Improving Character-Aware Neural Language Model by Warming Up Character Encoder under Skip-gram Architecture

Yukun Feng[1], **Chenlong Hu**[1], **Hidetaka Kamigaito**[1], **Hiroya Takamura**[2] and **Manabu Okumura**[1]
[1]Tokyo Institute of Technology
[2]National Institute of Advanced Industrial Science and Technology (AIST)
{yukun,huchenlong,kamigaito}@lr.pi.titech.ac.jp
oku@pi.titech.ac.jp takamura.hiroya@aist.go.jp

Abstract

Character-aware neural language models can capture the relationship between words by exploiting character-level information and are particularly effective for languages with rich morphology. However, these models are usually biased towards information from surface forms. To alleviate this problem, we propose a simple and effective method to improve a character-aware neural language model by forcing a character encoder to produce word-based embeddings under Skip-gram architecture in a warm-up step without extra training data. We empirically show that the resulting character-aware neural language model achieves obvious improvements of perplexity scores on typologically diverse languages, that contain many low-frequency or unseen words.

1 Introduction

Neural language models (NLM) usually maintain a fixed vocabulary and map each word to a continuous representation. These models cannot handle new words and are not effective for languages with rich morphology. One solution is to use smaller units, such as bytes, characters, or word pieces learned from word tokens (Wu et al., 2016; Sennrich et al., 2016). However, this approach has to process longer sequences than word-level alternatives and may increase modeling and computational challenges (Cherry et al., 2018). This paper focuses on another way based on word-level models in which a character encoder is used on top of characters of each word to calculate the word representation. They are often referred to as character-aware NLMs (CNLMs) (Ling et al., 2015; Kim et al., 2016; Vania and Lopez, 2017; Gerz et al., 2018; Assylbekov and Takhanov, 2018; Feng et al., 2019). However, the character encoders in CNLMs often show over-representation of orthography rather than semantic meaning in the resulting word embedding despite the fact that training word-based NLMs usually helps learn such semantic meaning (Kim et al., 2016; Vania and Lopez, 2017; Assylbekov and Takhanov, 2018). For example, in CNLMs, the nearest neighbors of the word 'his' with cosine similarity are 'hhs' and 'this' while 'my' is far from the nearest neighbors.

To alleviate the over-representation issue in CNLMs, we propose to directly force the character encoder to produce the word-based embedding in a warm-up step before the training of CNLMs starts. Specifically, the character encoder encodes an input word, and the encoded embedding will be forced to be close to the embeddings of its surrounding words and far from the word embeddings of negative samples. Unlike the dynamically constructed embedding of the input word, the embeddings of surrounding words and negative samples are word-based, and thus these embeddings will not be biased to surface forms. The above method is similar to the architecture of the Skip-gram model (Mikolov et al., 2013) with the difference that we use a complex character encoder which is shown to be powerful for languages with rich morphology.

In our experiments[1], we choose the widely used long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and the recent state-of-the-art AWD-LSTM-LM (Merity et al., 2018) for language modeling. For the character encoder, we experiment with bidirectional LSTM (BiLSTM) over character trigrams as this variant has shown better performance than other character encoders on 10 languages (Vania and Lopez, 2017). We evaluate our method in two types of datasets. One contains 14 typologically diverse languages with a large number of low-frequency words and unseen words in the test set. Thus, we can test our method in a real LM setup. Another one contains 5 lan-

[1]Our code can be obtained from https://github.com/yukunfeng/warmup_char_lm

Proceedings of Recent Advances in Natural Language Processing, pages 421–427
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_048

guages, where the new words in the test set have been replaced to <UNK>. It is commonly used for evaluating CNLMs in this field.

Our experiments empirically show our method can achieve obviously improved perplexity scores on a wide range of languages. Finally, we analyze the learned word embedding with our character encoder on English word similarity tasks.

2 Related Work

A lot of work has tried to improve CNLMs in recent years, such as analyzing performance of CNLMs with different character encoders and character units (Vania and Lopez, 2017), reusing subword embeddings in CNLMs (Assylbekov and Takhanov, 2018), injecting subword-level information into softmax (Gerz et al., 2018), and combining word- and character-level information in CNLMs (Miyamoto and Cho, 2016; Verwimp et al., 2017; Kang et al., 2011; Feng et al., 2019).

