to predict the output sequence

$$\hat{y} = \text{argmax}_{\tilde{y} \in Y_X} \prod_{i=1}^{T} p(\tilde{y}_t|x, \tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_{t-1}) \tag{16}$$

2.4 Conditional Random Field Joint Decoding

Locally normalized models suffer from the label bias problem (Andor et al., 2016), meaning that they have little to no ability to revise previous decisions. Conditional Random Fields (CRFs) have been shown to be effective at modeling sequences in tasks like part of speech tagging and named entity recognition (Lample et al., 2016). In this approach, we attempt to find the best sequence of parses that takes the entire sentence into account.

The CRF model is built on top of our full-context surface model, using the same process for embedding the parses and the surface context. Rather than taking the softmax over the combined representation of the surface context vector and the parse matrix, we use the product directly as a vector $\mathbf{u}_t$ of emission scores between each word x_t and its possible parses.

$$\mathbf{u}_t = \mathbf{R}_{x_t} \times \mathbf{h}_t \tag{17}$$

To model the transition scores between the j-th parse of word at position p and the i-th analysis of the previous word, we concatenate the embeddings of the two analyses and input the resulting vector to a feed-forward layer to give a transition score between the two parses, $\mathbf{v}(r_i^{t-1}, r_j^t)$.

$$\mathbf{v}(r_i^{t-1}, r_j^t) = \tanh(\mathbf{W}_{trans}[r_i^{t-1}, r_j^t]) \tag{18}$$

We can then produce a trellis of possible parses for each word and their transitions to use for picking the best parse sequence for a given sentence x of length N. We score sequences using the function

$$S(x, y) = \sum_{i=1}^{N} \mathbf{v}(y_{i-1}, y_i) + \mathbf{u}_i^{y_i} \tag{19}$$

To find the best parse sequence, we predict the output sequence

$$\hat{y} = \text{argmax}_{\tilde{y} \in Y_X} S(x, \tilde{y}) \tag{20}$$

We learned the parameters of the CRF as part of our neural architecture, then decode using the Viterbi algorithm to compute the best analysis sequence.

3 Experimental Setup

To demonstrate the language-agnostic nature of our disambiguation model, we applied our approach to Turkish, Russian, and Arabic, three morphologically-complex but typologically-distinct languages with well-established morphological analyzers (Oflazer, 1994; Korobov, 2015; Maamouri et al., 2010).

	Turkish		Russian		Arabic	
	Ambiguous	All	Ambiguous	All	Ambiguous	All
Training	332,457	783,209	830,055	1,815,414	253,058	318,821
Development	16,327	38,744	22,344	49,773	13,915	17,387
Annotated Test	379	946	16,340	50,083	14,231	18,021
Generated Test	18,022	42,000	-	-	-	-

Table 3: Token counts for data sets

3.1 Turkish

We used the same dataset for Turkish as in Sak et al. (2007) and Yildiz et al. (2016). The datasets were extracted from a corpus of approximately 1 million words of semi-automatically disambiguated Turkish (Yuret and Türe, 2006), which were split into training, development, and test sets. To limit the effect of noise from using semi-automatically disambiguated data in our training and evaluation, we also evaluated over the small test set of human disambiguated tokens in context that was provided with the data.

Preliminary analysis of the training set found that the average number of parses per word was 1.60 (std=1.304, max=24) for all tokens and 2.81 (std=1.208) for only ambiguous tokens. The average length of a Turkish sentence within our training set was 16 tokens (std=20.231).

3.2 Russian

Data for Russian was extracted from OpenCorpora, a freely available treebank for Russian (Bocharov et al., 2011). OpenCorpora provides a large corpus of approximately 1.7 million tokens, as well as a strict, manually disambiguated subset of approximately 50,000 tokens. Due to the relatively small amount of manually disambiguated data for training, we used the parse scores returned by the pymorphy2 morphological analyzer (Korobov, 2015) to semi-automatically disambiguate the full corpus, sampling the "gold" parse based on its parse score. One previous work in Russian morphology (Muzychka et al., 2014) used the SynTagRus dataset. However, this dataset used a different tagset than the pymorphy2 analyzer (Oflazer, 1994), which is based on OpenCorpora data.

The average number of parses per word in the training set was 3.10 (std=5.329, max=69) for all tokens and 5.81 (std=6.961) for only ambiguous tokens. The average length of a Russian sentence within our training set was 19.87 tokens (std=22.974).

We took the manually disambiguated data as our test set and randomly split the remaining data from the full corpus into a training set and development set.

