SEG-LM: A Factored-Output Model for Morphologically Rich Languages

Yash Shah[*] and **Ishan Tarunesh**[*] and **Preethi Jyothi**
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
{yashshah,ishan,pjyothi}@cse.iitb.ac.in

Abstract

Neural language models (LMs) have shown to benefit significantly from enhancing word vectors with subword-level information, especially for morphologically rich languages. This has been mainly tackled by providing subword-level information as an input; using subword units in the output layer has been far less explored. In this work, we propose a factored output LM that predicts a mixture of two output layers, one that is purely-word based and the other that predicts a sequence of subwords acquired using a new unsupervised word segmentation algorithm. The mixture weights are jointly learned, along with the word and subword probabilities. We also introduce a separate loss term during training that helps determine which output layer is more informative for a given word. We focus on two morphologically complex Indian languages, Hindi and Tamil, and observe significant perplexity gains on both using the factored output models when compared with a competitive baseline that incorporates subword level information in the output layer.

1 Introduction

Language modeling is a fundamental problem in NLP that involves predicting the next word given its context. Recurrent neural network language models (RNNLMs) have become the de facto standard for language modeling. They typically produce a next-word probability distribution over a fixed vocabulary of words. Such an approach has two main limitations. Word embeddings for infrequently occurring words in training data are poorly estimated. Also, predictions at word level are largely immune to the subword structure in words. Both these limitations are exacerbated for morphologically rich languages in which words have numerous morphological variants, leading to large vocabularies where a significant fraction of words appear in the long tail of the word distribution. Leveraging subword information becomes especially important for such languages.

In prior work, RNNLMs have typically exploited subword-level information at the input side and learn improved word embeddings by utilizing morpheme- and character-level information. Vania and Lopez (2017) present an exhaustive comparison of many such methods. Incorporating subword information within the output layer of RNNLMs has received less attention. We explore this direction and make the following specific contributions:

- We present SEG-LM, a mixture model-based LM with a factored output layer that makes use of an unsupervised word segmentation algorithm to identify subword units.
- We use our segmentation algorithm (that decides whether or not a word should be segmented) as a form of supervision over the mixture weights and add a separate loss term to penalize deviations from these labels.
- We demonstrate the effectiveness of our proposed approach by showing large reductions in perplexity on two morphologically rich languages, Hindi and Tamil, compared to a competitive baseline.

2 SEG-LM: Model Description

For a given word w_t, an RNNLM encodes its context $w_1, \ldots, w_{t-1}$ into a fixed-size representation, $\mathbf{h}_t$. An RNNLM predicts w_t by applying a softmax function to an affine transformation of $\mathbf{h}_t$ ($\mathbf{W}$ and $\mathbf{b}$ are model parameters):

$$\Pr(w_t | w_1, \ldots, w_{t-1}) = \text{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b})$$

[*] Joint first authors

We depart from this standard formulation and use a mixture of K different softmax distributions at the output layer. Compared to using a single softmax distribution, mixture models have led to LMs with improved generalization abilities that translate to substantial reductions in test perplexities (Neubig and Dyer, 2016; Matthews et al., 2018; Yang et al., 2018). If $\lambda_t \in [1, K]$ denotes the model at time t, the next-word distribution becomes:

$$\Pr(w_t|\{w_i\}_{i=1}^{t-1}) = \sum_{k=1}^{K} \Pr(w_t, \lambda_t = k|\mathbf{h}_t)$$
$$= \sum_{k=1}^{K} \Pr(\lambda_t = k|\mathbf{h}_t) \Pr(w_t|\lambda_t = k, \mathbf{h}_t) \quad (1)$$

We define a mixture model with two components: 1) Generates words from a fixed vocabulary 2) Generates a sequence of subword units for a word, that are discovered using an unsupervised word segmentation algorithm. We present four variants of this mixture model in the rest of this section.

2.1 SEG: Unsupervised word segmentation

Prior work has shown improved LM performance by using language-specific morphological analyzers (Matthews et al., 2018). Since it may not always be feasible to easily obtain access to morphological analyzers for a language, we propose an unsupervised word segmentation algorithm to discover subword units in a word.

