Contextual String Embeddings for Sequence Labeling

Alan Akbik
Zalando Research
Mühlenstraße 25
10243 Berlin

Duncan Blythe
Zalando Research
Mühlenstraße 25
10243 Berlin

Roland Vollgraf
Zalando Research
Mühlenstraße 25
10243 Berlin

{firstname.lastname}@zalando.de

Abstract

Recent advances in language modeling using recurrent neural networks have made it viable to model language as distributions over characters. By learning to predict the next character on the basis of previous characters, such models have been shown to automatically internalize linguistic concepts such as words, sentences, subclauses and even sentiment. In this paper, we propose to leverage the internal states of a trained character language model to produce a novel type of word embedding which we refer to as *contextual string embeddings*. Our proposed embeddings have the distinct properties that they (a) are trained without any explicit notion of words and thus fundamentally model words as sequences of characters, and (b) are *contextualized* by their surrounding text, meaning that the same word will have different embeddings depending on its contextual use. We conduct a comparative evaluation against previous embeddings and find that our embeddings are highly useful for downstream tasks: across four classic sequence labeling tasks we consistently outperform the previous state-of-the-art. In particular, we significantly outperform previous work on English and German named entity recognition (NER), allowing us to report new state-of-the-art F1-scores on the CoNLL03 shared task.

We release all code and pre-trained language models in a simple-to-use framework to the research community, to enable reproduction of these experiments and application of our proposed embeddings to other tasks: https://github.com/zalandoresearch/flair

1 Introduction

A large family of NLP tasks such as named entity recognition (NER) and part-of-speech (PoS) tagging may be formulated as *sequence labeling* problems; text is treated as a sequence of words to be labeled with linguistic tags. Current state-of-the-art approaches for sequence labeling typically use the LSTM variant of bidirectional recurrent neural networks (BiLSTMs), and a subsequent conditional random field (CRF) decoding layer (Huang et al., 2015; Ma and Hovy, 2016).

A crucial component in such approaches are *word embeddings*, typically trained over very large collections of unlabeled data to assist learning and generalization. Current state-of-the-art methods concatenate up to three distinct embedding types:

1. Classical word embeddings (Pennington et al., 2014; Mikolov et al., 2013), pre-trained over very large corpora and shown to capture latent syntactic and semantic similarities.

2. Character-level features (Ma and Hovy, 2016; Lample et al., 2016), which are not pre-trained, but trained on task data to capture task-specific subword features.

3. Contextualized word embeddings (Peters et al., 2017; Peters et al., 2018) that capture word semantics in context to address the polysemous and context-dependent nature of words.

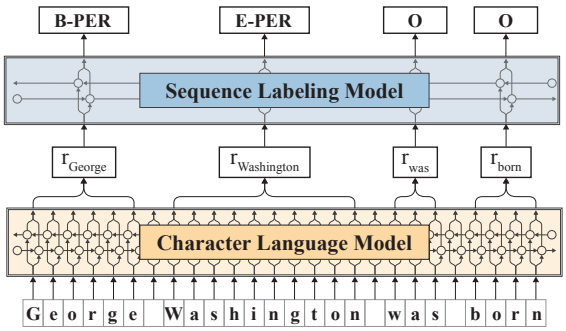

Figure 1: High level overview of proposed approach. A sentence is input as a character sequence into a pre-trained bidirectional character language model. From this LM, we retrieve for each word a contextual embedding that we pass into a vanilla BiLSTM-CRF sequence labeler, achieving robust state-of-the-art results on downstream tasks (NER in Figure).

Contextual string embeddings. In this paper, we propose a novel type of contextualized character-level word embedding which we hypothesize to combine the best attributes of the above-mentioned embeddings; namely, the ability to (1) pre-train on large unlabeled corpora, (2) capture word meaning in context and therefore produce different embeddings for polysemous words depending on their usage, and (3) model words and context fundamentally as sequences of characters, to both better handle rare and misspelled words as well as model subword structures such as prefixes and endings.

We present a method to generate such a contextualized embedding for any string of characters in a sentential context, and thus refer to the proposed representations as *contextual string embeddings*.

Neural character-level language modeling. We base our proposed embeddings on recent advances in neural language modeling (LM) that have allowed language to be modeled as distributions over sequences of characters instead of words (Sutskever et al., 2011; Graves, 2013; Kim et al., 2015). Recent work has shown that by learning to predict the next character on the basis of previous characters, such models learn internal representations that capture syntactic and semantic properties: even though trained without an explicit notion of word and sentence boundaries, they have been shown to generate grammatically correct text, including words, subclauses, quotes and sentences (Sutskever et al., 2014; Graves, 2013; Karpathy et al., 2015). More recently, Radford et al. (2017) showed that individual neurons in a large LSTM-LM can be attributed to specific semantic functions, such as predicting sentiment, without explicitly trained on a sentiment label set.