3.3 Arabic

Data for Arabic was extracted from the Arabic Penn Treebank (ATB) part 3 version 3.2 (catalog number LDC2010T08) (Maamouri et al., 2004), a corpus containing approximately 370,000 annotated tokens. Analyses for the tokens were generated using the Buckwalter Morphological Analyzer Version 1.0 (Buckwalter, 2002), with human annotators selecting the correct parse. Previous approaches used different versions and subsets of the ATB. Here, we used a comparable subset to previous work, split into training, development, and test sets.

Each token had 9.11 possible parses (std=8.5932, max=86) on average, with each ambiguous token having 11.31 possible parses (std=8.301). The average length of a sentence was 26.13 (std=30.78) tokens.

3.4 Training

We considered each sentence to be a minibatch for training. The objective function used for training was the total cross-entropy loss between the selected parse and the correct parse for every token in the sentence. Stochastic gradient descent and backpropagation were used to adjust the parameters for our model. To prevent overfitting on the training set, we used validation-based early-stopping, saving the model parameters based on a periodic accuracy evaluation step over the development set. All LSTMs in

our models were trained with a single hidden layer. We used a hidden dimension size of 100 for the tag, stem, and surface form LSTMs and 200 for the context and previous parse LSTMs.

4 Discussion

4.1 Results

In all cases, using some contextual information improved accuracy over the no-context baseline, but there is noticeable variation between languages regarding what kinds of context were most valuable. We report accuracy over ambiguous tokens, as well as all tokens for sake of comparison with other systems.

	Ambiguous Tokens		All Tokens	
	Annotated Test	Generated Test	Annotated Test	Generated Test
No Context	88.65	90.72	95.45	96.08
Local Context	89.18	92.65	95.67	96.90
Full Context	**91.03**	**93.46**	**96.41**	**97.24**
Left-to-Right	90.50	93.42	96.19	97.23
CRF	90.24	93.06	96.09	97.07

Table 4: Disambiguation results for Turkish

Table 4 shows the performance of all models in Turkish on a hand-annotated test set, as well as the generated test set. We see that each of the models with some form of context is able to beat the no context baseline. The best-performing model on both the generated and annotated datasets is the full surface context model, followed closely by the left-to-right and CRF models. This shows that some form of long-range contextual information is useful for disambiguation for Turkish. Our full context model is comparable to the previous state of the art of 96.28% on annotated test and 96.80% on generated test established in Sak et al. (2007).

As we will see below, of the languages considered here, Turkish is the only one in which the full surface context model approached (and slightly exceeded) other models, which may stem from the typological character of Turkish. Turkish is a strongly head-final subject-object-verb (SOV) language, and so the best disambiguating evidence for a token often follows it. For example, in disambiguating "evi" (cf. Table 2), the best evidence for its case (nominative or accusative) will lie in whether a transitive or intransitive verb follows, and in a verb-final language the verb can follow at some distance from its arguments. Meanwhile, as seen in Section 3.1, Turkish is the least ambiguous of these languages (with a mean of 1.60 parses per word compared to 3.10 for Russian and 9.11 for Arabic), so surface context is not much less informative than disambiguated context. In other words, Turkish combines the greatest need for full context with higher relative informativeness of the surface context.

	Ambiguous Tokens	All Tokens
No Context	64.97	88.58
Local Context	71.56	90.72
Full Context	69.49	90.05
Left-to-Right	68.55	89.75
CRF	**72.78**	**91.13**

Table 5: Disambiguation results for Russian

The results for Russian are detailed in Table 5. Again, each of the contextual models perform much better than the no context baseline. However, we see an interesting pattern when comparing the four contextual models; unlike in Turkish, the CRF model performs the best. Many words in Russian are required to agree in gender, case, and number to their heads. We argue that a model that takes the parse information of neighboring words into account is more suited to capturing agreement than one that only uses the surface form information. The CRF model over parse sequences would be able to capture this

phenomenon, whereas the left-to-right decoding model, which also operates over neighboring parses, may fail due to making locally but not globally optimal decisions near the beginning of the sentence.

Muzychka et al. (2014) reported an accuracy of 91.06% in their CRF-based disambiguation model. In comparison, our neural CRF-based model achieves a similar accuracy of 91.13%.

	Ambiguous Tokens	All Tokens
No Context	72.22	78.06
Local Context	80.10	84.29
Full Context	86.45	88.95
Left-to-Right	**89.30**	**91.27**

Table 6: Disambiguation results for Arabic

We see a different pattern in Table 6 for Arabic. Like in Turkish and Russian, there is a large gap between the no context baseline and the local context model. The left-to-right model performs the best on Arabic. Habash et al. (2005) report an accuracy of 96.2% on all parses, while Smith et al. (2005) achieves an accuracy of 95.4% on different subsets of the ATB.

