Character and Subword-Based word Representation for Neural Language Modeling prediction

Matthieu Labeau
LIMSI-CNRS / Orsay, France
labeau@limsi.fr

Alexandre Allauzen
LIMSI-CNRS / Orsay, France
allauzen@limsi.fr

Abstract

Most of neural language models use different kinds of embeddings for word prediction. While word embeddings can be associated to each word in the vocabulary or derived from characters as well as factored morphological decomposition, these word representations are mainly used to parametrize the input, *i.e.* the context of prediction. This work investigates the effect of using subword units (character and factored morphological decomposition) to build output representations for neural language modeling. We present a case study on Czech, a morphologically-rich language, experimenting with different input and output representations. When working with the full training vocabulary, despite unstable training, our experiments show that augmenting the output word representations with character-based embeddings can significantly improve the performance of the model. Moreover, reducing the size of the output look-up table, to let the character-based embeddings represent rare words, brings further improvement.

1 Introduction

Most of neural language models, such as n-gram models (Bengio et al., 2003) are word based and rely on the definition of a finite vocabulary $\mathcal{V}$. Therefore, a look-up table maps each word$w \in \mathcal{V}$ to a vector of real features, and is stored in a matrix. While this approach yields significant improvement for a variety of tasks and languages, see for instance (Schwenk, 2007) in speech recognition and (Le et al., 2012; Devlin et al., 2014; Bahdanau et al., 2014) in machine translation, it induces several limitations.

For morphologically-rich languages, like Czech or German, the lexical coverage is still an important issue, since there is a combinatorial explosion of word forms, most of which are hardly observed on training data. On the one hand, growing the look-up table is not a solution, since it would increase the number of parameters without having enough training examples for a proper estimation. On the other hand, rare words can be replaced by a special token. This acts as a word class merging very different words without any distinction, while using different word classes to handle out-of-vocabulary words (OOVs) (Allauzen and Gauvain, 2005) does not really solve this issue, since rare words are difficult to classify. Moreover, for most inflected or agglutinative forms, as well as for compound words, the word structure is overlooked, wasting parameters for modeling forms that could be more efficiently handled by word decomposition into subwords units.

Using subword units, whether they are built via a different supervised method with embedded language knowledge, or from the training data, has been attempted many times, especially for speech recognition. The main goal is to reduce the OOV rate. While most of them were focused on a specific language, (Creutz et al., 2007) is a representative example of such a model applied to several morphologically-rich languages.

One of the first occurrences of general language models integrating morphological features to represent words are the *factored language model* (Bilmes and Kirchhoff, 2003) and its neural version (Alexandrescu and Kirchhoff, 2006). Input words are represented by their embedding, plus several other features, some of which include morphemes. To alleviate the impact of OOVs, (Mueller and Schuetze, 2011) used morphological features for class-based predictions when input words are unknown, obtaining state-of-the-art

Proceedings of the First Workshop on Subword and Character Level Models in NLP, pages 1–13,
Copenhagen, Denmark, September 7, 2017.

results on English. More recently, several types of language models represent words as function of subwords units: using a recursive structure (Luong et al., 2013), or an additive one (Botha and Blunsom, 2014). Quite a lot of work has been made on language models that extract features directly from the character sequence, whether they use character n-grams (Sperr et al., 2013), or characters composed by a convolutional layer (Santos and Zadrozny, 2014; Kim et al., 2015) or a Bi-LSTM layer (Ling et al., 2015). This avoids using an external morphological analyser. We can note that these types of models have also been applied with success to several other task, including learning word representations (Qiu et al., 2014; Cotterell et al., 2016; Bojanowski et al., 2016; Wieting et al., 2016), POS tagging (Plank et al., 2016; Ma and Hovy, 2016; Heigold et al., 2017), Named entity recognition (Gillick et al., 2016), Parsing (Ballesteros et al., 2015) and Machine translation (Costa-jussà and Fonollosa, 2016). Recently, an exhaustive summary of previous work on word representation by composing subword units was presented in (Vania and Lopez, 2017). This work also compares the types of subword unit, how they are composed, and their impact on various morphological typologies.

While recurrent neural networks have shown excellent performances for character-level language modeling (Sutskever et al., 2011; Hermans and Schrauwen, 2013), the results of such models are usually worse than those that use word-level prediction, since they have to consider a far longer history of tokens to be able to predict the next one correctly. However, more recent work (Hwang and Sung, 2017) seems to obtain very satisfactory results with a supplementary word-level layer that allows a better processing of the longer history.

Our work focuses on replacing output word embeddings by representations built from subwords. To the best of our knowledge, such a model has only been proposed in (Józefowicz et al., 2016), which evaluates the use of convolutional and LSTM layers to build word representations for outputs words. They allow the model to trade size against perplexity, since their model performs worse than the classic softmax approach, but with far less parameters. We first propose to study the training of a language model which augments or completely replaces output words representations with character-based representations. We compare the effect of different architectures, as well as the effect of different input representations. Our results show that:

- When evaluating perplexity on the full training vocabulary, using an augmented output representation improves the model performance.
- Not using the look-up table for rare words also improves the model performance.

Finally, we describe a short experiment with factoring the output predictions using a morphological analysis, which we believe could lead to a facilitated word generation when combined with reinflexion models.

Our paper is organized as follows: Section 2 describes the general architecture of the language model, and of the representations used, as well as its training, Section 3 presents the experiments and Section 4 gives our results and discussion.