We assume a word could either remain unsegmented or be split into a prefix, stem and suffix.[1] In our mixture model, we define two probability distributions for w_t, $\Pr(w_t|\lambda_t = 1, \mathbf{h}_t)$ and $\Pr(w_t|\lambda_t = 2, \mathbf{h}_t)$. $\Pr(w_t|\lambda_t = 1, \mathbf{h}_t)$ is the standard softmax distribution over the entire word vocabulary. $\Pr(w_t|\lambda_t = 2, \mathbf{h}_t)$ is a product of probabilities estimated over the prefix, stem and suffix that w_t is segmented into. If w_t is written as a tuple $(w_t^{\text{pre}}, w_t^{\text{st}}, w_t^{\text{suf}})$, then:

$$\Pr(w_t|\lambda_t = 2, \mathbf{h}_t) = \Pr(w_t^{\text{pre}}, w_t^{\text{st}}, w_t^{\text{suf}}|\lambda_t = 2, \mathbf{h}_t)$$
$$= \Pr(w_t^{\text{st}}|\lambda_t = 2, \mathbf{h}_t) \Pr(w_t^{\text{suf}}|w_t^{\text{st}}, \lambda_t = 2, \mathbf{h}_t)$$
$$\Pr(w_t^{\text{pre}}|w_t^{\text{suf}}, w_t^{\text{st}}, \lambda_t = 2, \mathbf{h}_t) \quad (2)$$

Several unsupervised word segmentation algorithms have been previously developed to discover the morphology of a language (Goldsmith, 2001; Creutz and Lagus, 2002; Pitler and Keshava, 2007; Cotterell et al., 2016). However, in this work, we did not examine the impact of changing the segmentation algorithm on LM performance. We leave this exploration for future work.

Algorithm 1 describes the pseudocode of our unsupervised segmentation method. L_p and L_s refer to all prefixes and suffixes of words in the vocabulary, which we implemented using forward and backward tries. We create $(+s_i, -s_j)$ suffix pairs such that removing suffix s_j followed by addition of suffix s_i to a word forms another valid word.[2] Frequently occurring pairs are chosen to form the set of prefix/suffix rules governing segmentation. Finally, of all possible splits for a word, the one with the most frequent stem is selected as its canonical segmentation.

We use the standard cross-entropy loss function computed over all training tokens (indexed by $t = 1, \ldots, T$), $\mathcal{L}_{\text{SEG}} = -\frac{1}{T}\sum_{t=1}^{T} \log \Pr(w_t|\mathbf{h}_t)$ where $\Pr(w_t|\mathbf{h}_t)$ can be substituted as in Eqn 1.

Algo 1: Unsupervised word segmentation

```
Input: vocabulary $\mathcal{V}$, threshold parameters $\delta_s$ and $\delta_p$
Output: rules $\mathcal{R}_\text{s}$ and $\mathcal{R}_\text{p}$, word splits $\mathcal{S}$
$L_\text{s} \leftarrow$ SUFFIXES($\mathcal{V}$), $L_\text{p} \leftarrow$ PREFIXES($\mathcal{V}$) ;
for $(s_1, s_2) \in L_\text{s}^2$ s.t. $s_1 < s_2$ do
    $\text{wt}_s[s_1, s_2] \leftarrow |\{(w_1, w_2) \in \mathcal{V}^2 :;$
                $\exists u, w_1 = u + s_1, w_2 = u + s_2\}|$
end
$\mathcal{R}_\text{s} \leftarrow \{(s_1, s_2) \mid \text{wt}_s[s_1, s_2] \geq \delta_s$ or $s_1 = s_2 = \epsilon\}$;
for $(p_1, p_2) \in L_\text{p}^2$ s.t. $p_1 < p_2$ do
    $\text{wt}_p[p_1, p_2] \leftarrow |\{(w_1, w_2) \in \mathcal{V}^2 :;$
                $\exists u, w_1 = p_1 + u, w_2 = p_2 + u\}|$
end
$\mathcal{R}_\text{p} \leftarrow \{(p_1, p_2) \mid \text{wt}_p[p_1, p_2] \geq \delta_p$ or $p_1 = p_2 = \epsilon\}$;
$\mathcal{R} \leftarrow \{(v, w) \in \mathcal{V}^2 | \exists (p_1, p_2) \in \mathcal{R}_\text{p}, (s_1, s_2) \in \mathcal{R}_\text{s}$ ;
    and $u$, s.t. $v = p_1 + u + s_1, w = p_2 + u + s_2\}$ ;
for $v \in \mathcal{V}$ do
    $\text{wt}_{\text{stem}}[v] \leftarrow |\{w \in \mathcal{V} \mid (v, w) \in \mathcal{R}\}|$ ;
end
for $w \in \mathcal{V}$ do
    stem $\leftarrow \arg\max_{v:(v,w)\in\mathcal{R}} (\text{wt}_{\text{stem}}[v])$ ;
    Let $u$, $(p_1, p_2) \in \mathcal{R}_\text{p}$, $(s_1, s_2) \in \mathcal{R}_\text{s}$ be s.t.;
        stem $= p_1 + u + s_1, w = p_2 + u + s_2$ ;
    $\mathcal{S}(w) \leftarrow (p_2, \text{stem}, s_2)$ ;
end
```