As for alleviating the over-representation issue mentioned above, Kim et al. (2016) used a highway network on top of their character encoder and found their highway network can encode semantic features that are not discernible from orthography alone. Assylbekov and Takhanov (2018) used syllables and morphemes in a word to construct word embeddings and showed syllable- or morpheme-based CNLMs are less biased towards surface forms than a standard CNLM. However, this approach relies on extra toolkits to extract syllables or morphemes. To our knowledge, there is not much work particularly paying attention to this issue in CNLMs. It is discussed only in a section in the above mentioned work, and the experiment is limited to several languages. Furthermore, the analysis of character encoders is done by manually selecting several words with their nearest neighbors based on cosine similarity, while we formally verify that the character encoder captures more semantic features on 5 English word similarity tasks. Our method is simple and different from the highway network used by Kim et al. (2016). We do not need to change the existing architecture of CNLMs.

Another related work to ours is word representation learning as we utilize the Skip-gram architecture. One goal of this field is to learn word embeddings on large-scale corpus and use them on downstream tasks, which is different from ours. Our method works without extra training data, and we do not aim at transfer learning with other training data like a standard Skip-gram model.

3 Model Description

The whole architecture is shown in Figure 1. We use BiLSTM over character trigrams as our character encoder since this variant performed best on most datasets (Vania and Lopez, 2017). Given a word w_t, we denote its embedding as $\mathbf{x}_t \in \mathbb{R}^d$, where d is the embedding size. We compute the representation of w_t in BiLSTM as follows:

$$\mathbf{x}_t = \mathbf{W}_f \mathbf{h}^{fw} + \mathbf{W}_b \mathbf{h}^{bw} + \mathbf{b}, \quad (1)$$

where $\mathbf{h}^{fw}$, $\mathbf{h}^{bw} \in \mathbb{R}^d$ are the last states of the forward and backward LSTMs, respectively. $\mathbf{W}_f$, $\mathbf{W}_b \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ are trainable parameters.

We adopt the basic architecture of Skip-gram model for warming up our character encoder. Given an input word w_t which will be encoded by our character encoder, we then use the encoded embedding to predict a set of output words that surround the input word in a given window. For example, when the window size is 2, the output words are $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$. We use $\mathbf{o}_t$ to denote the embedding of an output word for w_t. The input word embedding $\mathbf{x}_t$ of w_t is computed with Eq. 1. Note that $\mathbf{o}_t$ is word-based and thus will be not biased to its surface form, which is different from $\mathbf{x}_t$. Given a single training example (w_t, w_{t+j}), we maximize the objective function:

$$\log\sigma(\mathbf{x}_t^T \mathbf{o}_{t+j}) + \sum_{i=1}^{k} \log\sigma(-\mathbf{x}_t^T \mathbf{o}_{t+i}), \quad (2)$$

where k is the size of the negative samples, and σ is the sigmoid function.

After warming up, we use the trained character encoder to initialize the one in our CNLM and then train our CNLM with a standard LM loss.

4 Experiments

4.1 Datasets

We can find common language modeling datasets for evaluating CNLMs in the work of Botha and Blunsom (2014). While these datasets contain languages with rich morphology, they have only 5 different languages. The most large-scale language modeling datasets are from the work of Gerz et al. (2018), who released 50 language modeling datasets covering typologically diverse languages. The difference of the newly released datasets from