We can also note that surface-context models performed relatively poorly in Arabic. This is likely because of the greater ambiguity of Arabic (9.11 parses per word), which stems in part from the absence of vowels in Arabic writing. Manual inspection of the Arabic output suggested that the surface-context models were frequently making the same mistakes as the no-context model (Table 7), which did not tend to occur in the Turkish output. So, in contrast to the Turkish systems, in which having only surface context was as good as having disambiguated context, in the Arabic systems having only surface context was more similar to having no context at all, emphasizing the need for disambiguated context in more highly ambiguous languages.[1]

Input:	... dwl kvyrp HAlAt **SEbp** wnsbp Alnmw mtdnyp ...
Gold:	`SaEob_1+ADJ+NSUFF_FEM_SG+CASE_INDEF_ACC` ✓
No Context:	`SaEob_1+ADJ+NSUFF_FEM_SG+CASE_INDEF_GEN`
Low Context:	`SaEob_1+ADJ+NSUFF_FEM_SG+CASE_INDEF_NOM`
Full Context:	`SaEob_1+ADJ+NSUFF_FEM_SG+CASE_INDEF_GEN`
Left-to-Right:	`SaEob_1+ADJ+NSUFF_FEM_SG+CASE_INDEF_ACC` ✓

Input:	... bAlAntSAr kmA nHyy **AlAntfADp** fy flsTyn mlyn An ...
Gold:	{`inotifADap_1+DET+NOUN+NSUFF_FEM_SG+CASE_DEF_ACC` ✓
No Context:	{`inotifADap_1+DET+NOUN+NSUFF_FEM_SG+CASE_DEF_GEN`
Low Context:	{`inotifADap_2+DET+NOUN+NSUFF_FEM_SG+CASE_DEF_GEN`
Full Context:	{`inotifADap_1+DET+NOUN+NSUFF_FEM_SG+CASE_DEF_GEN`
Left-to-Right:	{`inotifADap_1+DET+NOUN+NSUFF_FEM_SG+CASE_DEF_ACC` ✓

Table 7: Example outputs for Arabic for targets "`SEbp`" and "`AlAntfADp`"

Within our experiments, context clearly does play a role in learning to disambiguate possible morphological analyses. The type and extent of the context needed, however, appears to vary based on features of the language, such as the word order or the degree of morphological ambiguity. This raises the possibility of a future system that can choose the best disambiguation context based on a language's typological properties.

[1]Following this hypothesis, we would expect the CRF architecture to perform well on Arabic. However, the much higher ratio of parses-per-word in Arabic made our neural CRF model computationally impractical.

4.2 Future Work

As mentioned in Section 2.4, a disadvantage of using greedy decoding with our left-to-right analysis model is the label-bias problem. Because decisions depend on the previously selected parses, an incorrect decision made early on can propagate through the sentence. Thus, a natural extension to the left-to-right model is to use beam search for decoding instead of our greedy approach. The advantage of using beam search is that it allows the model to explore multiple previous parse paths. This partially gives the model the ability to recover from making a decision that appears locally optimal in the short-term but leads to greater losses in the long-term.

Another model of context inspired by recent advances in machine translation (Bahdanau et al., 2015) and caption generation (Xu et al., 2015) is using an attentional mechanism to define the important context for a target word. For the full surface context model, rather than using separate LSTMs over the left and right contexts of the word, an attentional context model can learn to attend to different parts of the surrounding context of a target word based on its sentence position and candidate parse representations. This model will potentially allow us to capture longer-range surface dependencies without decay.

5 Related Work

A common early approach to morphological disambiguation is to rely purely on hand-crafted rules to select the correct parse out of a set of candidate analyses (Daybelge and Cicekli, 2007; Daoud, 2009). These rule-based methods primarily try to capture the relationship between a target word and its candidate analyses. Other approaches used a blend of rules and statistical methods. For example, Oflazer and Tür (1996) used a set of linguistically motivated rules and corpus-dependent statistics to either choose or delete possible parses in Turkish. The model also learned rules based on unambiguous words that appear in unambiguous contexts within the corpus. Similarly, Hajič et al. (2007) built upon the same type of deletion disambiguation, using the output from a choose-delete rule-based system to first reduce the number of possible parses before running a statistical part of speech tagger in Czech.

Yuret and Türe (2006) used the Greedy Prepend Algorithm to learn rules for Turkish disambiguation. For every tag in their dataset, a decision list of patterns was created to determine whether the tag is contained in the best analysis. A heuristic search that added new attributes to patterns already in the decision list was used to generate more candidate patterns for a tag.