2.2 SEG+BCE: Additional loss term

For every word w_t, we define a binary variable α_t that is set to 0 when it remains unsegmented according to Algorithm 1 and is set to 1 otherwise. To help the mixture weights learn whether or

[1]We also experiment with dropping prefixes altogether and only using a stem+suffix combination.

[2]For every rule $(+a_i, -a_j)$, we enforce $a_i < a_j$ which is true if $|a_i| < |a_j|$, or $|a_i| = |a_j|$ and a_i is lexicographically smaller than a_j.

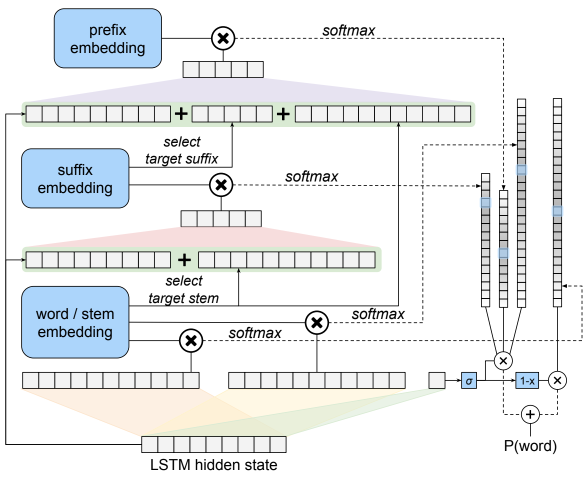

Figure 1: Schematic diagram illustrating our model architecture.

not the current word should be segmented, we add a binary cross-entropy term $\mathcal{L}_{\text{BCE}}$ to the loss function: $\mathcal{L}_{\text{SEG+BCE}} = \mathcal{L}_{\text{SEG}} + \eta \mathcal{L}_{\text{BCE}}$. Here, $\mathcal{L}_{\text{BCE}} = -\frac{1}{T} \sum_{t=1}^{T} \alpha_t \log P(\lambda_t = 2|\mathbf{h}_t) + (1 - \alpha_t) \log(1 - P(\lambda_t = 2|\mathbf{h}_t))$ and η is a tunable hyperparameter.

2.3 SEG+BCE+EMBED: Augment input word embeddings

Incorporating side information as an input to an RNNLM gives consistent improvements in performance (Hoang et al., 2016). We provide the bit α_t as a side input, along with every word w_t, and encode it as a fixed-size embedding.

Figure 1 shows a schematic diagram illustrating the output layers of our proposed framework. (The input side of our architecture is exactly as specified by Gerz et al. (2018).) Two separate affine transformations of $\mathbf{h}_t$ are learned, corresponding to unsegmented and segmented words; a softmax layer applied to these transformations gives $\Pr(w_t^{\text{st}}|\lambda_t = 1, \mathbf{h}_t)$ and $\Pr(w_t^{\text{st}}|\lambda_t = 2, \mathbf{h}_t)$, respectively. A third affine transformation followed by a sigmoid activation gives the mixture weights $\Pr(\lambda_t|\mathbf{h}_t)$. For a word $w_t = (w_t^{\text{pre}}, w_t^{\text{st}}, w_t^{\text{suf}})$, as shown in Eqn 2, its suffix probability is computed by conditioning on both w_t^{st} and $\mathbf{h}_t$. This is implemented by first concatenating the vector $\mathbf{h}_t$ with the embedding corresponding to w_t^{st} and applying softmax to an affine transformation of this concatenated vector. (Prefix probabilities are similarly learned by conditioning on both the stem and the suffix.)

