Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings

Bernd Bohnet, Ryan McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, Joshua Maynez
Google Inc.
{bohnetbd,ryanmcd,gsimoes,andor,epitler,joshuahm}@google.com

Abstract

The rise of neural networks, and particularly recurrent neural networks, has produced significant advances in part-of-speech tagging accuracy (Zeman et al., 2017). One characteristic common among these models is the presence of rich initial word encodings. These encodings typically are composed of a recurrent character-based representation with learned and pre-trained word embeddings. However, these encodings do not consider a context wider than a single word and it is only through subsequent recurrent layers that word or sub-word information interacts. In this paper, we investigate models that use recurrent neural networks with sentence-level context for initial character and word-based representations. In particular we show that optimal results are obtained by integrating these context sensitive representations through synchronized training with a meta-model that learns to combine their states. We present results on part-of-speech and morphological tagging with state-of-the-art performance on a number of languages.

1 Introduction

Morphosyntactic tagging accuracy has seen dramatic improvements through the adoption of recurrent neural networks—specifically BiLSTMs (Schuster and Paliwal, 1997; Graves and Schmidhuber, 2005) to create sentence-level context sensitive encodings of words. A successful recipe is to first create an initial context insensitive word representation, which usually has three main parts: 1) A dynamically trained word embedding; 2) a fixed pre-trained word-embedding, induced from a large corpus; and 3) a sub-word character model, which itself is usually the final state of a recurrent model that ingests one character at a time. Such word/sub-word models originated with Plank et al. (2016). Recently, Dozat et al. (2017) used precisely such a context insensitive word representation as input to a BiLSTM in order to obtain context sensitive word encodings used to predict part-of-speech tags. The Dozat et al. model had the highest accuracy of all participating systems in the CoNLL 2017 shared task (Zeman et al., 2017).

In such a model, sub-word character-based representations only interact indirectly via subsequent recurrent layers. For example, consider the sentence *I had shingles, which is a painful disease*. Context insensitive character and word representations may have learned that for unknown or infrequent words like 'shingles', 's' and more so 'es' is a common way to end a plural noun. It is up to the subsequent BiLSTM layer to override this once it sees the singular verb is to the right. Note that this differs from traditional linear models where word and sub-word representations are directly concatenated with similar features in the surrounding context (Giménez and Marquez, 2004).

In this paper we aim to investigate to what extent having initial sub-word and word context insensitive representations affects performance. We propose a novel model where we learn context sensitive initial character and word representations through two separate sentence-level recurrent models. These are then combined via a meta-BiLSTM model that builds a unified representation of each word that is then used for syntactic tagging. Critically, while each of these three models—character, word and meta—are trained synchronously, they are ultimately separate models using different network configurations, training hyperparameters and loss functions. Empirically, we found this optimal as it allowed control

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pages 1–11
Melbourne, Australia, July 15 - 20, 2018.

over the fact that each representation has a different learning capacity.

We tested the system on the 2017 CoNLL shared task data sets and gain improvements compared to the top performing systems for the majority of languages for part-of-speech and morphological tagging. As we will see, a pattern emerged where gains were largest for morphologically rich languages, especially those in the Slavic family group. We also applied the approach to the benchmark English PTB data, where our model achieved 97.9 using the standard train/dev/test split, which constitutes a relative reduction in error of 12% over the previous best system.

2 Related Work

While sub-word representations are often attributed to the advent of deep learning in NLP, it was, in fact, commonplace for linear featurized machine learning methods to incorporate such representations. While the literature is too large to enumerate, Giménez and Marquez (2004) is a good example of an accurate linear model that uses both word and sub-word features. Specifically, like most systems of the time, they use n-gram affix features, which were made context sensitive via manually constructed conjunctions with features from other words in a fixed window.

Collobert and Weston (2008) was perhaps the first modern neural network for tagging. While this first study used only word embeddings, a subsequent model extended the representation to include suffix embeddings (Collobert et al., 2011).

The seminal dependency parsing paper of Chen and Manning (2014) led to a number of tagging papers that used their basic architecture of highly featurized (and embedded) feed-forward neural networks. Botha et al. (2017), for example, studied this architecture in a low resource setting using word, sub-word (prefix/suffix) and induced cluster features to obtain competitive accuracy with the state-of-the-art. Zhou et al. (2015), Alberti et al. (2015) and Andor et al. (2016) extended the work of Chen et al. to a structured prediction setting, the later two use again a mix of word and sub-word features.

The idea of using a recurrent layer over characters to induce a complementary view of a word has occurred in numerous papers. Perhaps the earliest is Santos and Zadrozny (2014) who compare character-based LSTM encodings to traditional word-based embeddings. Ling et al. (2015) take this a step further and combine the word embeddings with a recurrent character encoding of the word—instead of just relying on one or the other. Alberti et al. (2017) use characters encodings for parsing. Peters et al. (2018) show that contextual embeddings using character convolutions improve accuracy for number of NLP tasks. Plank et al. (2016) is probably the jumping-off point for most current architectures for tagging models with recurrent neural networks. Specifically, they used a combined word embedding and recurrent character encoding as the initial input to a BiLSTM that generated context sensitive word encodings. Though, like most previous studies, these initial encodings were context insensitive and relied on subsequent layers to encode sentence-level interactions.