Other statistical approaches to morphological disambiguation tried to directly model the context of a target word. Hakkani-Tür et al. (2002) proposed a statistical approach for Turkish disambiguation using a language model trained on disambiguated data. They trained trigram language models under different root and inflectional group independence assumptions and used the resulting language models to select the best candidate parses. Smith et al. (2005) used a conditional random field to learn to disambiguate over sentence by modeling local contexts. Sak et al. (2007) and (2008) used the perceptron algorithm on a set of 23 handcrafted features, including bigrams and trigrams at the word and inflectional group level.

There is relatively little work on designing neural models specifically for morphological disambiguation. Yildiz et al. (2016) proposed a convolutional architecture that creates a representation for the surface form of a word from a root and a set of morpheme features. They then trained their model to predict the correct analysis of a word given the ground truth annotations for the previous words within a window of the target. Their model was able to achieve an accuracy of 84% over ambiguous tokens in Turkish. In contrast, our proposed model uses long short-term memory (LSTM)-based architectures to capture longer range dependencies between a target word and its surrounding context. Additionally, we consider the context of a target word at both the surface-form and analysis level, providing additional information to our models.

6 Conclusion

In this paper, we present several LSTM-based neural network architecture for disambiguating morphological parses using varying amounts of surrounding context. We demonstrate using these architectures that the type and amount of context needed for disambiguation varies between languages based on the

linguistic features of a particular language. For a language like Turkish, where most of the morphological information is apparent based on the surrounding context, a model that uses the surface context can capture long-range information to make disambiguation decisions. Other languages, where the surface representation is less informative, such as Arabic, greatly benefit from using representations of the surrounding parse candidates in addition to the surface forms of the surrounding words. We also show that, while this system is language agnostic and can be applied to typologically different languages, the best architecture for a language depends on its morphological and syntactic properties.

Acknowledgements

This work was sponsored in part by the Defense Advanced Research Projects Agency Information Innovation Office (I2O) Program under the Low Resource Languages for Emergent Incidents (LORELEI) program issued by DARPA/I2O under Contract No. HR0011-15-C-0114. We would like to thank Graham Neubig, Yulia Tsvetkov, Michael Miller Yoder, and the anonymous reviewers for their helpful comments and feedback.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *ArXiv preprint*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of the International Conference on Learning Representations*.

Victor Bocharov, Svetlana Bichineva, Dmitry Granovsky, Natalia Ostapuk, and Maria Stepanova. 2011. Quality assurance tools in the OpenCorpora project. In *Computational Linguistics and Intelligent Technology: Proceeding of the International Conference Dialog*, pages 10–17.

Tim Buckwalter. 2002. Buckwalter {Arabic} morphological analyzer version 1.0.

Daoud Daoud. 2009. Synchronized morphological and syntactic disambiguation for Arabic. *Advances in Computational Linguistics*, pages 73–86.

Turhan Daybelge and Ilyas Cicekli. 2007. A rule-based morphological disambiguator for Turkish. In *Proceedings of Recent Advances in Natural Language Processing*, pages 145–149.

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

Jan Hajič, Jan Votrubec, Pavel Krbec, Pavel Květoň, et al. 2007. The best of two worlds: Cooperation of statistical and rule-based taggers for czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies*, pages 67–74.

Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, (4):381–410.

Mikhail Korobov. 2015. Morphological Analyzer and Generator for Russian and Ukrainian Languages. pages 320–332.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*.

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.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR conference on Arabic language resources and tools*, pages 466–467.

Mohamed Maamouri, Dave Graff, Basma Bouziri, Sondos Krouna, Ann Bies, and Seth Kulick. 2010. Standard Arabic morphological analyzer (SAMA) version 3.1. *Linguistic Data Consortium, Catalog No.: LDC2010L01*.

S Muzychka, A Romanenko, and I Piontkovskaja. 2014. Conditional random field for morphological disambiguation in russian. In *Conference Dialog-2014, Bekasovo*.

Kemal Oflazer and Gokhan Tür. 1996. Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation. *Proceedings of the ACL-SIGDAT Conference on Empirical Methods in Natural Language Processing*.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. *Literary and linguistic computing*, (2):137–148.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2007. Morphological disambiguation of Turkish text with perceptron algorithm. pages 107–118.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2008. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. pages 417–427.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 475–482.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *International Conference on Machine Learning*.

Eray Yildiz, Caglar Tirkaz, H. Bahadir Sahin, Mustafa Tolga Eren, and Ozan Sonmez. 2016. A Morphology-Aware Network for Morphological Disambiguation. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Deniz Yuret and Ferhan Türe. 2006. Learning morphological disambiguation rules for Turkish. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 328–334.