We show that an appropriate selection of hidden states from such a language model can be utilized to generate word-level embeddings that are highly effective in downstream sequence labeling tasks.

State-of-the-art sequence labeling. Based on this, we propose the sequence tagging architecture illustrated in Figure 1: each sentence is passed as a sequence of characters to a bidirectional character-level neural language model, from which we retrieve for each word the internal character states to create a contextual string embedding. This embedding is then utilized in the BiLSTM-CRF sequence tagging module to address a downstream NLP task (NER in the Figure).

We experimentally verify our approach in the classic sequence labeling tasks of named entity recognition for English and German, phrase chunking and part-of-speech tagging, and find that our approach reliably achieves state-of-the-art results. In particular, for both German and English NER, our approach significantly improves the state-of-the-art. But even for highly saturated tasks such as PoS tagging and chunking we find slight improvements over the already strong state-of-the-art (see Table 1).

We also find that our proposed embeddings on some tasks subsume previous embedding types, enabling simplified sequence labeling architectures. In addition to this, the character-level LM is compact and relatively efficient to train in comparison to word-level models. This allows us to easily train models for new languages or domains.

Contributions. To summarize, this paper proposes contextual string embeddings, a novel type of word embeddings based on character-level language modeling, and their use in a state-of-the-art sequence labeling architecture. Specifically, we

- illustrate how we extract such representations from a character-level neural language model, and

Task	PROPOSED	Previous best
NER English	**93.09**±0.12	*92.22*±0.1 (Peters et al., 2018)
NER German	**88.32**±0.2	*78.76* (Lample et al., 2016)
Chunking	**96.72**±0.05	*96.37*±0.05 (Peters et al., 2017)
PoS tagging	**97.85**±0.01	*97.64* (Choi, 2016)

Table 1: Summary of evaluation results for best configuration of proposed architecture, and current best published results. The proposed approach significantly outperforms previous work on the CoNLL03 NER task for German and English and slightly outperforms previous works on CoNLL2000 chunking and Penn treebank PoS tagging.

integrate them into a simplified sequence labeling architecture;

- present experiments in which we quantiatively evaluate the usefulness and inherent semantics of the proposed embeddings against previous embeddings and their stacked combinations in downstream tasks;

- report a new state-of-the-art on the CoNLL03 NER task for English (**93.09** F1, ↑0.87 pp vs. previous best) and German (**88.33** F1, ↑9.56 pp vs. previous best), and state-of-the-art scores for chunking and PoS;

- release all code and pre-trained language models in a simple-to-use framework to the research community, to enable reproduction of these experiments and application of our proposed embeddings to other tasks.

This paper is structured as follows: we present our approach for extracting contextual string embeddings from character-level language models in Section 2. We evaluate our approach against prior work in Section 3. We then discuss the results and present an outlook into future work in Section 4.

2 Contextual String Embeddings

Our proposed approach passes sentences as sequences of characters into a character-level language model to form word-level embeddings. Refer to Figure 2 for an example illustration.

2.1 Recurrent Network States

Like recent work, we use the LSTM variant (Hochreiter and Schmidhuber, 1997; Graves, 2013; Zaremba et al., 2014) of recurrent neural networks (Sutskever et al., 2011) as language modeling architecture. These have been shown to far outperform earlier n-gram based models (Jozefowicz et al., 2016) due to the ability of LSTMs to flexibly encode long-term dependencies with their hidden state. We use characters as atomic units of language modeling (Graves, 2013), allowing text to be treated as a sequence of characters passed to an LSTM which at each point in the sequence is trained to predict the next character[1]. This means that the model possesses a hidden state for each character in the sequence.

Formally, the goal of a character-level language model is to estimate a good distribution $P(\mathbf{x}_{0:T})$ over sequences of charaters $(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T) =: \mathbf{x}_{0:T}$ reflecting natural language production (Rosenfeld, 2000). By training a language model, we learn $P(\mathbf{x}_t|\mathbf{x}_0, \ldots, \mathbf{x}_{t-1})$, an estimate of the predictive distribution over the next character given past characters. The joint distribution over entire sentences can then be decomposed as a product of the predictive distribution over characters conditioned on the preceeding characters:

$$P(\mathbf{x}_{0:T}) = \prod_{t=0}^{T} P(\mathbf{x}_t|\mathbf{x}_{0:t-1}) \tag{1}$$

[1]Note that character-level LM is different from *character-aware* LM (Kim et al., 2015) which still operates on the word-level, but also takes into account character-level features through an additional CNN encoding step.