Finally, Dozat et al. (2017) showed that sub-word/word combination representations lead to state-of-the-art morphosyntactic tagging accuracy across a number of languages in the CoNLL 2017 shared task (Zeman et al., 2017). Their word representation consisted of three parts: 1) A dynamically trained word embedding; 2) a fixed pretrained word embedding; 3) a character LSTM encoding that summed the final state of the recurrent model with vector constructed using an attention mechanism over all character states. Again, the initial representations are all context insensitive. As this model is currently the state-of-the-art in morphosyntactic tagging, it will serve as a baseline during our discussion and experiments.

3 Models

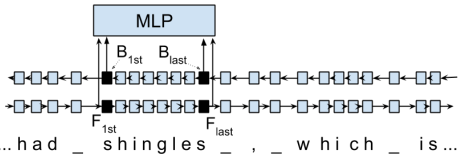
In this section, we introduce models that we investigate and experiment with in §4.

3.1 Sentence-based Character Model

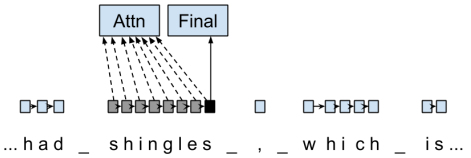
The feature that distinguishes our model most from previous work is that we apply a bidirectional recurrent layer (LSTM) on all characters of a sentence to induce fully context sensitive initial word encodings. That is, we do not restrict the context of this layer to the words themselves (as in Figure 1b). Figure 1a shows the sentence-based character model applied to an example token in context.

The character model uses, as input, sentences split into UTF8 characters. We include the spaces between the tokens[1] in the input and map each

[1]As input, we assume the sentence has been tok-



(a) Sentence-based Character Model. The representation for the token *shingles* is the concatenation of the four shaded boxes. Note the surrounding sentence context affects the representation.



(b) Token-based Character Model[a]. The token is represented by the concatenation of attention over the lightly shaded boxes with the final cell (dark shaded box). The rest of the sentence has no impact on the representation.

[a]This is specifically the model of Dozat et al. (2017).

Figure 1: Token representations are sensitive to the context in the sentence-based character model (§3.1) and are context-independent in the token-based character model (§3.2).

character to a dynamically learned embedding.

Next, a forward LSTM reads the characters from left to right and a backward LSTM reads sentences from right to left, in standard BiLSTM fashion.

More formally, for an n-character sentence, we apply for each character embedding $(e_1^{char}, ..., e_n^{char})$ a BiLSTM:

$$f_{c,i}^0, b_{c,i}^0 = \text{BiLSTM}(r_0, (e_1^{char}, ..., e_n^{char}))_i$$

As is also typical, we can have multiple such layers (l) that feed into each other through the concatenation of previous layer encodings. The last layer l has both forward $(f_{c,1}^l, ..., f_{c,n}^l)$ and backward $(b_{c,1}^l, ..., b_{c,n}^l)$ output vectors for each character.

To create word encodings, we need to combine a relevant subset of these context sensitive character encodings. These word encodings can then be used in a model that assigns morphosyntactic tags to each word directly or via subsequent layers. To accomplish this, the model concatenates up to four character output vectors: the {*forward*, *backward*} output of the {*first*, *last*} character in the token ($F_{1st}(w)$, $F_{last}(w)$, $B_{1st}(w)$, $B_{last}(w)$). In Figure 1a, the four shaded boxes indicate these four outputs for the example token. Thus, the proposed model concatenates all four of these and passes it as input to an multilayer perceptron (MLP):

$$\begin{aligned} g_i &= \text{concat}(F_{1st}(w), F_{last}(w), \\ &\qquad B_{1st}(w), B_{last}(w)) \qquad (1) \\ m_i^{chars} &= \text{MLP}(g_i) \end{aligned}$$

A tag can then be predicted with a linear classifier that takes as input the output of the MLP m_i^{chars}, applies a softmax function and chooses for each word the tag with highest probability. Table 8 investigates the empirical impact of alternative definitions of g_i that concatenate only subsets of $\{F_{1st}(w), F_{last}(w), B_{1st}(w), B_{last}(w)\}$.

enized/segmented.

3.2 Word-based Character Model

To investigate whether a sentence sensitive character model is better than a character model where the context is restricted to the characters of a word, we reimplemented the word-based character model of Dozat et al. (2017) as shown in Figure 1a. This model uses the final state of a unidirectional LSTM over the characters of the word, combined with the attention mechanism of Cao and Rei (2016) over all characters. We refer the reader to those works for more details. Critically, however, all the information fed to this representation comes from the word itself, and not a wider sentence-level context.