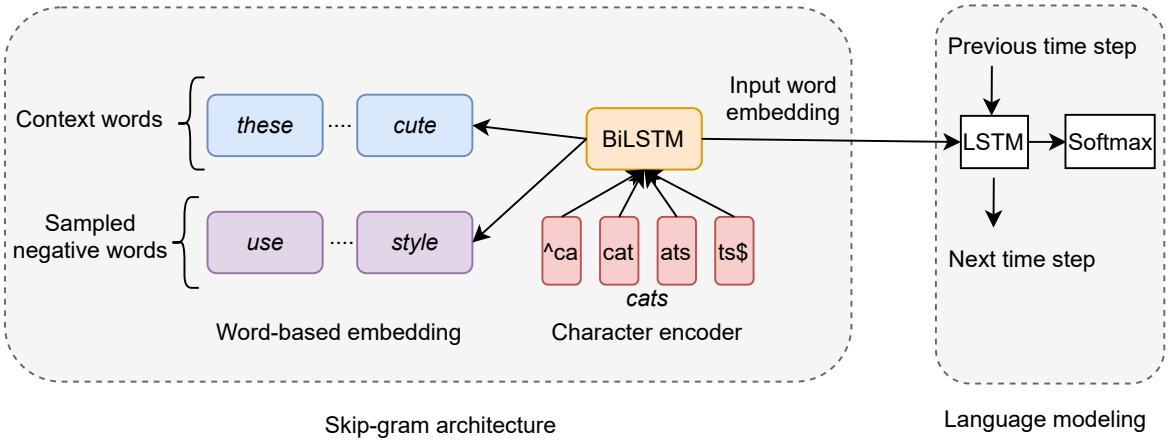

Figure 1: Our character-aware language model with Skip-gram architecture for warming up to avoid being too biased to surface forms for the character encoder. The example sentence is "these cats are cute" with "cats" as the current input word.

the previously common datasets is that many unseen words are kept in the test set. Thus, on the datasets, we can test our methods in a real LM setup. These languages were selected to represent a wide spectrum of different morphological systems and have a large number of low-frequency or unseen words. Thus, these datasets are desirable for checking the performance of CNLMs. Due to the large number of experiments, we chose datasets of only 14 languages from these datasets and tried to cover different language typologies as well as different type/token ratios (TTRs). The statistics of our chosen datasets are shown in Table 1.

To compare with other models, we also set up above mentioned 5 common non-English LM datasets with rich morphology from the 2013 ACL Workshop on Machine Translation, which have been commonly used for evaluating CNLMs (Botha and Blunsom, 2014; Kim et al., 2016; Bojanowski et al., 2017; Assylbekov and Takhanov, 2018; Feng et al., 2019). Note that the new words in the test set have been replaced with special <UNK>, which is not a practical setting. The data statistics is in Table 2.

4.2 Models

The hyperparameters of our LSTM language model are shown in Table 3. The learning rate was decreased if no improvement is observed in the validation set. We trained the Skip-gram architecture in warm-up step for 7 epochs with 5 negative samples for all datasets. We define **Char-BiLSTM-LSTM** as our CNLM, and **Warmed-Char-BiLSTM-LSTM** as our CNLM with a warmed character encoder.

To check our idea with a stronger baseline, we used the recent state-of-the-art AWD-LSTM-LM codebase[2](Merity et al., 2018). We replaced the word embedding layer of this model with our BiLSTM character encoder. We refer to it as Char-BiLSTM-AWD-LSTM and the warmed one as Warmed-Char-BiLSTM-AWD-LSTM. Due to time constraints, we set the training epoch on all datasets to 200. We refer to the original AWD-LSTM which is word-level as Word-AWD-LSTM. For the other parameters, we followed the setting in the source code. We make sure that all models under our chosen epochs are trained to convergence so that the gain from our method is not due to longer training in warm-up.

4.3 Results on 14 Languages

The results on 14 languages are shown in Table 4. Our Char-BiLSTM-LSTM baseline outperforms Char-CNN-LSTM from (Gerz et al., 2018) on all datasets. It is also shown that as the TTR increases, Char-BiLSTM-AWD-LSTM achieves a better result than Word-AWD-LSTM. One reason may be that higher TTR languages have more low-frequency words and unseen tokens, as shown in Table 1. Thus, utilizing character information is important in these languages. Our proposed Warmed-Char-BiLSTM-LSTM and Warmed-Char-BiLSTM-AWD-LSTM achieves further obvious improvements compared with Char-BiLSTM-LSTM and Char-BiLSTM-AWD-LSTM respectively on most datasets without extra training

[2]https://github.com/salesforce/awd-lstm-lm

Dataset	Typology	TTR	Train Vocab	#Train tokens	#Test tokens	Freq<=15 (Train)	#Unseen tokens
zh (Chinese)	Isolating	0.06	43672	746K	56.8K	16%	2132
ja (Japanese)	Agglutinative	0.06	44863	729K	54.6K	15%	2558
pt (Portuguese)	Fusional	0.07	56167	780K	59.3K	17%	2947
en (English)	Fusional	0.07	55521	783K	59.5K	17%	3618
es (Spanish)	Fusional	0.08	60196	781K	57.2K	18%	3486
he (Hebrew)	Introflexive	0.12	83217	717K	54.6K	27%	4855
de (German)	Fusional	0.12	80741	682K	51.3K	24%	5451
ar (Arabic)	Introflexive	0.12	89089	722K	54.7K	26%	6076
cs (Czech)	Fusional	0.14	86783	641K	49.6K	30%	5436
ru (Russian)	Fusional	0.15	98097	666K	48.4K	32%	4881
et (Estonian)	Agglutinative	0.17	94184	556K	38.6K	34%	4960
fi (Finnish)	Agglutinative	0.2	115579	585K	44.8K	38%	7899
ko (Korean)	Agglutinative	0.22	143794	648K	50.6K	42%	9745
kn (Kannada)	Agglutinative	0.22	94660	434K	29.4K	41%	5214

Table 1: The statistics of our language modeling datasets. TTR represents the type/token ratio.