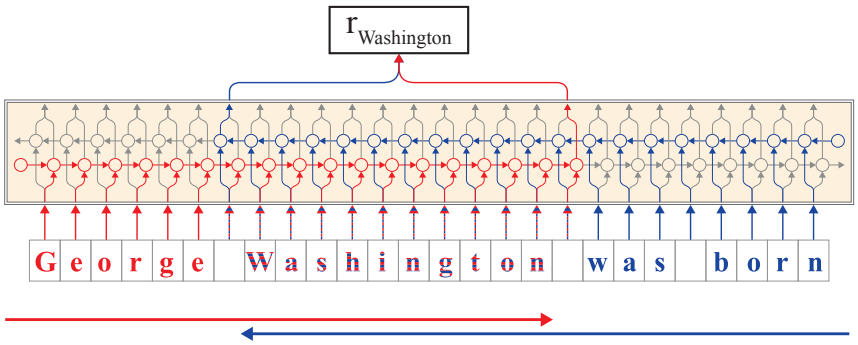

Figure 2: Extraction of a contextual string embedding for a word ("Washington") in a sentential context. From the forward language model (shown in red), we extract the output hidden state after the last character in the word. This hidden state thus contains information propagated from the beginning of the sentence up to this point. From the backward language model (shown in blue), we extract the output hidden state before the first character in the word. It thus contains information propagated from the end of the sentence to this point. Both output hidden states are concatenated to form the final embedding.

In the LSTM architecture, the conditional probability $P\left(\mathbf{x}_t|\mathbf{x}_{0:t-1}\right)$ is approximately a function of the network output $\mathbf{h}_t$.

$$P\left(\mathbf{x}_t|\mathbf{x}_{0:t-1}\right) \approx \prod_{t=0}^{T} P\left(\mathbf{x}_t|\mathbf{h}_t;\theta\right) \tag{2}$$

$\mathbf{h}_t$ represents the entire past of the character sequence. In an LSTM in particular, it is computed recursively, with the help of an additional recurrent quantity $\mathbf{c}_t$, the memory cell,

$$\begin{aligned}\mathbf{h}_t\left(\mathbf{x}_{0:t-1}\right) &= f_{\mathbf{h}}\left(\mathbf{x}_{t-1},\mathbf{h}_{t-1},\mathbf{c}_{t-1};\theta\right)\\ \mathbf{c}_t\left(\mathbf{x}_{0:t-1}\right) &= f_{\mathbf{c}}\left(\mathbf{x}_{t-1},\mathbf{h}_{t-1},\mathbf{c}_{t-1};\theta\right)\ ,\end{aligned}$$

where θ denotes all the parameters of the model. $\mathbf{h}_{-1}$ and $\mathbf{c}_{-1}$ can be initialized with zero or can be treated as part of the model parameters θ. In our model, a fully conected softmax layer (without bias) is placed ontop of $\mathbf{h}_t$, so the likelihood of every character is given by

$$P\left(\mathbf{x}_t|\mathbf{h}_t;\mathbf{V}\right) = \text{softmax}\left(\mathbf{V}\mathbf{h}_t+\mathbf{b}\right) \tag{3}$$

$$= \frac{\exp\left(\mathbf{V}\mathbf{h}_t+\mathbf{b}\right)}{\left\|\exp\left(\mathbf{V}\mathbf{h}_t+\mathbf{b}\right)\right\|_1} \tag{4}$$

where $\mathbf{V}$ and $\mathbf{b}$, weights and biases, are part of the model parameters θ (Graves, 2013; Jozefowicz et al., 2016).

2.2 Extracting Word Representations

We utilize the hidden states of a forward-backward recurrent neural network to create contextualized word embeddings. This means, alongside with the forward model (2), we also have a backward model, which works in the same way but in the reversed direction:

$$P^b\left(\mathbf{x}_t|\mathbf{x}_{t+1:T}\right) \approx \prod_{t=0}^{T} P^b\left(\mathbf{x}_t|\mathbf{h}_t^b,\theta\right) \tag{5}$$

$$\mathbf{h}_t^b = f_{\mathbf{h}}^b\left(\mathbf{x}_{t+1},\mathbf{h}_{t+1}^b,\mathbf{c}_{t+1}^b;\theta\right) \tag{6}$$

$$\mathbf{c}_t^b = f_{\mathbf{c}}^b\left(\mathbf{x}_{t+1},\mathbf{h}_{t+1}^b,\mathbf{c}_{t+1}^b;\theta\right) \tag{7}$$

Note that, in the following, we will use the superscript $\cdot^f$ to define $\mathbf{h}_t^f := \mathbf{h}_t$, $\mathbf{c}_t^f := \mathbf{c}_t$ for the forward model described in the previous section.