3.3 Sentence-based Word Model

We used a similar setup for our context sensitive word encodings as the character encodings. There are a few differences. Obviously, the inputs are the words of the sentence. For each of the words, we use pretrained word embeddings $(p_1^{word}, ..., p_n^{word})$ summed with a dynamically learned word embedding for each word in the corpus $(e_1^{word}, ..., e_n^{word})$:

$$in_i^{word} = e_i^{word} + p_i^{word}$$

The summed embeddings in_i are passed as input to one or more BiLSTM layers whose output $f_{w,i}^l, b_{w,i}^l$ is concatenated and used as the final encoding, which is then passed to an MLP

$$\begin{aligned} o_i^{word} &= \text{concat}(f_{w,i}^l, b_{w,i}^l) \\ m_i^{word} &= \text{MLP}(o_i^{word}) \end{aligned}$$

It should be noted, that the output of this BiLSTM is essentially the Dozat et al. model before tag prediction, with the exception that the word-based character encodings are excluded.

3.4 Meta-BiLSTM: Model Combination

Given initial word encodings, both character and word-based, a common strategy is to pass these through a sentence-level BiLSTM to create context sensitive encodings, e.g., this is precisely what Plank et al. (2016) and Dozat et al. (2017) do. However, we found that if we trained each of the character-based and word-based encodings with their own loss, and combined them using an additional meta-BiLSTM model, we obtained optimal performance. In the meta-BiLSTM model, we concatenate the output, for each word, of its context sensitive character and word-based encodings, and put this through another BiLSTM to create an additional combined context sensitive encoding. This is followed by a final MLP whose output is passed to a linear layer for tag prediction.

$$cw_i = \text{concat}(m_i^{char}, m_i^{word})$$
$$f_{m,i}^l, b_{m,i}^l = \text{BiLSTM}(r_0, (cw_0, ..., cw_n))_i$$
$$m_i^{comb} = \text{MLP}(\text{concat}(f_{m,i}^l, b_{m,i}^l))$$

With this setup, each of the models can be optimized independently which we describe in more detail in §3.5. Figure 2b depicts the architecture of the combined system and contrasts it with that of the Dozat et al. model (Figure 2a).

3.5 Training Schema

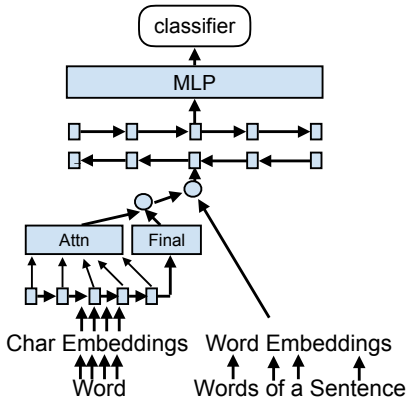
As mentioned in the previous section, the character and word-based encoding models have their own tagging loss functions, which are trained independently and joined via the meta-BiLSTM. I.e., the loss of each model is minimized independently by separate optimizers with their own hyperparameters. Thus, it is in some sense a multi-task learning model and we must define a schedule in which individual models are updated. We opted for a simple synchronous schedule outline in Algorithm 1. Here, during each epoch, we update each of the models in sequence—character, word and meta—using the entire training data.

In terms of model selection, after each epoch, the algorithm evaluates the tagging accuracy of the development set and keeps the parameters of the best model. Accuracy is measured using the meta-BiLSTM tagging layer, which requires a forward pass through all three models. Though we use all three losses to update the models, only the meta-BiLSTM layer is used for model selection and test-time prediction.

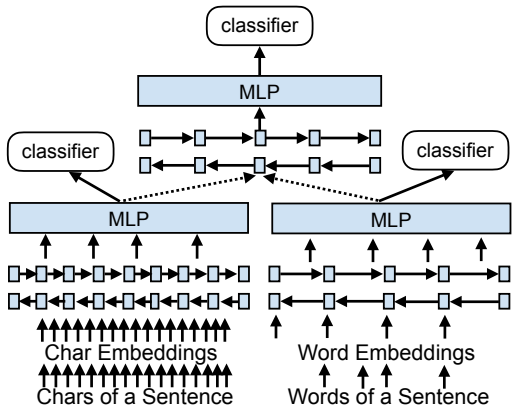
Data: train-corpus, dev-corpus
/* The following models are defined in §3. */
Input: char-model, word-model, meta-model
/* Model optimizers */
Input: char-opt, word-opt, meta-opt
/* Results are parameter sets for each model. */
Result: best-char, best-word, best-meta
/* Initialize parameter sets (cf. Table 1) */
Initialize(pa_c, pa_w, pa_m)
/* Iteration on over training corpus. */
for *epoch = 1 to MAX* **do**
  /* Update character model. */
  char-logits, char-preds = char-model(train-corpus, pa_c)
  pa_c = char-opt.update(char-preds, train-data)
  /* Update word model. */
  word-logits, word-preds = word-model(train-corpus, pa_w)
  pa_w = word-opt.update(char-preds, train-data)
  /* Update Meta-BiLSTM model. */
  meta-preds = meta-model(train-corpus, pa_c, pa_w, pa_m)
  pa_m = meta-opt.update(train-corpus, meta-preds)
  /* Evaluate model due to dev set accuracy. */
  F1 = DevEval(par_c, par_w, par_m)
  /* Keep the best model. */
  if *F1 > best-F1* **then**
    best-char = pa_c; *best-word* = pa_w
    best-meta = pa_m; *best-F1* = *F1*
  end
end