2 Language model

We use a recurrent neural language model (Mikolov et al., 2010). The input of the network is a sequence of words $S = (w_1, \ldots, w_{|S|})$. Given a fixed sized vocabulary $\mathcal{V}$, the language model outputs a multinomial distribution $P(w_i = j | w_1^{i-1}), \forall j \in \mathcal{V}$ for each position i in the sequence, and with the prediction contexte $w_1^{i-1} = w_1, \ldots, w_{i-1}$. This allows us to compute the following probability :

$$P(w_1, \ldots, w_{|S|}) = \prod_{i=1}^{|S|} P(w_i | w_1^{i-1})$$

Our model uses the LSTM variant (Hochreiter and Schmidhuber, 1997). The hidden state $\mathbf{h}_i$ will be computed using the previous hidden state and a computed representation $\mathbf{r}_{w_i}$ of the word in position i in the sequence:

$$\mathbf{h}_i = LSTM(\mathbf{r}_{w_i}, \mathbf{h}_{i-1})$$

The conditional probability distribution of the next word is computed with a softmax function:

$$P(w_i = j | w_1^{i-1}) = \frac{\exp\left(\mathbf{h}_i \mathbf{r}_j^{out} + b_j\right)}{\sum_{k \in \mathcal{V}} \exp\left(\mathbf{h}_i \mathbf{r}_k^{out} + b_k\right)} \quad (1)$$

We propose to improve output word embeddings by using representations built from subwords, as it is often done for input words.

Usually, input and output word embeddings are parameters, stored in look-up matrices $\mathbf{W}$ and $\mathbf{W}_{out}$. The word embedding $\mathbf{r}_w^{word}$ of a word w is simply the column of $\mathbf{W}$ corresponding to its index in the vocabulary $\mathcal{V}$:

$$\mathbf{r}_w^{word} = [\mathbf{W}]_w$$

2.1 Representing words

We consider two other types of representations: decomposition of the words into characters (or n-grams of characters), and decomposing them into a Lemma and positional tags using a morphological analysis. An example of these different decompositions is shown in table 1.

Representation	Decomposition
Word	počátku
Characters	p+o+č+á+t+k+u
Character 3-grams	poč+očá+čát+átk+tku
Lemma + Tags	počátek+N+MascIn+Sg+Loc+Act

Table 1: Example of subword decompositions used for Czech word

2.1.1 Character-based representations

A word w is a character sequence $\{c_1, .., c_{|w|}\}$ represented by their embeddings $\{\mathbf{r}_{c_1}^{char}, .., \mathbf{r}_{c_{|w|}}^{char}\}$, where $\mathbf{r}_{c_i}^{char} = [\mathbf{C}]_{c_i}$ denotes the vector associated to the character c_i. To infer a word embedding from its character embeddings, we use two different architectures:

First, a *convolution layer* (Waibel et al., 1990; Collobert et al., 2011), similar to layers used in (Santos and Zadrozny, 2014; Kim et al., 2015), applies a convolution filter $\mathbf{W}_{n_c}^{CNN}$ over a sliding window of n_c characters, producing local features:

$$x_{n_c}^n = \mathbf{W}_{n_c}^{CNN}(\mathbf{r}_{c_{n-n_c+1}}^{char} : .. : \mathbf{r}_{c_n}^{char})^T + \mathbf{b}_{n_c}^{CNN}$$

where $x_{n_c}^n$ is a vector obtained for each position n in the word. The embeddings of w is then obtained by applying a max-pooling and the activation function ϕ:

$$[\mathbf{r}_w^{n_c}]_i = \phi\left(\max_{n=1}^{|w|-n_c+1} [\mathbf{x}_{n_c}^n]_i\right) \quad (2)$$

We can use multiple filters of n_{c_f} different sizes and concatenate their results:

$$\mathbf{r}_w^{CharCNN} = (\mathbf{r}_w^{n_{c_1}} : \ldots : \mathbf{r}_w^{n_{c_f}}) \quad (3)$$

Our second method uses a *bi-LSTM* (Hochreiter and Schmidhuber, 1997; Graves et al., 2005), on characters, similarly to (Ling et al., 2015). It combines the final states $\overrightarrow{\mathbf{h}_{|w|}}$ and $\overleftarrow{\mathbf{h}_1}$ of two LSTMs, respectively over the character sequence and the reverse character sequence, which are computed as such:

$$\overrightarrow{\mathbf{h}_i} = LSTM(\mathbf{r}_{c_i}^{char}, \overrightarrow{\mathbf{h}_{i-1}})$$

$$\overleftarrow{\mathbf{h}_j} = LSTM(\mathbf{r}_{c_j}^{char}, \overleftarrow{\mathbf{h}_{j+1}})$$

$$\mathbf{r}_w^{CharBiLSTM} = \overrightarrow{\mathbf{h}_{|w|}} : \overleftarrow{\mathbf{h}_1} \quad (4)$$

2.1.2 Lemma+Tags decomposition

For morphologically-rich languages, the different morphological properties of a word (gender, case, ...) are usually encoded using mutliple tags as shown in table 1. Therefore a word w is decomposed into a lemma l along with a set of associated sub-tags $T = \{t_1, .., t_{|T|}\}$ of fixed size $|T|$. For a given word, a single tag can be simply created by the concatenation of the subtags. However, this implies a large tagset and mitigates the generalization power since some sub-tags combinations can remain unobserved on training data. In this work we prefer a factored representation where each sub-tags is considered independently.