From this forward-backward LM, we concatenate the following hidden character states for each word: from the fLM, we extract the output hidden state after the last character in the word. Since the fLM

is trained to predict likely continuations of the sentence after this character, the hidden state encodes semantic-syntactic information of the sentence up to this point, including the word itself. Similarly, we extract the output hidden state before the word's first character from the bLM to capture semantic-syntactic information from the end of the sentence to this character. Both output hidden states are concatenated to form the final embedding and capture the semantic-syntactic information of the word itself as well as its surrounding context.

Formally, let the individual word-strings begin at character inputs with indices $t_0, t_1, \ldots, t_n$, then we define contextual string embeddings of these words as:

$$\mathbf{w}_i^{CharLM} := \begin{bmatrix} \mathbf{h}^f_{t_{i+1}-1} \\ \mathbf{h}^b_{t_i-1} \end{bmatrix} \tag{8}$$

We illustrate our approach in Figure 2 for a word in an example sentence, with the fLM in red and the bLM shown in blue.

Our approach thus produces embeddings from hidden states that are computed not only on the characters of a word, but also the characters of the surrounding context, since it influences the LM's ability to predict likely continuations of a sentence. As we later illustrate in Section 3.4, our proposed approach thus produces different embeddings for the same lexical word string in different contexts, and is able to accurately capture the semantics of contextual use together with word semantics itself.

2.3 Sequence Labeling Architecture

In the default configuration of our approach, the final word embeddings are passed into a BiLSTM-CRF sequence labeling module as proposed by Huang et al. (2015) to address downstream sequence labeling tasks. Then let us call the inputs to the BiLSTM g_l: $\mathbf{w}_0, \ldots, \mathbf{w}_n$. Then we have that:

$$\mathbf{r}_i := \begin{bmatrix} \mathbf{r}^f_i \\ \mathbf{r}^b_i \end{bmatrix} \tag{9}$$

Where $\mathbf{r}^f_i$ and $\mathbf{r}^f_i$ are the forward and backward output states of the BiLSTM g_l. The final sequence probability is then given by a CRF over the possible sequence labels y:

$$\widehat{P}(\mathbf{y}_{0:n}|\mathbf{r}_{0:n}) \propto \prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, \mathbf{r}_i) \tag{10}$$

Where:

$$\psi_i(y', y, \mathbf{r}) = \exp(\mathbf{W}_{y',y}\mathbf{r} + \mathbf{b}_{y',y}) \tag{11}$$

Alternatively, we also experiment with directly applying a simple feedforward linear architecture (essentially multinomial logistic regression (Menard, 2018)). This configuration simply linearly projects the hidden states of the neural character LM to make predictions:

$$\mathbf{r}_i = \mathbf{W}_r\mathbf{w}_i + \mathbf{b}_r \tag{12}$$

Then the prediction of the label is given by:

$$P(\mathbf{y}_i = j|\mathbf{r}_i) = \text{softmax}(\mathbf{r}_i)[j] \tag{13}$$

Stacking Embeddings. Current sequence labeling models often combine different types of embeddings by concatenating each embedding vector to form the final word vectors. We similarly experiment with different stackings of embeddings; for instance in many configurations it may be benficial to add classic word embeddings to add potentially greater latent word-level semantics to our proposed embeddings. In this case, the final words representation is given by

$$\mathbf{w}_i = \begin{bmatrix} \mathbf{w}_i^{CharLM} \\ \mathbf{w}_i^{GloVe} \end{bmatrix} \tag{14}$$

Here $\mathbf{w}_i^{GloVe}$ is a precomputed GLOVE embedding (Pennington et al., 2014). We present different configurations of stacked embeddings in the next section for the purpose of evaluation.

3 Experiments

We conduct an experimental evaluation to assess in how far our proposed contextual string embeddings are useful for sequence labeling, and how they compare to existing embedding types. In addition, we aim to gain insights into the inherent semantics of the proposed embeddings.

Tasks. To this end, we evaluate our embeddings in four classic sequence labeling tasks in their default evaluation setups, namely named entity recognition in the CoNLL03 setup (Tjong Kim Sang and De Meulder, 2003), chunking in the CoNLL2000 setup (Tjong Kim Sang and Buchholz, 2000) and PoS tagging in the default Penn treebank setup described by Collins (2002). We evaluate NER to determine in how far our embeddings are helpful for shallow semantic tasks, while we use chunking and PoS to determine in how far they are helpful for shallow syntactic tasks. We also include the German NER task from the CoNLL03 setup to evaluate our embeddings in a language with richer morphology.