Algorithm 1: Training procedure for learning initial character and word-based context sensitive encodings synchronously with meta-BiLSTM.

While each of the three models—character, word and meta—are trained with their own loss functions, it should be emphasized that training is synchronous in the sense that the meta-BiLSTM model is trained in tandem with the two encoding models, and not after those models have converged. Since accuracy from the meta-BiLSTM model on the development set determines the best parameters, training is not completely independent. We found this to improve accuracy overall. Crucially, when we allowed the meta-BiLSTM to back-propagate through the whole network, performance degraded regardless of whether one or multiple loss functions were used.



(a) The overall architecture of Dozat et al. (2017)



(b) The overall architecture of the system. The data flows along the arrows. The optimizers minimizes the loss of the classifiers independently and backpropagates along the bold arrows.

Figure 2: Tagging architectures. (a) Dozat et al. (2017); (b) Meta-BiLSTM architecture of this work.

Each language could in theory use separate hyperparameters, optimized for highest accuracy. However, for our main experiments we use identical settings for each language which worked well for large corpora and simplified things. We provide an overview of the selected hyperparameters in §4.1. We explored more settings for selected individual languages with a grid search and ablation experiments and present the results in §5.

4 Experiments and Results

In this section, we present the experimental setup and the selected hyperparameter for the main experiments where we use the CoNLL Shared Task 2017 treebanks and compare with the best systems of the shared task.

4.1 Experimental Setup

For our main results, we selected one network configuration and set of the hyperparameters. These settings are not optimal for all languages. However, since hyperparameter exploration is computationally demanding due to the number of languages we optimized these hyperparameters on initial development data experiments over a few languages. Table 1 shows an overview of the architecture, hyperparameters and the initialization settings of the network. The word embeddings are initialized with zero values and the pre-trained embeddings are not updated during training. The dropout used on the embeddings is achieved by a single dropout mask and we use dropout on the input and the states of the LSTM.

Architecture		
Model	**Parameter**	**Value**
Chr, Wrd	BiLSTM layers	3
Mt	BiLSTM layers	1
Chr, Wrd, Mt	BiLSTM size	400
Chr, Wrd, Mt	Dropout LSTMs	0.33
Chr, Wrd, Mt	Dropout MLP	0.33
Wrd	Dropout embeddings	0.33
Chr	Dropout embeddings	0.05
Chr, Wrd, Mt	Nonlinear act. (MLP)	ELU
Initialization		
Model	**Parameter**	**Value**
Wrd	embeddings	Zero
Chr	embeddings	Gaussian
Chr, Wrd, Mt	MLP	Gaussian
Training		
Model	**Parameter**	**Value**
Chr, Wrd, Mt	Optimizer	Adam
Chr, Wrd, Mt	Loss	Cross entropy
Chr, Wrd, Mt	Learning rate	0.002
Chr, Wrd, Mt	Decay	0.999994
Chr, Wrd, Mt	Adam epsilon	1e-08
Chr, Wrd, Mt	beta1	0.9
Chr, Wrd, Mt	beta2	0.999

Table 1: Selected hyperparameters and initialization of parameters for our models. *Chr*, *Wrd*, and *Mt* are used to indicate the character, word, and meta models respectively. The Gaussian distribution is used with a mean of 0 and variance of 1 to generate the random values.

As is standard, model selection was done measuring development accuracy/F1 score after each epoch and taking the model with maximum value on the development set.

4.2 Data Sets

For the experiments, we use the data sets as provided by the CoNLL Shared Task 2017 (Zeman et al., 2017). For training, we use the training sets which were denoted as big treebanks [2].

We followed the same methodology used in the CoNLL Shared Task. We use the training treebank for training only and the development sets for hyperparameter tuning and early stopping. To keep our results comparable with the Shared Task, we use the provided precomputed word embeddings. We excluded Gothic from our experiments as the available downloadable content did not include embeddings for this language.

As input to our system—for both part-of-speech tagging and morphological tagging—we use the output of the UDPipe-base baseline system (Straka and Straková, 2017) which provides segmentation. The segmentation differs from the gold segmentation and impacts accuracy negatively for a number of languages. Most of the top performing systems for part-of-speech tagging used as input UDPipe to obtain the segmentation for the input data. For morphology, the top system for most languages (IMS) used its own segmentation (Björkelund et al., 2017). For the evaluation, we used the official evaluation script (Zeman et al., 2017).