Statistic	Language	
	Hi	Ta
# of training tokens	666,446	507,226
Vocabulary Size	50,384	106,403
Type / Token (train)	0.08	0.21
# of dev tokens	50,042	39,379
# of test tokens	49,125	39,661
OoV rate (test)	5.3%	15.2%

Table 1: Dataset statistics.

2.4 NOSEG: No word segmentation

In order to distinguish the improvements in performance obtained by using the segmentation algorithm from the improvements obtained by using a mixture of softmax distributions at the output layer without any segmentation, we train a model that uses a mixture of two softmax distributions both operating at the word-level.

3 Experiments

3.1 Dataset description

We use the Hindi (Hi) and Tamil (Ta) datasets, from Gerz et al. (2018), along with their specified training/dev/test splits. Table 1 shows detailed statistics for these two languages. Ta is more morphologically complex compared to Hi, which is apparent from the higher out-of-vocabulary (OoV) rate and higher type-token ratio in Table 1.

3.2 Implementation Details

We used PyTorch (Paszke et al., 2017) to implement all our models. We report two baseline numbers: (A) CHAR-CNN-LSTM: RNNLM proposed by Kim et al. (2016) that uses character-level inputs and works well on a variety of languages and (B) LMMRL: RNNLM proposed by Gerz et al. (2018) that improves over Kim et al. (2016) by finetuning the output embeddings to capture subword-level information. (Our reimplementations of these baselines produce much better numbers than those reported in Gerz et al. (2018))

The word/stem, prefix and suffix embeddings were of size 500, 40 and 40 for Hi and 800, 70 and 70 for Ta, respectively. For the SEG + BCE + EMBED model, we use an embedding of size 10 to encode α_t. All models were trained for 15 epochs using the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 5e-5 that was decayed by 0.8 every 5 epochs.

Model	Language	
	Hi	Ta
CHAR-CNN-LSTM	366.23	1871.87
LMMRL	366.21	1843.53
NOSEG(2)	268.56	1197.77
SEG(2)	249.59	952.39
SEG(2) + BCE	248.42	**815.73**
SEG(2) + BCE + EMBED	235.24	921.84
NOSEG(3)	275.46	1155.49
SEG(3)	270.67	1106.57
SEG(3) + BCE	265.95	1125.68
SEG(3) + BCE + EMBED	**228.68**	893.95

Table 2: Perplexity scores on the Hi and Ta test sets. (2) and (3) are used to differentiate between a stem/suffix and a prefix/stem/suffix segmentation, respectively.

Following Gerz et al. (2018), we set `<unk>`'s embedding, after every training epoch, to a sample drawn from a multivariate Gaussian distribution whose mean and variance are computed using the embeddings of all other words. The number of model parameters in LMMRL for Hi and Ta are 45.6M and 82.0M; the largest SEG-LM models for Hi and Ta contain 38.8M and 88.4M parameters, respectively.

3.3 Results and analysis

Table 2 summarizes test perplexity (ppl) scores on Hi and Ta. The NOSEG models give large reductions in ppl compared to the baselines, highlighting the benefits from using a mixture of softmax distributions.[3] Using the segmentation algorithm further improves ppl scores, when compared to NOSEG. Including the BCE loss, together with including α_t as part of the input gives the best ppl scores for each language (except in the case of SEG(2) for Ta).

To understand the notably large ppl improvements better, we compute ppl differences compared to LMMRL by considering subsets of test tokens. First, we consider test tokens that were unsegmented ($\alpha_t = 0$) followed by tokens that were segmented ($\alpha_t = 1$). We see enormous gains in ppl on the segmented tokens compared to the unsegmented tokens. Next, we consider `<unk>` test tokens that can be recovered using the prefix/suffix rules returned by our segmentor. There are 348 and 1202 such tokens in Hi and Ta, respectively. We see large ppl improvements on these tokens, except from the SEG(2) + BCE + EMBED model for Hi. This further validates our factored model in being able to recover words that do not appear in the vocabulary.

[3]We try to match the performance of NOSEG by increasing the size of word embeddings in the LMMRL baseline. However, this leads to overfitting.