Lemmas, similarly to surface forms, are represented by $|\mathcal{V}_L|$ vectors stored in a look-up matrix $\mathbf{L}$, and $\mathbf{r}_l^{lemma} = [\mathbf{L}]_l$. For every words, each sub-tag has its own vocabulary and its own look-up matrix. However, the additional cost is negligible given their small size (see table 3). To infer a word embedding from a sub-tags set, we also use two methods. First, we simply concatenate their embeddings:

$$\mathbf{r}_T^{TagConcat} = \mathbf{r}_{t_1}^{tag_1} : \ldots : \mathbf{r}_{t_i}^{tag_i} : \ldots : \mathbf{r}_{t_{|T|}}^{tag_{|T|}} \quad (5)$$

The second method uses a bidirectionnal LSTM on the sequence of tags T, using exactly the same structure as in section 2.1.1:

$$\mathbf{r}_T^{TagBiLSTM} = \overrightarrow{\mathbf{h}_{|T|}} : \overleftarrow{\mathbf{h}_1} \quad (6)$$

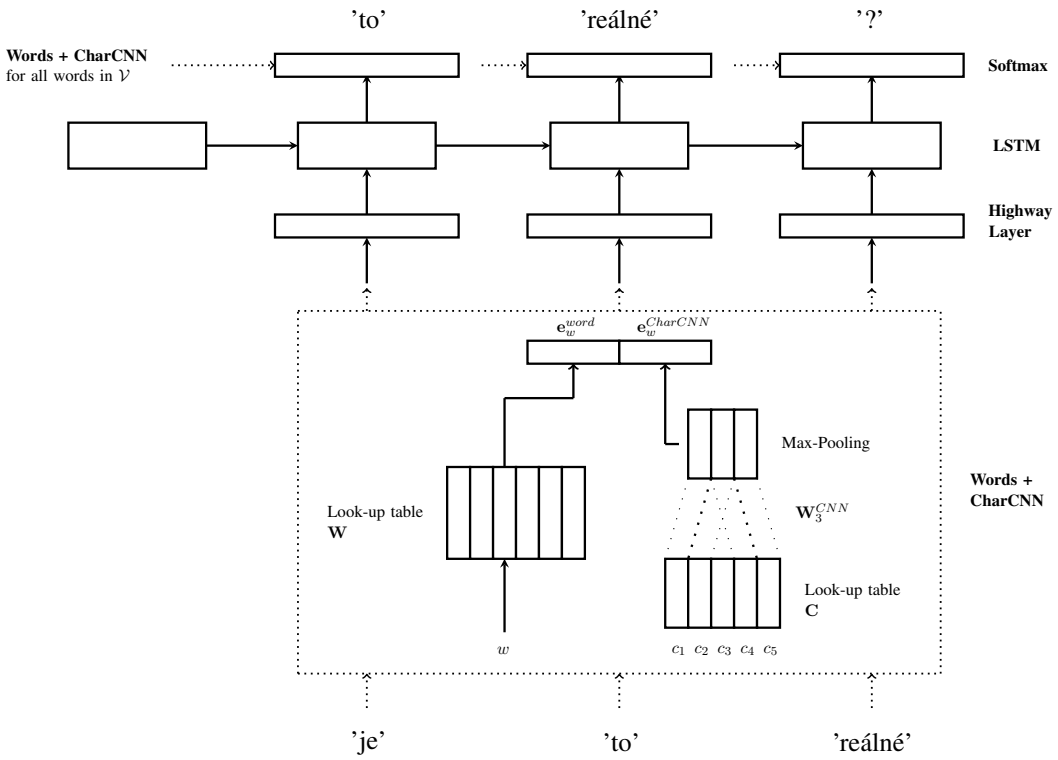

Figure 1: Example architecture of our language model, when using word embeddings and a character CNN to build both input and output word representations.

2.2 Training

Our final model, as illustrated in figure 1, uses concatenation of word, character-based or lemma and tags embeddings, to obtain input and output word representations. Following (Kim et al., 2015), we used a Highway layer (Srivastava et al., 2015) to model interactions between concatenated embeddings of various sources.

Usually, such a model is trained by maximizing the log-likelihood. For a given word w_i given its preceding sequence $w_1, \ldots, w_{i-1}$, the model parameters θ are estimated in order to maximize the following function for all the sequences observed in the training data:

$$LL(\theta) = \sum_{i=1}^{|S|} \log P_\theta(w_i | w_1^{i-1}) \qquad (7)$$

This objective function implies a very costly summation imposed by the softmax activation of the output layer: large output vocabularies cause a computational bottleneck due to the output normalization.

Different solutions have been proposed, as *shortlists* (Schwenk, 2007), *hierarchical softmax* (Morin and Bengio, 2005; Mnih and Hinton, 2009; Le et al., 2011), or self-normalisation techniques (Devlin et al., 2014; Andreas et al., 2015; Chen et al., 2016). Sampling-based techniques explore a different solution, where a limited number of negative examples are sampled to reduce the normalization cost. Working with a large vocabulary, and with output representations potentially more costly to compute, we choose to use the following sampling-based training algorithms:

- Target sampling, which is based on importance sampling (Bengio and Sénécal, 2008; Jean et al., 2015), directly approximates the normalization over $\mathcal{V}$ by normalizing over a sampled subset.