4.3 Part-of-Speech Tagging Results

In this section, we present the results of the application of our model to part-of-speech tagging. In our first experiment, we used our model in the setting of the CoNLL 2017 Shared Task to annotate words with XPOS[3] tags (Zeman et al., 2017). We compare our results against the top systems of the CoNLL 2017 Shared Task. Table 2 contains the results of this task for the large treebanks.

Because Dozat et al. (2017) won the challenge for the majority of the languages, we first compare our results with the performance of their system. Our model outperforms Dozat et al. (2017) in 32 of the 54 treebanks with 13 ties. These ties correspond mostly to languages where XPOS tagging anyhow obtains accuracies above 99%. Our model tends to produce better results, especially for morphologically rich languages (e.g. Slavic

[2]In the CONLL 2017 Shared Task, a big treebank is one that contains a development set. In total, there are 55 out of the 64 UD treebanks which are considered big treebanks.

[3]These are the language specific fine-grained part-of-speech tags from the Universal Dependency Treebanks.

lang.	CONLL Winner	DQM	ours	RRIE
cs_cac	95.16	95.16	**96.91**	36.2
cs	95.86	95.86	**97.28**	35.5
fi	97.37	97.37	**97.81**	16.7
sl	94.74	94.74	**95.54**	15.2
la_ittb	94.79	94.79	**95.56**	14.8
grc	84.47	84.47	**86.51**	13.1
bg	96.71	96.71	**97.05**	10.3
ca	98.58	98.58	**98.72**	9.9
grc_proiel	97.51	97.51	**97.72**	8.4
pt	83.04	83.04	**84.39**	8.0
cu	96.20	96.20	**96.49**	7.6
it	97.93	97.93	**98.08**	7.2
fa	97.12	97.12	**97.32**	6.9
ru	96.73	96.73	**96.95**	6.7
sv	96.40	96.40	**96.64**	6.7
ko	93.02	93.02	**93.45**	6.2
sk	85.00	85.00	**85.88**	5.9
nl	90.61	90.61	**91.10**	5.4
fi_ftb	95.31	95.31	**95.56**	5.3
de	97.29	97.29	**97.39**	4.7
tr	93.11	93.11	**93.43**	4.6
hi	97.01	97.01	**97.13**	4.0
es_ancora	98.73	98.73	**98.78**	3.9
ro	96.98	96.98	**97.08**	3.6
la_proiel	96.93	96.93	**97.00**	2.3
pl	91.97	91.97	**92.12**	1.9
ar	87.66	87.66	**87.82**	1.3
gl	97.50	97.50	**97.53**	1.2
sv_lines	94.84	94.84	**94.90**	1.2
cs_clt	89.98	89.98	**90.09**	1.1
lv	80.05	80.05	**80.20**	0.8
zh	**88.40**	85.07	85.10	0.2
da	**100.00**	99.96	99.96	0.0
es	**99.81**	99.69	99.69	0.0
eu	**99.98**	99.96	99.96	0.0
fr_sequoia	**99.49**	99.06	99.06	0.0
fr	**99.50**	98.87	98.87	0.0
hr	**99.93**	**99.93**	**99.93**	0.0
hu	**99.85**	99.82	99.82	0.0
id	**100.00**	99.99	99.99	0.0
ja	**98.59**	89.68	89.68	0.0
nl_lassy	**99.99**	99.93	99.93	0.0
no_bok.	**99.88**	99.75	99.75	0.0
no_nyn.	**99.93**	99.85	99.85	0.0
ru_syn.	**99.58**	99.57	99.57	0.0
en_lines	**95.41**	**95.41**	95.39	-0.4
ur	**92.30**	**92.30**	92.21	-1.2
he	**83.24**	82.45	82.16	-1.7
vi	**75.42**	73.56	73.12	-1.7
gl_treegal	**91.65**	**91.65**	91.40	-3.0
en	**94.82**	**94.82**	94.66	-3.1
en_partut	**95.08**	**95.08**	94.81	-5.5
pt_br	**98.22**	**98.22**	98.11	-6.2
et	**95.05**	**95.05**	94.72	-6.7
el	**97.76**	**97.76**	97.53	-10.3
macro-avg	93.18	93.11	**93.40**	-

Table 2: Results for XPOS tags. The first column shows the language acronym, the column named **DQM** shows the results of Dozat et al. (2017). Our system outperforms Dozat et al. (2017) on 32 out of 54 treebanks and Dozat et al. outperforms our model on 10 of 54 treebanks, with 13 ties. **RRIE** is the relative reduction in error. We excluded ties in the calculation of macro-avg since these treebanks do not contain meaningful xpos tags.