3.1 Experimental Setup

We utilize the BiLSTM-CRF sequence labeling architecture proposed by Huang et. al (2015) in all configurations of our comparative evaluation. In this architecture, we evaluate our proposed contextual string embeddings in stacked combinations with the three types of previous embeddings discussed in Section 1; namely (1) pre-trained static ("classic") word embeddings (Pennington et al., 2014), (2) task-trained (i.e. not pre-trained) static character features (Ma and Hovy, 2016; Lample et al., 2016), and (3) pre-trained contextual word embeddings (Peters et al., 2018).

Baselines. We also evaluate setups that involve only previous word embeddings. These are effectively reimplementations of state-of-the-art approaches within our sequence labeling architecture. We conduct these experiments to isolate the impact of our proposed embeddings vis-a-vis earlier approaches. The setups are as follows:

HUANG : A standard BiLSTM-CRF setup with pre-trained word embeddings, and thus a reimplementation of Huang et al. (2015).

LAMPLE : A hierarchical BiLSTM-CRF setup with pre-trained word embeddings, in which task-trained character features are additionally computed by a character-level BiLSTM for each word. It is thus a reimplementation of Lample et al. (2016).

PETERS : A BiLSTM-CRF setup that utilizes the release of Peters et al. (2018) in the ALLENNLP library to produce contextualized word embeddings, and thus is effectively a reimplementation of Peters et al. (2018) in our framework. Note that only English embeddings are provided, so we use this baseline only on the English tasks.

Proposed approach. We evaluate our proposed contextual string embeddings in the following configurations:

PROPOSED : The simplest setup of our proposed approach that relies solely on our proposed contextual string embeddings, passed to a standard BiLSTM-CRF architecture. This configuration thus matches the illustration in Figure 1.

PROPOSED$_{+\text{WORD}}$: An extension of PROPOSED in which we concatenate pre-trained static word embeddings with our contextual string embeddings as per Equation (14), to determine whether they complement or subsume our proposed embeddings.

PROPOSED$_{+\text{CHAR}}$:A similar extension in which we concatenate task-trained character features in a hierarchical BiLSTM-CRF architecture to our contextual string embeddings, to determine whether they complement or subsume our proposed embeddings.

PROPOSED$_{+\text{WORD+CHAR}}$: We also evaluate a setting in which we add both pre-trained word and task-trained character embeddings.

PROPOSED$_{+\text{ALL}}$: Finally, we evaluate a setup that uses all four embeddings. Since Peters et al. (2018) embeddings are only distributed for English, we use this setup only on the English tasks.

Approach	NER-English F1-score	NER-German F1-score	Chunking F1-score	POS Accuracy
proposed				
PROPOSED	91.97±0.04	85.78 ± 0.18	96.68±0.03	97.73±0.02
PROPOSED$_{+\text{WORD}}$	93.07±0.10	88.20 ± 0.21	96.70±0.04	97.82±0.02
PROPOSED$_{+\text{CHAR}}$	91.92±0.03	85.88 ± 0.20	**96.72**±0.05	97.8±0.01
PROPOSED$_{+\text{WORD+CHAR}}$	**93.09**±0.12	**88.32** ± 0.20	96.71±0.07	97.76±0.01
PROPOSED$_{+\text{ALL}}$	92.72±0.09	n/a	96.65±0.05	**97.85**±0.01
baselines				
HUANG	88.54±0.08	82.32 ± 0.35	95.4±0.08	96.94±0.02
LAMPLE	89.3±0.23	83.78 ± 0.39	95.34±0.06	97.02±0.03
PETERS	92.34±0.09	n/a	96.69±0.05	97.81± 0.02
best published				
	92.22±0.10	78.76	96.37±0.05	97.64
	(Peters et al., 2018)	(Lample et al., 2016)	(Peters et al., 2017)	(Choi, 2016)
	91.93±0.19	77.20	95.96±0.08	97.55
	(Peters et al., 2017)	(Seyler et al., 2017)	(Liu et al., 2017)	(Ma and Hovy, 2016)
	91.71±0.10	76.22	95.77	97.53±0.03
	(Liu et al., 2017)	(Gillick et al., 2015)	(Hashimoto et al., 2016)	(Liu et al., 2017)
	91.21	75.72	95.56	97.30
	(Ma and Hovy, 2016)	(Qi et al., 2009)	Søgaard et al. (2016)	(Lample et al., 2016)

Table 2: Summary of evaluation results or all proposed setups and baselines. We also list the best published scores for each task for reference. We significantly outperform all previous works on NER, and slightly outperform the previous state-of-the-art in PoS tagging and chunking.