  Indeed, the gradient of the objective described in equation 7 is written as:

$$\frac{\partial}{\partial\theta} \log P_\theta(w_i | w_1^{i-1}) = \frac{\partial}{\partial\theta}(\mathbf{h}_i \mathbf{r}_{w_i}^{out} + b_{w_i}) - \mathbb{E}_{w \sim P_\theta(.|w_1^{i-1})} \left[\frac{\partial}{\partial\theta}(\mathbf{h}_i \mathbf{r}_{w_i}^{out} + b_{w_i}) \right] \qquad (8)$$

  The idea is to approximate the expectation of the second term by importance sampling a subset of $\mathcal{V}$ from a proposal distribution $\mathcal{Q}$. Target sampling implies associating with a part

$\mathcal{D}_i$ of the training data a subset $\mathcal{V}_i$ of $\mathcal{V}$ that corresponds to the target words of $\mathcal{D}_i$ plus a small subset of the remaining words. The resulting objective is equivalent to approximating the probability computed in equation 1 by normalizing it only over $\mathcal{V}_i$.

- Noise contrastive estimation (NCE), introduced in (Gutmann and Hyvärinen, 2012; Mnih and Teh, 2012), aims to discriminate between one example sampled from the real data $\mathcal{D}$ and k from a noise distribution P_n, and results in the model being theoretically unnormalized. The idea is to sample examples according to a mixture:

$$P(w|w_1^{i-1}) = \frac{1}{k+1}P_{\mathcal{D}}(w|w_1^{i-1}) + \frac{k}{k+1}P_n(w|w_1^{i-1}) \quad (9)$$

and train the model to recover whether the sample came from the data or the noise distribution. This is done by minimizing the binary cross-entropy of recognizing the current sample's origin, using the posterior probabilities:

$$P(w \sim P_{\mathcal{D}}|w, w_1^{i-1}) = \frac{P_\theta(w|w_1^{i-1})}{P_\theta(w|w_1^{i-1}) + kP_n(w|w_1^{i-1})} \quad (10)$$

$$P(w \sim P_n|w, w_1^{i-1}) = 1 - P(w \sim P_{\mathcal{D}}|w, w_1^{i-1}) \quad (11)$$

Besides, the probabilities intervening in equation 10 can be replaced by unnormalized scores at training time, since we can consider normalizing quantities as parameters to be learned.

- BlackOut (Ji et al., 2015), also approximating the normalization computation, with a weighted sampling scheme and a discriminative objective.It can be considered as a variant from NCE where we sample a set of k examples S_k from a proposal distribution $\mathcal{Q}$. We then proceed to apply NCE with a re-weighted noise distribution

$$P_n(w|w_1^{i-1}) = \frac{1}{k}\sum_{w_j \in S_k}\frac{\mathcal{Q}(w_j)}{\mathcal{Q}(w)}P_\theta(w_w|w_1^{i-1}) \quad (12)$$

which empirically behaves far better than NCE, providing an improved stability. BlackOut can also be linked to Importance sampling.

Ultimately, these three algorithms approximate the negative log-likelihood computed on a number k of negative samples from $\mathcal{V}$, using an easy to sample distribution.

3 Experiments

Experiments are carried out on Czech, a morphologically rich language using the different criteria described in section 2.2.

3.1 Data

We used data from the parallel corpus *News-commentary* 2015, from the WMT News MT Task. The data consists in 210K word sequences, amounting in about 4,7M tokens. We divided the data into a training, development and testing sets, these last two amounting to 150K tokens each. In our experiments, we use different vocabulary sizes by varying the frequency threshold: words are selected when their frequency in the training data are stricly higher than the threshold. Table 2 shows the correspondences between vocabulary sizes and these thresholds.

f_{Th}	$\lvert\mathcal{V}_{Th}\rvert$
0 (All words)	159142
1	66743
5	37010
10	25295

Table 2: Vocabulary sizes for different frequency thresholds

The lemma and tags decomposition presented in section 2.1.2 were obtained with Morphodita (Straková et al., 2014). There is 12 tag categories for Czech. Vocabulary sizes for characters, lemma and tags are detailed in table 3.

$\lvert\mathcal{V}_C\rvert$	$\lvert\mathcal{V}_L\rvert$	$\lvert\mathcal{V}_{tag_i}\rvert_{i=1..\lvert T\rvert}$
155	61364	[12, 65, 11, 6, 9, 6, 3, 5, 5, 4, 3, 3]

Table 3: Vocabulary sizes for subword units

3.2 Setup

The different versions of our model used in experiments are shown in table 4. We used a Highway layer when there is a concatenation of embeddings of different sources, which is for almost all architectures. We tried applying a Highway layer to the output representation, but it seemed

almost always counter-productive, rendering training more unstable. In all experiments presented here, weights are not tied between input and output representations, since our preliminary experiments with tied weights always gave worst results. Besides, we didn't mix structures for character-level representations (for example, using an input CharCNN and output CharLSTM) since our first experiments gave systematically worse results than using the same structures). When using different types of representations, we kept consistency between vocabularies: if both lemmas and words are used in a model, any lemma considered unknown will have its corresponding word unknown, and inversely. The same (or corresponding) vocabularies are used for inputs, outputs, and evaluation. The only exception is presented in section 4.4. When using a character-based output representation, during evaluation, the *unknown* token is built from a specific character token, a specific lemma token, and 12 specific tag tokens that are parameters of the model.