System	Accuracy
Søgaard (2011)	97.50
Huang et al. (2015)	97.55
Choi (2016)	97.64
Andor et al. (2016).	97.44
Dozat et al. (2017)	97.41
ours	**97.96**

Table 3: Results on WSJ test set.

languages), whereas Dozat et al. (2017) showed higher performance in 10 languages in particular English, Greek, Brazilian Portuguese and Estonian.

4.4 Part-of-Speech Tagging on WSJ

We also performed experiments on the Penn Treebank with the usual split in train, development and test set. Table 3 shows the results of our model in comparison to the results reported in state-of-the-art literature. Our model significantly outperforms these systems, with an absolute difference of 0.32% in accuracy, which corresponds to a RRIE of 12%.

4.5 Morphological Tagging Results

In addition to the XPOS tagging experiments, we performed experiments with morphological tagging. This annotation was part of the CONLL 2017 Shared Task and the objective was to predict a bundle of morphological features for each token in the text. Our model treats the morphological bundle as one tag making the problem equivalent to a sequential tagging problem. Table 4 shows the results.

Our models tend to produce significantly better results than the winners of the CoNLL 2017 Shared Task (i.e., 1.8% absolute improvement on average, corresponding to a RRIE of 21.20%). The only cases for which this is not true are again languages that require significant segmentation efforts (i.e., Hebrew, Chinese, Vietnamese and Japanese) or when the task was trivial.

Given the fact that Dozat et al. (2017) obtained the best results in part-of-speech tagging by a significant margin in the CoNLL 2017 Shared Task, it would be expected that their model would also perform significantly well in morphological tagging since the tasks are very similar. Since they did not participate in this particular challenge, we decided to reimplement their system to serve

lang.	CONLL Winner	**DQM** Reimpl.	**ours**	**RRIE**
cs_cac	90.72	94.66	**96.41**	27.9
ru_syn.	94.55	96.70	**97.53**	23.1
cs	93.14	96.32	**97.14**	22.3
la_ittb	94.28	96.45	**97.12**	18.9
sl	90.08	95.26	**96.03**	16.2
ca	97.23	97.85	**98.13**	13.0
fi_ftb	93.43	95.96	**96.42**	11.4
no_bok.	95.56	96.95	**97.26**	10.2
grc_proiel	90.24	91.35	**92.22**	10.1
fr_sequoia	96.10	96.62	**97.62**	10.1
la_proiel	89.22	91.52	**92.35**	9.8
es_ancora	97.72	98.15	**98.32**	9.7
da	94.83	96.62	**96.94**	9.5
fi	92.43	94.29	**94.83**	9.5
sv	95.15	96.52	**96.84**	9.2
pt	94.62	95.89	**96.27**	9.2
grc	88.00	90.39	**91.13**	9.0
no_nyn.	95.25	96.79	**97.08**	9.0
de	83.11	89.78	**90.70**	9.0
ru	87.27	91.99	**92.69**	8.7
hi	91.03	90.72	**91.78**	8.1
cu	88.90	88.93	**89.82**	8.0
fa	96.34	97.23	**97.45**	7.9
tr	87.03	89.39	**90.21**	7.7
en_partut	92.69	93.93	**94.40**	7.7
sk	81.23	87.54	**88.48**	7.5
eu	89.57	92.48	**93.04**	7.4
pt_br	99.73	99.73	**99.75**	7.4
es	96.34	96.42	**96.68**	7.3
ko	99.41	99.44	**99.48**	7.1
ar	87.15	85.45	**88.29**	6.7
it	97.37	97.72	**97.86**	6.1
nl_lassy	97.55	98.04	**98.15**	5.2
nl	90.04	92.06	**92.47**	5.2
pl	86.53	91.71	**92.14**	5.2
ur	81.03	83.16	**84.02**	5.1
bg	96.47	97.71	**97.82**	4.8
hr	85.82	90.64	**91.50**	3.8
he	**85.06**	79.34	79.76	2.0
et	84.62	88.18	**88.25**	0.6
zh	**92.90**	87.67	87.74	0.6
vi	**86.92**	82.23	82.30	0.4
ja	**96.84**	89.65	89.66	0.1
en_lines	99.96	99.99	99.99	0.0
fr	96.12	95.98	95.98	0.0
gl	**99.78**	99.72	99.72	0.0
id	**99.55**	99.50	99.50	0.0
ro	96.24	97.26	97.26	0.0
sv_lines	99.98	99.98	99.98	0.0
cs_cltt	87.88	**90.41**	90.36	-0.5
lv	84.14	**87.00**	86.92	-0.6
el	91.37	**94.00**	93.92	-1.3
hu	72.61	**82.67**	82.44	-1.3
en	94.49	**95.93**	95.71	-5.4
macro-avg	91.51	92.89	**93.31**	-

Table 4: Results for morphological features. The column **CoNLL Winner** shows the top system of the ST 17, the **DQM Reimpl.** shows our reimplementation of Dozat et al. (2017), the column **ours** shows our system with a sentence-based character model; **RRIE** gives the relative reduction in error between the Reimpl. DQM and sentence-based character system. Our system outperforms the CoNLL Winner by 48 out of 54 treebanks and the reimplementation of DQM, by 43 of 54 treebanks, with 6 ties.

as a strong baseline. As expected, our reimplementation of Dozat et al. (2017) tends to significantly outperform the winners of the CONLL 2017 Shared Task. However, in general, our models still obtain better results, outperforming Dozat et al. on 43 of the 54 treebanks, with an absolute difference of 0.42% on average.