3.2 Model Training and Parameters

Character-level language models. We train our LMs using SGD to perform truncated backpropagation through time (BPTT) with a window length of 250, a non-annealed learning rate of 20.0, a batch size of 100, clipping gradients at 0.25 and dropout probabilities of 0.25. We prepare the data by unking-the rarest 0.0001 percent of characters. We set the number of hidden states of the (one-layered) LSTM to 2048. We halt training by tracking the performance on the validation set, stopping when negligible gains were observed, or after 1 week, whichever came first.

We train our English models on the 1-billion word corpus (Chelba et al., 2013) and our German models on a corpus of half a billion words aggregated from various open source corpora in the OPUS project[2]. After training, we find the models trained for the full week achieve character level perplexity on the supplied test-set of 2.42 for English and 2.38 for German. Resources did not allow us to train for longer or on larger language model variants. We hypothesize that more resources and time would permit learning an even better model.

Sequence tagging model. We train the sequence tagging model using vanilla SGD with no momentum, clipping gradients at 5, for 150 epochs. We employ a simple learning rate annealing method in which we halve the learning rate if training loss does not fall for 5 consecutive epochs. Following recommendations of Reimers et al. (2017)'s in-depth analysis of hyperparameters in sequence labeling, we utilize variational dropout, set the number of hidden states per-layer of the LSTM to 256, set the number of LSTM layers to 1, and perform model selection over the learning rate $\in \{0.01, 0.05, 0.1\}$ and mini-batch size $\in \{8, 16, 32\}$, choosing the model with the best F-measure (for NER and chunking) or accuracy (for PoS) in the best epoch as judged by performance on the validation set. Following Peters et al. (2017), we then repeat the experiment for the model chosen 5 times with different random seeds, and train using both train and development set, reporting both average performance and standard deviation over these runs on the test set as final performance.

Classic word embeddings. Following Reimers et al. (2017), we use GLOVE embeddings for English NER and KOMNIOS embeddings (Komninos and Manandhar, 2016) for PoS tagging and chunking. For German, we use the German FASTTEXT embeddings (Grave et al., 2018). In configurations that train character features, we apply a BiLSTM with 25 hidden states to each word separately and extract the final hidden output states (see Lample et al. (2016)).

[2]We aggregate over the PARACRAWL, EUROPARL, OPENSUBTITLES2018, and WIKIPEDIA releases in OPUS for versatile text and to allow reproduction of our results.

Embedding + Architecture	NER-English F1-score	NER-German F1-score	Chunking F1-score	POS Accuracy
PROPOSED$_{+\text{WORD}}$				
+BiLSTM-CRF	93.07 ± 0.10	88.20 ± 0.21	96.70 ± 0.04	97.82 ± 0.02
+Map-CRF	90.17 ± 0.06	85.17 ± 0.04	96.05 ± 0.04	97.62 ± 0.01
+Map	79.86 ± 0.12	76.97 ± 0.16	90.55 ± 0.05	97.35 ± 0.01
PROPOSED				
+BiLSTM-CRF	91.97 ± 0.04	85.78 ± 0.18	96.68 ± 0.03	97.73 ± 0.02
+Map-CRF	88.62 ± 0.15	82.27 ± 0.22	95.96 ± 0.05	97.53 ± 0.02
+Map	81.42 ± 0.16	73.90 ± 0.09	90.50 ± 0.06	97.26 ± 0.01
CLASSIC WORD EMBEDDINGS				
+BiLSTM-CRF	88.54 ± 0.08	82.32 ± 0.35	95.40 ± 0.08	96.94 ± 0.02
+Map-CRF	66.53 ± 0.03	72.69 ± 0.12	91.26 ± 0.04	94.06 ± 0.02
+Map	48.79 ± 0.27	57.43 ± 0.12	65.01 ± 0.50	89.58 ± 0.02

Table 3: Additional ablation experiment in which we evaluate our approach without BiLSTM and CRF. Here, we instead use a simple linear map over the embeddings to determine their direct information content.