Input representation	$\mathbf{r}_w$	Eq
Words	$\mathbf{r}_w^{word}$	
CharCNN	$Hw(\mathbf{r}_w^{CharCNN})$	3
CharBiLSTM	$Hw(\mathbf{r}_w^{CharBiLSTM})$	4
Words + CharCNN	$Hw(\mathbf{r}_w^{word} : \mathbf{r}_w^{CharCNN})$	
Words + CharBiLSTM	$Hw(\mathbf{r}_w^{word} : \mathbf{r}_w^{CharBiLSTM})$	
Lemma + Tags Concat.	$Hw(\mathbf{r}_l^{lemma} : \mathbf{r}_T^{TagConcat})$	5
Lemma + TagsBiLSTM	$Hw(\mathbf{r}_l^{lemma} : \mathbf{r}_T^{TagBiLSTM})$	6
Output representation	$\mathbf{r}_w^{out}$	
Words	$\mathbf{r}_w^{word}$	
Words + CharCNN	$\mathbf{r}_w^{word} : \mathbf{r}_w^{CharCNN}$	
Words + CharBiLSTM	$\mathbf{r}_w^{word} : \mathbf{r}_w^{CharBiLSTM}$	
Lemmas	$\mathbf{r}_l^{lemma}$	
Lemmas + CharCNN	$\mathbf{r}_l^{lemma} : \mathbf{r}_l^{CharCNN}$	
Lemmas + CharLSTM	$\mathbf{r}_l^{lemma} : \mathbf{r}_l^{BiLSTM}$	

Table 4: Detail of input and output representations used in our experiments. Hw designate the use of a Highway layer

Our experiments aim at comparing potential use of subword-based word representation, and thus are not directed towards performance. For this reason, we used the same implementation for all experiments and did not specifically try to optimize the general model structure or the dimensional hyperparameters, neither compared our results with benchmarks on Czech corpora.

3.3 Training and evaluation

Language models are evaluated with perplexity:

$$PPL = \exp\left(\sum_{i=1}^{|S|} \frac{-\log P_\theta(w_i|w_1^{i-1})}{|S|}\right)$$

over all sequences in the testing data. Perplexity is computed for a fixed output vocabulary $\mathcal{V}$, which allows to compare models using the same output vocabulary. However, we can't evaluate model performance on out-of-vocabulary words, since those are to be classified as the unknown token in $\mathcal{V}$.

Our models are implemented with Tensorflow (Abadi et al., 2015). We use the Adam algorithm (Kingma and Ba, 2014) with an initial learning rate of $5 * 10^{-4}$ for training, over a maximum of 10 epochs, with a batch size of 128 sequences. However, since the training is often unstable, the model backtracks to the last checkpoint if it does not improve its performance on validation data after $1/10$ of an epoch, and stop training after 10 unsuccessful loadings in a row. To avoid overfitting, we use dropout with probability 0.5 on recurrent layers, and $L2$ regularization on feedforward layers.

We use two hidden layers, and choose our embeddings dimensions in order to obtain, for each type of representation, an embedding dimension of 150. In the case of the CNN, we used filters of 3, 5 and 7 characters, of dimension 30, 50, and 70. Whether we use NCE, blackOut, or importance sampling, we draw $k = 500$ noise samples by batch. For all experiments, we report the perplexity on test data at the end of training. Results presented in tables 5, 6, 7 are the average of the results obtained on 5 models, and the standard deviation.

4 Results

4.1 Influence of the vocabulary size

We first train our model with different vocabulary sizes. As shown in figure 2, our model fails to improve upon the conventional word model when the output vocabulary size is relatively small (shown on the two leftmost graphs). More precisely, models that use word and character-based representations at the output seem unable to learn after a couple of iterations. We first link this behaviour to the difficulty met by the authors in (Józefowicz et al., 2016): since most logits are tied when we use an output character-based representation - as

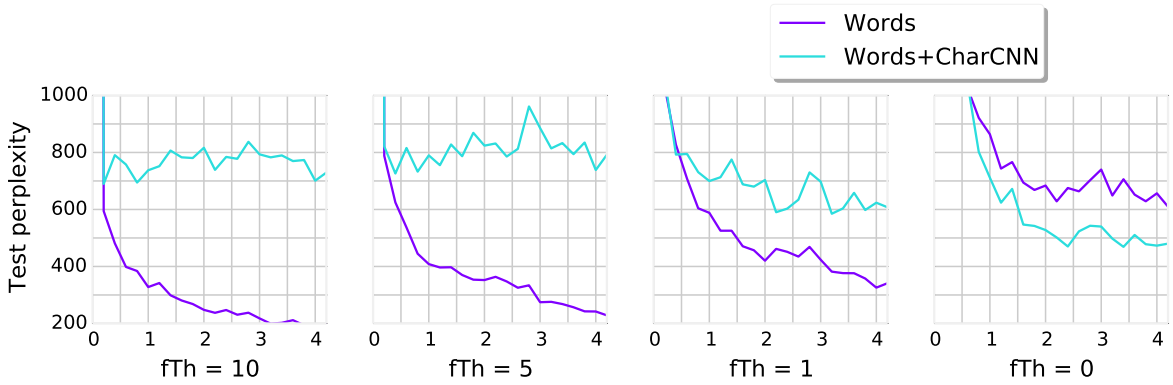

Figure 2: Test perplexities obtained when training models using Words as input representation and Words+CharCNN as output representations, for various vocabulary sizes. Corresponding vocabulary sizes are given in table 2. The models are trained with target sampling.

opposed to independently learned word embeddings, the function mapping from word to word representation is smoother and training becomes more difficult. They used a smaller learning rate and a low dimensional correction factor, learned for each word, as a work-around.