Model	Language	
	Hi	Ta
SEG(2) [$\alpha = 0$]	68.07	427.83
SEG(2) + BCE [$\alpha = 0$]	69.09	499.66
SEG(2) + BCE + EMBED [$\alpha = 0$]	50.48	352.99
SEG(2) [$\alpha = 1$]	2370.83	10141.61
SEG(2) + BCE [$\alpha = 1$]	2355.10	11174.73
SEG(2) + BCE + EMBED [$\alpha = 1$]	4344.86	15397.68
SEG(2) [`<unk>`]	4906.35	16603.92
SEG(2) + BCE [`<unk>`]	4930.95	18618.97
SEG(2) + BCE + EMBED [`<unk>`]	-18805.0	15590.24

Table 3: Differences in test ppl compared to LMMRL.

4 Related Work

Prior work has investigated various ways in which morpheme or character-level information can be provided as an input to RNNLMs (dos Santos and Zadrozny, 2014; Kim et al., 2016; Ling et al., 2015; Lample et al., 2016). Approaches tailored specifically for morphologically rich languages include the use of constituent morpheme embeddings (Botha and Blunsom, 2014), using morphological recursive neural networks (Luong et al., 2013), concatenating word and character embeddings (Verwimp et al., 2017) and using other factored representations of words (Vylomova et al., 2017; Ataman and Federico, 2018; Labeau and Allauzen, 2017).

Fewer approaches have focused on injecting subword level information into the output layer of LMs. Gerz et al. (2018) proposed a finetuning technique for word embeddings using a loss based on character-level similarities. Yuret and Biici (2009) and Varjokallio and Klakow (2016) split words into subwords and trained an LM using subwords as tokens. Matthews et al. (2018) is most closely related to our work. They used a mixture model to predict at the word, morpheme and character-level; however, they did not use supervision for the mixture weights and they used language-specific morphological analyzers.

5 Conclusions

We present SEG-LM that predicts a mixture of two output layers producing segmented and unsegmented words. Words are segmented into subword units using an unsupervised algorithm. Our models achieve substantial reductions in perplexity on two morphologically complex languages, Hindi and Tamil. In future work, we will examine the effect of using different segmentation algorithms on LM performance.

References

Duygu Ataman and Marcello Federico. 2018. Compositional representation of morphologically-rich input for neural machine translation. In *ACL*.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *ICML*.

Ryan Cotterell, Arun Kumar, and Hinrich Schütze. 2016. Morphological segmentation inside-out. In *EMNLP*.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. *CoRR*, cs.CL/0205057.

Daniela Gerz, Ivan Vulic, Edoardo Maria Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018. Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *Transactions of the Association for Computational Linguistics*, 6:451–465.

John A. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27:153–198.

Cong Duy Vu Hoang, Trevor Cohn, and Gholamreza Haffari. 2016. Incorporating side information into recurrent neural network language models. In *HLT-NAACL*.

Yoon Kim, Yacine Jernite, David A Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *AAAI*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Matthieu Labeau and Alexandre Allauzen. 2017. Character and subword-based word representation for neural language modeling prediction. In *SWCN@EMNLP*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *HLT-NAACL*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015. Character-based neural machine translation. *CoRR*, abs/1511.04586.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*.

Austin Matthews, Graham Neubig, and Chris Dyer. 2018. Using morphological knowledge in open-vocabulary neural language models. In *NAACL-HLT*.

Graham Neubig and Chris Dyer. 2016. Generalizing and hybridizing count-based and neural language models. In *EMNLP*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Emily Pitler and Samarth Keshava. 2007. A segmentation approach to morpheme analysis.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *ACL*.

Matti Varjokallio and Dietrich Klakow. 2016. Unsupervised morph segmentation and statistical language models for vocabulary expansion. In *ACL*.

Lyan Verwimp, Joris Pelemans, Hugo Van hamme, and Patrick Wambacq. 2017. Character-word lstm language models. In *EACL*.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2017. Word representation models for morphologically rich languages in neural machine translation. In *SWCN@EMNLP*.

Zhilin Yang, Zihang Dai, Ruslan R. Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank rnn language model. *CoRR*, abs/1711.03953.

Deniz Yuret and Ergun Biici. 2009. Modeling morphologically rich languages using split words and unstructured dependencies. In *ACL/IJCNLP*.