3.3 Results

Our experimental results are summarized in Table 2 for each of the four tasks. We find that our approach either slightly or strongly surpasses all previously published results. This shows that our proposed contextual string embeddings are indeed highly useful for sequence labeling. In more detail, we make the following observations:

New state-of-the-art for NER. We find that our approach performs particularly well on the task of named entity recognition. For English, we surpass the previous state-of-the-art approach by a significant margin (**93.09** F1, ↑0.87 pp vs. Peters et al. (2018)). For German in particular, we significantly raise the state-of-the-art (**88.33** F1, ↑9.56 pp vs. Lample et al. (2016)). We hypothesize that pre-trained contextual character-level features are particularly helpful for this task, since entities are an open vocabulary of names that are often indicated by character features (such as capitalization or endings), as well as their contextual use. In addition, the CONLL03 shared task data set contains many sentences (such as article titles) that are in all-caps; since most traditional embeddings perform lower casing to manage vocabulary size, we assume that our character model can better capture the features of such sentences.

Good performance on syntactic tasks. We also find that our approach slightly outperforms the latest state-of-the-art approaches for chunking and PoS tagging. We do not however see the same improvements as for NER, which we mainly attribute to the fact that syntactic tasks are already solved by current approaches to very high accuracy, making it difficult to record further strong improvements (see Manning et al. (2014) for a discussion of this issue).

Traditional word embeddings helpful. Additionally, we observe that the added use of classic word embeddings in setup PROPOSED$_{+\text{WORD}}$ often produces the best results. Especially for NER, the use of classic word embeddings increases average F1 score by 1.1 pp to 93.07, compared with 91.97 for PROPOSED. We hypothesize that classic word embeddings capture word-level semantics that complement the strong character-level features of our proposed embeddings. We therefore recommend the setup PROPOSED$_{+\text{WORD}}$ as default setup for sequence labeling tasks.

Task-specific character features unnecessary. Another observation is that setups with task-specific character features (PROPOSED$_{+\text{CHAR}}$ and PROPOSED$_{+\text{WORD+CHAR}}$) do not perform better than those without. This indicates that they are largely subsumed by our proposed embeddings, and are thus no longer required. A positive side-effect of this is that this simplifies training of the sequence labeler (see Section 3.5).

3.4 Inherent Semantics

Quantiative investigation (Table 3). To gain additional insight into the content of the contextual string embeddings relative to the task, we replace the standard BiLSTM-CRF in the sequence labeling architecture with a direct feedforward map as per Equation (12), with and without a CRF decoder. This simplified approach makes predictions directly on the basis of the proposed embeddings without additional learning of the BiLSTM recurrence. The results for this simplified architecture are displayed in Table 3 for the

word	context	selected nearest neighbors
Washington	(a) ***Washington*** *to curb support for [..]*	(1) ***Washington*** *would also take [..] action [..]* (2) ***Russia*** *to clamp down on barter deals [..]* (3) ***Brazil*** *to use hovercrafts for [..]*
Washington	(b) *[..] Anthony* ***Washington*** *(U.S.) [..]*	(1) *[..] Carla* ***Sacramento*** *(Portugal) [..]* (2) *[..] Charles* ***Austin*** *(U.S.) [..]* (3) *[..] Steve* ***Backley*** *(Britain) [..]*
Washington	(c) *[..] flown to* ***Washington*** *for [..]*	(1) *[..] while visiting* ***Washington*** *to [..]* (2) *[..] journey to New York City and* ***Washington*** *[..]* (14) *[..] lives in* ***Chicago*** *[..]*
Washington	(d) *[..] when* ***Washington*** *came charging back [..]*	(1) *[..] point for victory when* ***Washington*** *found [..]* (4) *[..] before* ***England*** *struck back with [..]* (6) *[..] before* ***Ethiopia*** *won the spot kick decider [..]*
Washington	(e) *[..] said* ***Washington*** *[..]*	(1) *[..] subdue the never-say-die* ***Washington*** *[..]* (4) *[..] a private school in* ***Washington*** *[..]* (9) *[..] said* ***Florida*** *manager John Boles [..]*

Table 4: Examples of the word "Washington" in different contexts in the CoNLL03 data set, and nearest neighbors using cosine distance over our proposed embeddings. Since our approach produces different embeddings based on context, we retrieve different nearest neighbors for each mention of the same word.

setups PROPOSED and PROPOSED$_{+\text{WORD}}$, as well as for a setup that involves only traditional word embeddings (GLOVE for English NER, KOMNIOS for English PoS and chunking, FASTTEXT for German NER).

We find that the effect of removing the BiLSTM layer on downstream task accuracy is far lower for the proposed embeddings than for classic embeddings. For the setups PROPOSED and PROPOSED$_{+\text{WORD}}$, we record only an average drop of 3% in F-score/accuracy between the BiLSTM-CRF and Map-CRF architectures. This stands in contrast to classic embeddings in which we find an average drop of 20% from BiLSTM-CRF to Map-CRF. This indicates that the inherent semantics of the proposed embeddings are meaningful enough as to require much less powerful learning architectures on top to perform downstream sequence labeling tasks. In particular, for PoS tagging, the simple feedforward map is competitive to BiLSTM and much more effective to train.