However, increasing the vocabulary size reduces this effect . This is especially clear with the whole training vocabulary (on the rightmost graph of figure 2): in this setup, using a character-based representation improves the performance of the model. We can assume that, for rare words, learning independent embeddings fails since scarce updates of these embeddings are insufficient. For the rare words, combining word and character-based embeddings allows the model to better counteract the sparsity issue.

4.2 Choice of the training criterion

Given the previous results, we use the full training vocabulary to assess the impact of the training criterion. However, using this full training vocabulary renders training very unstable, especially with sampling-based algorithms. Stability issues, especially for the Noise-contrastive estimation, have previously been discussed (Chen et al., 2016; Józefowicz et al., 2016). We shortly experimented to choose the most practical criterion to use. Figure 3 shows the shape of the training curves. While target sampling and blackOut both seem to work properly, NCE needs far more noise samples to converge. We believe this is related to the tensorflow implementation, which re-use the same noise samples for every example in the batch, which leads to a lack of diversity in negative

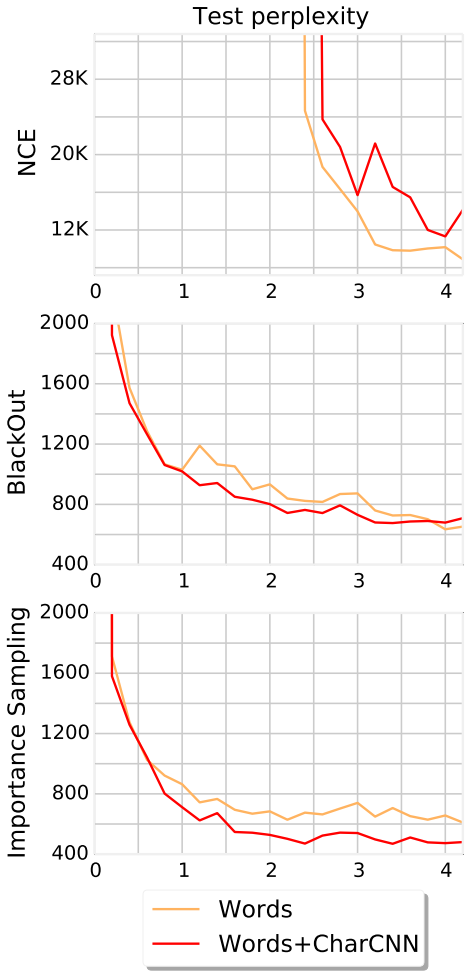

Figure 3: Test perplexities obtained when training models using Words as input representation and Words+CharCNN as output representations, for various training methods: Noise contrastive estimation, blackOut and Target sampling.

examples. Augmenting the number of samples or reducing the size of the batch are possible solutions, but they increase the training time. BlackOut obtains better results because, while very similar to NCE, the scores used as coming from the noise distribution are context-dependent, which brings diversity to negative examples.

Overall, across several experiments, target sampling performs better than blackOut, and we choose to use it for the rest of our experiments. Since training is still quite unstable, depending on the architecture, we report results across 5 trainings for the next sections.

Output Representation / Input Representation		Words	Words + Char	
			CNN	BiLSTM
Words		563 ± 53	432 ± 18	480 ± 31
Char	CNN	698 ± 41	543 ± 16	-
	BiLSTM	971 ± 24	-	938 ± 48
Words + Char	CNN	$\mathbf{495 \pm 34}$	$\mathbf{411 \pm 40}$	-
	BiLSTM	537 ± 24	-	$\mathbf{480 \pm 21}$
Lemmas + Tags	Concat.	521 ± 47	424 ± 22	502 ± 54
	BiLSTM	541 ± 8	445 ± 24	496 ± 39

Table 5: Average test perplexities obtained when training 5 models with target sampling, for various input/output representations. Results in bold are the best models for a given output representation.

4.3 Effects of the representation choice

Table 5 gathers the main experimental results to assess which combination of input and output representations gives the best performance. For any input representation, augmenting the output representation with a character-based embedding improves the performance of the model. It is especially true for convolutional layers. We also can notice that the improvement is better for models that performed badly with basic output word embeddings.

Overall, biLSTMs perform worse than their convolution/concatenation counterparts. Finally, the best average perplexity of 495 for word only output representations is improved to an average of 411 for augmented output representations.

Other output representations: First, our experiments with only character-based embeddings as output representations give results far worse than those reported in 5, with our best model obtaining an average perplexity of $\approx$ 2500. Training is also far more unstable. We believe these results are linked to the difficulties mentioned in (Józefowicz et al., 2016) and in section 4.1.

We also tried to use the lemma+tags decomposition presented in section 2.1.2, but without success. When tags were ambiguous across several occurrences of the same words, we tried using specific tokens, or choosing the most frequent tags, but in both cases the model severely overfits.

Finally, we tried to use word embeddings pretrained with word2vec (Mikolov et al., 2013) as output representations. We obtained results very similar to those of classical word embeddings, with a small but noticeable improvement when the input representation used LSTM. However, these improvements are still well under those obtained by augmenting the output representation with a character-based embedding.