	Vocab size	#Train token
Czech (CS)	46K	1M
German (DE)	37K	1M
Spanish (ES)	27K	1M
French (FR)	25K	1M
Russian (RU)	86K	1M

Table 2: The statistics of our 5 language modeling datasets.

Embedding size d	650
LSTM layers	2
LSTM sequence length	35
Param. init: rand uniform	
Dropout	0.5
Epochs	40
Optimizer	SGD
Learning rate	20
Learning rate decay	4
Gradient clipping	0.25
Batch size	20

Table 3: Hyperparameters of our model. We use d for the size of the character/word embeddings and for the number of hidden units of LSTM and Bi-LSTM.

data. This indicates that our method is effective on typologically diverse languages and for different CNLMs. In addition to obtaining large improvements, our method does not change the speed of CNLMs as it adds only one extra warm-up phase.

4.4 Results on 5 Common Datasets

The results on common datasets are shown in Table 5. Most work aims at improving CNLMs at different aspects and the gain comes from different new information. For example, the gain of CNLM from Feng et al. (2019) comes from injecting word-level information into CNLM, and Assylbekov and Takhanov (2018) improved CNLMs by using morphemes and reusing weights. Bojanowski et al. (2017) used conventional word-level LSTM-LM instead of CNLM, and their goal is not to improve CNLMs. The gain from their model comes from transferring word embeddings learned through Skip-gram that considers character-level information to word-level LSTM-LM without character-level information. That is, their method used new information for their LSTM-LM while in our method there is no extra new information. As shown in Table 5, our baseline model is strong compared with most models, and our method can further improve it without extra new information.

5 Analysis

5.1 Analysis of Character Encoder

Unlike prior work which analyzes their character encoder by manually selecting several words and their nearest neighbors based on cosine similarity, we formally verify whether our method helps the character encoder capture more semantic features on English word similarity tasks. We chose the English dataset 'en' shown in Table 1 as our training set. Specifically, after finishing the training of

Dataset TTR	zh 0.06	ja 0.06	pt 0.07	en 0.07	es 0.08	he 0.12	de 0.12	ar 0.12	cs 0.14	ru 0.15	et 0.17	fi 0.2	ko 0.22	kn 0.22
Char-CNN-LSTM (Gerz et al., 2018)	797	136	214	371	275	1519	602	1659	1252	812	1478	2236	4778	2558
Char-BiLSTM-LSTM	578	107	178	302	230	1170	483	1337	973	620	967	1648	3247	1543
Warmed-Char-BiLSTM-LSTM	480	99	162	278	208	1005	439	1158	843	503	877	1435	2472	1314
Word-AWD-LSTM	481	98	165	289	234	1351	575	1424	1140	760	1359	2116	3909	2308
Char-BiLSTM-AWD-LSTM	497	99	156	263	205	1042	464	1062	743	499	805	**1262**	2472	1724
Warmed-Char-BiLSTM-AWD-LSTM	**414**	**87**	**136**	**236**	**183**	**878**	**408**	**971**	**718**	**485**	**768**	1278	**2082**	**1271**

Table 4: Perplexity results for our models and several baselines.