5 Ablation Study

The model proposed in this paper of a Meta-BiLSTM with a sentence-based character model differs from prior work in multiple aspects. In this section, we perform ablations to determine the relative impact of each modeling decision.

For the experimental setup of the ablation experiments, we report accuracy scores for the development sets. We split off 5% of the sentences from each training corpus and we use this part for early stopping. Ablation experiments are either performed on a few selected treebanks to show individual language results or averaged across all treebanks for which tagging is non-trivial.

Impact of the Training Schema We first compare jointly training the three model components (Meta-BiLSTM, character model, word model) to training each separately. Table 5 shows that separately optimized models are significantly more accurate on average than jointly optimized models. Separate optimization leads to better accuracy for 34 out of 40 treebanks for the morphological features task and for 30 out of 39 treebanks for xpos tagging. Separate optimization outperformed joint optimization by up to 2.1 percent absolute, while joint never out-performed separate by more than 0.5% absolute. We hypothesize that separately training the models forces each submodel (word and character) to be strong enough to make high accuracy predictions and in some sense serves as a regularizer in the same way that dropout does for individual neurons.

Impact of the Sentence-based Character Model We compared the setup with sentence-based character context (Figure 1a) to word-based character context (Figure 1b). We selected for these experiments a number of morphological rich languages. The results are shown in Table 6. The accuracy of the word-based character model joint with a word-based model were significantly lower than a sentence-based character model. We conclude also from these results and comparing with results of the reimplementation of DQM that early integration of the word-based character model performs much better as late integration via Meta-BiLSTM for a word-based character model.

Optimization	Avg. F1 Score morphology	Avg. F1 Score xpos
separate	**94.57**	**94.85**
jointly	94.15	94.48

Table 5: Comparison of optimization methods: Separate optimization of the word, character and meta model is more accurate on average than full back-propagation using a single loss function.The results are statistically significant with two-tailed paired t-test for xpos with p<0.001 and for morphology with p <0.0001.

dev. set	word char model	sentence char model
el	89.05	93.41
la_ittb	93.22	95.69
ru	88.94	92.31
tr	87.78	90.77

Table 6: F1 score for selected languages on sentence vs. word level character models for the prediction of morphology using late integration.

dev. set lang.	num. exp.	mean char	mean word	mean joint	stdev char	stdev word	stdev joint
el	10	96.43	95.36	**97.01**	0.13	0.11	0.09
grc	10	88.28	73.52	**88.85**	0.21	0.29	0.22
la_ittb	10	91.45	87.98	**91.94**	0.14	0.30	0.05
ru	10	95.98	93.50	**96.61**	0.06	0.17	0.07
tr	10	93.77	90.48	**94.67**	0.11	0.33	0.14

Table 7: F1 score for the character, word and joint models. The standard deviation of 10 random restarts of each model is show in the last three columns. The differences in means are all statistically significant at $p < 0.001$ (paired t-test).

Impact of the Meta-BiLSTM Model Combination The proposed model trains word and character models independently while training a joint model on top. Here we investigate the part-of-speech tagging performance of the joint model compared with the word and character models on their own (using hyperparameters from in 4.1).

Table 5 shows, for selected languages, the results averaged over 10 runs in order to measure standard deviation. The examples show that the combined model has significantly higher accuracy compared with either the character and word models individually.

dev. set. lang.	F_{last} B_{1st}	F_{1st} B_{last}	F_{last} B_{last}	F_{1st} B_{1st}	DQM	|xpos|
el	**96.6**	**96.6**	96.2	96.1	95.9	16
grc	**87.3**	87.1	87.1	86.8	86.7	3130
la_ittb	91.1	91.5	**91.9**	91.3	91.0	811
ru	95.6	95.4	95.6	95.3	**95.8**	49
tr	93.5	93.3	93.2	92.5	**93.9**	37

Table 8: F1 score of char models and their performance on the dev. set for selected languages with different gather strategies, concatenate to g_i (Equation 1). DQM shows results for our reimplementation of Dozat et al. (2017) (cf. §3.2), where we feed in only the characters. The final column shows the number of xpos tags in the training set.