4.4 Influence of the size of the word embeddings vocabulary

Input Representation	f_{Th}^{W}	Words + CharCNN		
		All words	Frequent words	Rare words
Words	0	432 ± 18	286 ± 13	$\mathbf{5170 \pm 1310}$
	1	415 ± 26	258 ± 13	6510 ± 780
	5	$\mathbf{390 \pm 20}$	$\mathbf{250 \pm 14}$	7210 ± 1290
	10	416 ± 20	265 ± 11	8100 ± 580
CharCNN	0	543 ± 16	348 ± 10	6400 ± 1480
	1	523 ± 17	328 ± 22	$\mathbf{5070 \pm 1080}$
	5	$\mathbf{478 \pm 16}$	$\mathbf{316} \pm 27$	7000 ± 1800
	10	488 ± 31	338 ± 25	8210 ± 2160
Words + CharCNN	0	411 ± 40	271 ± 30	$\mathbf{4470 \pm 190}$
	1	374 ± 10	$\mathbf{242 \pm 7}$	5020 ± 870
	5	$\mathbf{367 \pm 14}$	$\mathbf{241 \pm 8}$	5560 ± 1220
	10	393 ± 19	254 ± 13	6600 ± 1910
Lemma + TagsConcat.	All	449 ± 26	293 ± 16	5830 ± 760
	1	439 ± 34	287 ± 11	6220 ± 1080
	5	$\mathbf{408 \pm 32}$	$\mathbf{269 \pm 20}$	$\mathbf{4600 \pm 1280}$
	10	430 ± 32	$\mathbf{269 \pm 20}$	8410 ± 1140
Lemma + TagsBiLSTM	All	445 ± 24	288 ± 13	7150 ± 1560
	1	424 ± 34	281 ± 21	5380 ± 1000
	5	$\mathbf{387 \pm 9}$	$\mathbf{258 \pm 5}$	$\mathbf{4300 \pm 1390}$
	10	442 ± 17	287 ± 13	7390 ± 1690

Table 6: Test perplexity averaged on 5 models trained with target sampling, for various input representations and output word look-up table sizes. Corresponding vocabulary sizes are given in table 2. Test perplexities are given for all words, frequent words (frequency > 10) and rare words (frequency < 10). In bold are the best models for a given input representation.

Following our observations in section 4.1, we then assess the effect of reducing the word vocabulary size for Words+CharCNN output representation. We don't change the size of the event space: when constructing output representations for words under a chosen frequency, we simply don't use the word representation. For example, using a threshold of $f^W_{Th} = 10$ means that words that appear less thant ten times won't have their own word embedding, and will be represented by the unknown word token combined with their character-based representation. Results are shown in table 6. We can see that for all input representations, using a specific unknown token in place of a specific word embedding for words appearing less than 5 times in training data gives the best performance. Reducing the look-up table to words only appearing more than 10 times gives worse results, while they are still better than if we keep the full table. However, there is no clear trend when looking at the rare words perplexities, which are very hard to interpret, given their very high standard deviation. With a smaller output word look-up table, our best average perplexity of 411 is reduced to 376, which is a very sizeable overall improvement.

Output Representation / Input Representation		Lemmas	Lemmas + Char	
			CNN	BiLSTM
Words		240 ± 12	220 ± 9	222 ± 12
Char	CNNN	308 ± 15	270 ± 11	-
	BiLSTM	477 ± 17	-	429 ± 9
Words + Char	CNN	234 ± 9	$\mathbf{203 \pm 7}$	-
	BiLSTM	238 ± 6	-	225 ± 11
Lemmas + Tags	Concat.	239 ± 3	211 ± 5	217 ± 9
	BiLSTM	$\mathbf{232 \pm 5}$	$\mathbf{203 \pm 6}$	$\mathbf{212 \pm 6}$

Table 7: Test perplexities averaged on 5 models on lemmas with a multiple objectives cost function. Results are given for various input/output representations. In bold are the best models for a given output representation.

4.5 Predicting root and tags jointly

While using the lemma+tags decomposition to build output representation was not, in our experiments, successful, we investigated a factorised prediction of lemma and tags. We used different costs for predicting lemmas and each tag, which are summed into a final objective function. As recently seen in (Martinez et al., 2016; Burlot and Yvon, 2017), these objectives are individually easier when working with morphologically-rich languages, and fully inflected words can be obtained by using morphological inflection models, which have been shown to be quite successful (Faruqui et al., 2016; Kann et al., 2017).

Table 7 shows the test perplexities on lemmas for various input and output representations. We can observe that in all cases training is far more stable, with generally lower standard deviations. In this case, using a lemma+tags with a BiLSTM or a Words+CharCNN input representation both give the best results, while augmenting the output representation of the lemma with a character-build embedding also improves results. This makes the joint learning of a factored prediction and reinflection language model a very interesting direction for future work.

5 Conclusion

We described a neural language model allowing the use of subword units for both input and output word representations. While in our experiments training with a full vocabulary is unstable, we can identify important trends: augmenting output representations with character-based embeddings improves the model performance, and in this setup, replacing independent word embeddings by the unknown token for rare words yields further improvement. It is worth noticing that this also opens the vocabulary, since our model can be used to rescore unknown words. Additional experiments suggest that factoring the output of the model with a lemma+tags decomposition, then re-inflecting these into words, could make generation easier: this is a direction we plan to investigate.

Acknowledgements

We wish to thank the anonymous reviewers for their helpful comments. This work has been funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh

Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. http://tensorflow.org/.