Qualitative inspection (Table 4). To illustrate the contextualized nature of our proposed embeddings, we present example embeddings of the polysemous word "Washington" in different contexts. We compute contextual string embeddings for all words in the English CoNLL03 corpus and compute nearest neighbors in the embedding space using the cosine distance. We then look up nearest neighbors for different mentions of the word "Washington".

As Table 4 shows, the embeddings successfully pry apart person, place, legislative entity and team (a-d). For instance, "Washington" used as last name in context (b) is closest to other last names, many of which are also place names ("Carla Sacramento"); "Washington" used as a sport team name in context (d) is closest to other place names used in sports team contexts. We include a negative example (e) in Table 4 in which the context is not sufficient to determine the type of mention. We hypothesize that modeling semantics in context is a key feature that allows our proposed embeddings to better address downstream sequence labeling task.

3.5 Discussion

Our proposed approach is one of the first to leverage hidden states from a language model to improve sequence labeling performance. Two prior works have suggested related approaches: The first is Liu et al. (2017) that jointly train a character-level language model together with the sequence labeling BiLSTM. In effect, this means that the language model is trained only on labeled task data and therefore has orders of magnitude fewer data available than our proposed approach (which we can pre-train on basically unlimited amounts of unlabled data). We hypothesize that this is the main reason for why our approach outperforms Liu et al. (2017) across all tasks.

A second approach is the method by Peters et. al (2017) which proposed to extract hidden states from pre-trained word-level language models as features for downstream NLP tasks. They report new state-

of-the-art results for NER (Peters et al., 2018) and chunking (Peters et al., 2017), but require the training of massive word-level language models: their best configuration uses a language model that was trained for 5 weeks on 32 GPUs (Jozefowicz et al., 2016), and still requires lower casing of all words to deal with the large vocabulary size. Their approach is the strongest baseline against which we compared.

We similarly propose to utilize trained LMs to generate embeddings, but consider LM at the character level. This has a number of advantages: First, character-level LM is independent of tokenization and a fixed vocabulary. Second, they produce stronger character-level features, which is particularly useful for downstream tasks such as NER. Finally, such models have a much smaller vocabulary size (distinct characters vs. distinct words) and are thus significantly easier to train and deploy in applications: the LM parameter-count scales according to $n_{layers}\, n^2_{hidden}$ (not true for word-level LM since these contain many inputs) and may be trained in 1 week on 1 GPU (e.g. significantly less than the 5 weeks on 32 GPUs used by Peters et al. (2017)). This allowed us to train models for other languages such as German with moderate resources, and thus allows our method to more effectively scale to new languages or domains.

4 Conclusion

We have proposed novel *contextual string embeddings* and a method for producing them using neural character LM. We find that they capture syntactic-semantic word features and disambiguate words in context, resulting in state-of-the-art performance in classic NLP sequence labeling tasks. To facilitate reproduction of our experiments, and enable application of our proposed embeddings to other tasks, we release all code and pre-trained language models in a simple-to-use framework to the research community[3]. Future work will focus on applying these embeddings to additional sentence level tasks such as image-retrieval and neural translation.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732328 ("FashionBrain").

References

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Jinho D Choi. 2016. Dynamic feature induction: The last gist to the state-of-the-art. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 271–281.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

[3] https://github.com/zalandoresearch/flair

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower sequence labeling with task-aware neural language model. *arXiv preprint arXiv:1709.04109*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTMs-CNNs-CRF. *arXiv preprint arXiv:1603.01354*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Scott Menard. 2018. *Applied logistic regression analysis*, volume 106. SAGE publications.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada, July. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, and Christopher Clark Kenton Lee Luke Zettlemoyer Mohit Iyyer, Matt Gardner. 2018. Deep contextualized word representations. *6th International Conference on Learning Representations*.

Yanjun Qi, Ronan Collobert, Pavel Kuksa, Koray Kavukcuoglu, and Jason Weston. 2009. Combining labeled and unlabeled data with word-class distribution learning. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1737–1740. ACM.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark, 09.

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278.

Dominic Seyler, Tatiana Dembelova, Luciano Del Corro, Johannes Hoffart, and Gerhard Weikum. 2017. Knowner: Incremental multilingual knowledge in named entity recognition. *arXiv preprint arXiv:1709.03544*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 231–235.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 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*, pages 142–147. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.