Concatenation Strategies for the Context-Sensitive Character Encodings The proposed model bases a token encoding on both the forward and the backward character representations of both the first and last character in the token (see Equation 1). Table 8 reports, for a few morphological rich languages, the part-of-speech tagging performance of different strategies to gather the characters when creating initial word encodings. The strategies were defined in §3.1. The Table also contains a column with results for our reimplementation of Dozat et al. (2017). We removed, for all systems, the word model in order to assess each strategy in isolation. The performance is quite different per language. E.g., for Latin, the outputs of the forward and backward LSTMs of the last character scored highest.

Sensitivity to Hyperparameter Search We picked Vietnamese for a more in-depth analysis since it did not perform well and investigated the influence of the sizes of LSTMs for the word and character model on the accuracy of development set. With larger network sizes, the capacity of the network increases, however, on the other hand it is prune to overfitting. We fixed all the hyperparameters except those for the network size of the character model and the word model, and ran a grid search over dimension sizes from 200 to 500 in steps of 50. The surface plot in 3 shows that the grid peaks with more moderate settings around 350 LSTM cells which might lead to a higher accuracy. For all of the network sizes in the grid search, we still observed during training that the accuracy reach a high value and degrades with more iterations for the character and word model. This suggests that future variants of this model might benefit from higher regularization.

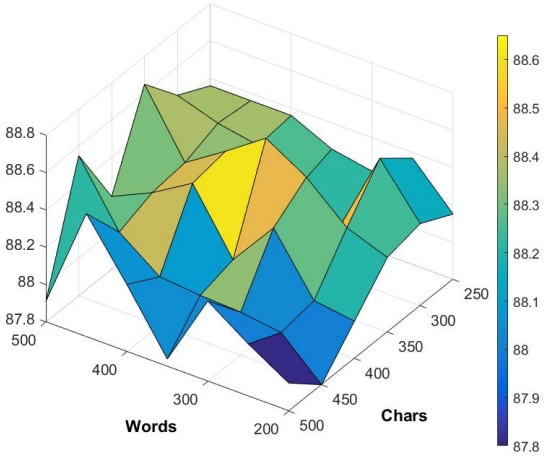


Figure 3: 3D surface plot for development set accuracy for XPOS (y-axis) depending on LSTM size of the character and word model for the Vietnamese treebank. The snapshot is take after 195 training epochs and we average the values of neighboring epochs.

Discussion Generally, the fact that different techniques for creating word encodings from character encodings and different network sizes can lead to different accuracies per language suggests that it should be possible to increase the accuracy of our model on a per language basis via a grid search over all possibilities. In fact, there are many variations on the models we presented in this work (e.g., how the character and word models are combined with the meta-BiLSTM). Since we are using separate losses, we could also change our training schema. For example, one could use methods like stack-propagation (Zhang and Weiss, 2016) where we burn-in the character and word models and then train the meta-BiLSTM backpropagating throughout the entire network.

6 Conclusions

We presented an approach to morphosyntactic tagging that combines context-sensitive initial character and word encodings with a meta-BiLSTM layer to obtain state-of-the art accuracies for a wide variety of languages.

Acknowledgments

We would like to thank the anonymous reviewers as well as Terry Koo, Slav Petrov, Vera Axelrod, Kellie Websterk, Jan Botha, Kuzman Ganchev, Zhuoran Yu, Yuan Zhang, Eva Schlinger, Ji Ma, and John Alex for their helpful suggestions, comments and discussions.

References

Chris Alberti, Daniel Andor, Ivan Bogatyy, Michael Collins, Dan Gillick, Lingpeng Kong, Terry Koo, Ji Ma, Mark Omernick, Slav Petrov, Chayut Thanapirom, Zora Tung, and David Weiss. 2017. Syntaxnet models for the conll 2017 shared task http://arxiv.org/abs/1703.04929.

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1354–1359.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2442–2452.

Anders Björkelund, Agnieszka Falenska, Xiang Yu, and Jonas Kuhn. 2017. Ims at the conll 2017 ud shared task: Crfs and perceptrons meet neural networks. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 40–51.

Jan A. Botha, Emily Pitler, Ji Ma, Anton Bakalov, Alex Salcianu, David Weiss, Ryan McDonald, and Slav Petrov. 2017. Natural language processing with small feed-forward networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2879–2885.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. pages 18–26.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 740–750.

Jinho D. Choi. 2016. Dynamic Feature Induction: The Last Gist to the State-of-the-Art. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*. San Diego, CA, NAACL'16, pages 271–281.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, August 3-4, 2017*. pages 20–30.

Jesús Giménez and Lluis Marquez. 2004. Fast and accurate part-of-speech tagging: The svm approach revisited. *Recent Advances in Natural Language Processing III* pages 153–162.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5):602–610.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging http://arxiv.org/abs/1508.01991.

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

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations http://arxiv.org/abs/1802.05365.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 412–418.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1818–1826.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Anders Søgaard. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 48–52.

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

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağr Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 1–19.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1557–1566.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Beijing, China, pages 1213–1222.