Andrei Alexandrescu and Katrin Kirchhoff. 2006. Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. Association for Computational Linguistics, New York City, USA, pages 1–4. http://www.aclweb.org/anthology/N/N06/N06-2001.

A. Allauzen and J.L Gauvain. 2005. Open vocabulary asr for audiovisual document indexation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Jacob Andreas, Maxim Rabinovich, Michael I. Jordan, and Dan Klein. 2015. On the accuracy of self-normalized log-linear models. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 1783–1791. http://papers.nips.cc/paper/5806-on-the-accuracy-of-self-normalized-log-linear-models.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 349–359. http://aclweb.org/anthology/D15-1041.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.

Yoshua Bengio and Jean-Sébastien Sénécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Trans. Neural Networks* 19(4):713–722.

Jeff A. Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL-Short '03, pages 4–6. https://doi.org/10.3115/1073483.1073485.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR* abs/1607.04606. http://arxiv.org/abs/1607.04606.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of the International Conference of Machine Learning (ICML)*. Beijing, China.

Franck Burlot and François Yvon. 2017. Learning morphological normalization for translation from and into morphologically rich language. *The Prague Bulletin of Mathematical Linguistics (Proc. EAMT)* (108):49–60.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1975–1985. http://www.aclweb.org/anthology/P16-1186.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* 12:2493–2537. http://dl.acm.org/citation.cfm?id=1953048.2078186.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 357–361. http://anthology.aclweb.org/P16-2058.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1651–1660. http://www.aclweb.org/anthology/P16-1156.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Trans. Speech Lang. Process.* 5(1):3:1–3:29.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul.

2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1370–1380. http://www.aclweb.org/anthology/P14-1129.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 634–643. http://www.aclweb.org/anthology/N16-1077.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1296–1306. http://www.aclweb.org/anthology/N16-1155.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional LSTM networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, 15th International Conference, Warsaw, Poland, September 11-15, 2005, Proceedings, Part II*. pages 799–804.

Michael U. Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.* 13(1):307–361.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 505–513. http://www.aclweb.org/anthology/E17-1048.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 190–198. http://papers.nips.cc/paper/5166-training-and-analysing-deep-recurrent-neural-networks.pdf.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Kyuyeon Hwang and Wonyong Sung. 2017. Character-level language modeling with hierarchical recurrent neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*. pages 5720–5724. https://doi.org/10.1109/ICASSP.2017.7953252.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1–10. http://www.aclweb.org/anthology/P15-1001.

Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. 2015. Blackout: Speeding up recurrent neural network language models with very large vocabularies. *CoRR* abs/1511.06909. http://arxiv.org/abs/1511.06909.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR* abs/1602.02410. http://arxiv.org/abs/1602.02410.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. Neural multi-source morphological reinflection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 514–524. http://www.aclweb.org/anthology/E17-1049.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Montréal, Canada, pages 39–48. http://www.aclweb.org/anthology/N12-1005.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and Francois Yvon. 2011. Structured output layer neural network language model. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Prague, Czech Republic, pages 5524–5527.

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. http://aclweb.org/anthology/D15-1176.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, pages 104–113. http://www.aclweb.org/anthology/W13-3512.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1064–1074. http://www.aclweb.org/anthology/P16-1101.

Mercedes Garcia Martinez, Loc Barrault, and Fethi Bougares. 2016. Factored neural machine translation architectures. In *International Workshop on Spoken Language Translation (IWSLT'16)*. Seattle (USA).

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*. pages 1045–1048.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, Curran Associates, Inc., pages 1081–1088. http://papers.nips.cc/paper/3583-a-scalable-hierarchical-distributed-language-model.pdf.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *ICML*. icml.cc / Omnipress.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, pages 246–252. http://www.iro.umontreal.ca/ lisa/pointeurs/hierarchical-nnlm-aistats05.pdf.

Thomas Mueller and Hinrich Schuetze. 2011. Improved modeling of out-of-vocabulary words using morphological classes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 524–528. http://www.aclweb.org/anthology/P11-2092.

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. http://anthology.aclweb.org/P16-2067.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 141–150. http://www.aclweb.org/anthology/C14-1015.

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. JMLR Workshop and Conference Proceedings, pages 1818–1826. http://jmlr.org/proceedings/papers/v32/santos14.pdf.

Holger Schwenk. 2007. Continuous space language models. *Comput. Speech Lang.* 21(3):492–518.

Henning Sperr, Jan Niehues, and Alex Waibel. 2013. Letter n-gram-based input encoding for continuous space language models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*. Association for Computational Linguistics, Sofia, Bulgaria, pages 30–39. http://www.aclweb.org/anthology/W13-3204.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. *CoRR* abs/1507.06228. http://arxiv.org/abs/1507.06228.

Jana Straková, Milan Straka, and Jan Hajič. 2014. Open-source tools for morphology, lemmatization, pos tagging and named entity recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 13–18. http://www.aclweb.org/anthology/P14-5003.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural net-

works. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. ACM, New York, NY, USA, ICML '11, pages 1017–1024.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? *CoRR* abs/1704.08352. http://arxiv.org/abs/1704.08352.

Alexander Waibel, Toshiyuki Hanazawa, Geofrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. 1990. *Readings in Speech Recognition*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, chapter Phoneme Recognition Using Time-delay Neural Networks, pages 393–404.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1504–1515. https://aclweb.org/anthology/D16-1157.