	CS	DE	ES	FR	RU
MLBL (Botha and Blunsom, 2014)	465	296	200	225	304
MorphSum (Kim et al., 2016)	398	263	177	196	271
CharCNN (Kim et al., 2016)	371	239	165	184	261
SkipGram initialization (Bojanowski et al., 2017)	312	206	145	159	206
MorphSum+RE+RW (Assylbekov and Takhanov, 2018)	338	222	157	172	210
Word-Char-LSTM (Feng et al., 2019)	**287**	192	135	152	**201**
Char-BiLSTM-LSTM	311	198	144	164	223
Warmed-Char-BiLSTM-LSTM	290	**190**	**134**	**150**	203

Table 5: Perplexity of our models and previous work.

our CNLMs on the 'en' dataset, we fed all the test words in the word similarity tasks into the character encoder with and without warm-up, to obtain word representations for these test words. Then, we evaluated the quality of these representations by computing the Spearman's rank correlation coefficient (Spearman, 1904). We chose MEN (Bruni et al., 2012), MTurk287 (Radinsky et al., 2011), RW (Luong et al., 2013), MTurk771 (Halawi et al., 2012) and WS353 (Finkelstein et al., 2002) as our datasets. The results are shown in Table 6. The warm up helps the character encoder better capture semantic relationships between word pairs. Note that the results on these word similarity tasks in our paper are not comparable to the ones for recent models that are designed to directly learn word representations instead of being trained on a language modeling task and that are usually trained with a large corpus. Our models are for the language modeling task and are trained on 'en', which is a small dataset.

	#Word pairs	Char-BiLSTM-LSTM	Warmed-Char-BiLSTM-LSTM
MEN	3000	10.55	**12.52**
MTurk287	287	24.47	**26.84**
MTurk771	771	3.06	**8.59**
RW	2034	17.30	**18.85**
WS353	353	15.17	**18.54**

Table 6: Results on word similarity datasets.

5.2 Analysis of Targeted Perplexity

We measured the perplexity for frequent and rare words in the test data separately to show that our method is beneficial for frequent and rare words. For example, we calculated the perplexity of the next word, when a rare word, whose frequency is less than 15, is given as the current word. A similar analysis on language models can be found in Vania and Lopez (2017). For simplicity, we only choose the English dataset 'en' and the German dataset 'de'. To fairly compare with Word-LSTM, our analysis does not contain new words in the test data. As we see in Table 7, Char-BiLSTM-LSTM mainly obtained improvements on rare word group compared with Word-LSTM. When warmed up, Char-BiLSTM-LSTM obtained further improvements both on frequent and rare word groups. Note that the reason of the gain is not that the warm

up increases the training epochs of CNLMs as our CNLMs have already been trained to convergence on all datasets.

		Freq.	Rare	All
Word-LSTM	en	299	417	314
Char-BiLSTM-LSTM	en	292	245	285
Warmed-Char-BiLSTM-LSTM	en	267	225	261
Word-LSTM	de	541	617	554
Char-BiLSTM-LSTM	de	524	353	487
Warmed-Char-BiLSTM-LSTM	de	491	333	457

Table 7: Targeted perplexity results of our CNLMs.

6 Conclusion

In this paper, we proposed to warm up a character encoder of a character-aware neural language model under the Skip-gram architecture to capture better semantic relationships between word pairs. Our method is simple and effective. It was tested on a standard character-aware neural language model and a recent state-of-the-art model. The results showed that our method is effective on typologically diverse language datasets. For future work, we plan to extend our method to Transformer-based language models and investigate how our model works for other tasks such as text generation.

Acknowledgments

We would like to thank anonymous reviewers for their constructive comments and Hu also thanks the support from China Scholarship Council.

References

Zhenisbek Assylbekov and Rustem Takhanov. 2018. Reusing weights in subword-aware neural language models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1413–1423, New Orleans, Louisiana. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning*, pages 1899–1907.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, Brussels, Belgium. Association for Computational Linguistics.

Yukun Feng, Hidetaka Kamigaito, Hiroya Takamura, and Manabu Okumura. 2019. A simple and effective method for injecting word-level information into character-aware neural language models. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 920–928, Hong Kong, China. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131.

Daniela Gerz, Ivan Vulić, Edoardo 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 of Computational Linguistics*, 6:451–465.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM.

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

Moonyoung Kang, Tim Ng, and Long Nguyen. 2011. Mandarin word-character hybrid-input neural network language model. In *Twelfth Annual Conference of the International Speech Communication Association*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

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*, pages 1520–1530. Association for Computational Linguistics.

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*, pages 104–113.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1992–1997. Association for Computational Linguistics.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Charles Spearman. 1904. The proof and measurement of association between two things. *American journal of Psychology*, 15(1):72–101.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2016–2027, Vancouver, Canada. Association for Computational Linguistics.

Lyan Verwimp, Joris Pelemans, Hugo Van hamme, and Patrick Wambacq. 2017. Character-word lstm language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 417–427. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.