Sub-Word Similarity based Search for Embeddings: Inducing Rare-Word Embeddings for Word Similarity Tasks and Language Modelling

Mittul Singh[1,2,3] Clayton Greenberg[1,2,3] Youssef Oualil[1,3] Dietrich Klakow[1,2,3] *
[1]Spoken Language Systems (LSV)
[2]Saarbrücken Graduate School of Computer Science, Saarland Informatics Campus
[3]Collaborative Research Center on Information Density and Linguistic Encoding
Saarland University, Saarbrücken, Germany
{firstname.lastname}@lsv.uni-saarland.de

Abstract

Training good word embeddings requires large amounts of data. Out-of-vocabulary words will still be encountered at test-time, leaving these words without embeddings. To overcome this lack of embeddings for rare words, existing methods leverage morphological features to generate embeddings. While the existing methods use computationally-intensive rule-based (Soricut and Och, 2015) or tool-based (Botha and Blunsom, 2014) morphological analysis to generate embeddings, our system applies a computationally-simpler sub-word search on words that have existing embeddings. Embeddings of the sub-word search results are then combined using string similarity functions to generate rare word embeddings. We augmented pre-trained word embeddings with these novel embeddings and evaluated on a rare word similarity task, obtaining up to 3 times improvement in correlation over the original set of embeddings. Applying our technique to embeddings trained on larger datasets led to on-par performance with the existing state-of-the-art for this task. Additionally, while analysing augmented embeddings in a log-bilinear language model, we observed up to 50% reduction in rare word perplexity in comparison to other more complex language models.

1 Introduction

Word embeddings have been successfully applied to many NLP tasks (Collobert and Weston, 2008; Collobert, 2011; Socher et al., 2011; Socher et al., 2012; Hermann and Blunsom, 2014; Bengio and Heigold, 2014; Yang et al., 2015), and these systems often achieved state-of-the-art performance. This success has been ascribed to embeddings' ability to capture regularities traditionally represented in core NLP features. Most of these embeddings were trained on large amounts of data, allowing them to have good coverage of the relevant vocabularies. However, embeddings often still cannot satisfactorily represent *rare words*, i.e. words with few occurrences in training data.

To generate useful embeddings for words too rare for standard methods to handle, Luong et al. (2013) and Botha and Blunsom (2014) leveraged the segmentation tool, Morfessor (Creutz and Lagus, 2005), while Cotterell et al. (2016) used morphological lexica to generate rare-word embeddings. In general, these methods added resource-based knowledge to their systems in order to form word vector representations, showing impressive performance gains over methods which did not address the rare words problem.

In contrast, Soricut and Och (2015) applied an automatic method to induce morphological rules and transformations as vectors in the same embedding space. More specifically, they exploited automatically-learned prefix- and suffix-based rules using the frequency of such transformations in the data and induced a morphological relationship-based word graph. Then, they searched over this graph for rules that best infer the morphology of the rare words. The embeddings were then estimated using these rare-word explaining rules. In this method, creating and tuning this morphological graph could lead to a high initial cost.

* This work was supported by the Cluster of Excellence for Multimodal Computing and Interaction, the German Research Foundation (DFG) as part of SFB 1102, the EU FP7 Metalogue project (grant agreement number: 611073) and the EU Malorca project (grant agreement number: 698824).

Language	V	RW	#ENF	Coverage
German	36602	15715	13103	99.9
Tagalog	22492	10568	8407	98.1
Turkish	24840	13624	9555	99.0
Vietnamese	6423	1332	305	69.1

Table 1: This table reports various statistics for different language datasets used for language modelling. The last column shows the coverage of our method in percentage.

Task	V	#ENF	Coverage
Rare Word (Luong et al., 2013)	2951	1073	100
Gur65 (Gurevych, 2005)	49	4	100
Rare Word + Google News	2951	173	100

Table 2: This table reports various statistics of a few language word similarity datasets used in our experiments. The last column shows the coverage of our method in percentage.

In order to overcome this cost and still be able to automatically induce rare word representations, we propose a sub-word similarity-based search. This technique maps a rare word to a set of its morphologically-similar words and combines the embeddings of these similar words to generate the rare word's representation (further discussed in Section 2). These generated embeddings can then be combined with existing word embeddings to be applied in various tasks.

In Section 3, we evaluate our embeddings on word similarity tasks. For further evaluation, in Section 4, we instantiate a log-bilinear language model (Mnih and Hinton, 2007) with our word embeddings and analyse their perplexity performance on rare words over various language modelling corpora. Finally, we summarise our findings in Section 5.

2 Rare-Word Embeddings

Rare words form a large part of a language's vocabulary. This is illustrated in Table 1, which reports the vocabulary size and number of rare words (RW) with zero (out-of-vocabulary words) or one training set occurrence for our corpora. As shown in this table, rare words constitute 10%-50% of the vocabulary. Further, it is widely known that in English, roughly half of all tokens in a given corpus occur only once. Thus, it is essential to handle rare words properly to obtain good performance.

In the context of word embeddings-related tasks, training good word embeddings can incur huge computational costs (Al-Rfou et al., 2013). So, in this work, we focus on augmenting readily available embeddings rather than creating new ones from scratch. To increase the availability of resources for many languages, Al-Rfou et al. (2013) released[1] pre-trained word embeddings for more than one hundred languages. These pre-trained word embeddings, namely *Polyglot*, were constructed by applying the method outlined in Bengio et al. (2009) on Wikipedia text, which vary in size from millions of tokens to a few billion tokens.

Among other available pre-trained word embeddings, Google released `word2vec` (Mikolov et al., 2013)-based embeddings[2] trained on their English News dataset (about 100 billion tokens). In our experiments, we applied both of these embeddings sets to jump start generating the rare word embeddings for different languages.

2.1 Inducing Rare-Word Embeddings

Statistics about the various language modelling corpora and word similarity tasks that we used in our experiments are shown in Table 1 and Table 2. In these tables, along with the vocabulary size and number of rare words, we also report the number of words for which the embeddings were not found (ENF = Embedding Not Found) in the pre-trained embedding sets. For most of the language and pre-trained embedding pairs, number of ENFs formed a large share of the vocabulary for word similarity tasks and of rare-word set size for language modelling tasks. Hence, we estimated the missing word embeddings before using them in our tasks.

We first provide a high level description of the steps of our method to induce the word embeddings for these missing rare words, followed by detailed description of each step. For a given set of pre-trained embeddings with a finite vocabulary V_E applied to a task with vocabulary V_T and a finite set of given rare words $RW = \{w | w \notin V_E \& \, w \in V_T\}$, we apply the following steps:

[1]`https://sites.google.com/site/rmyeid/projects/polyglot`
[2]`https://code.google.com/archive/p/word2vec/`

1. Map every word $w \in V_E$ to its sub-word features
2. Index $w \in V_T$ using its sub-word features
3. Search the index for matches of $w' \in RW$
4. For every $w' \in RW$, combine matched words' embeddings to generate its embedding

Step 1: Map words to sub-words
Although a word may be rare, substrings of that word are, in general, less rare. Hence, we start by breaking down each word $w \in V$ into its constituent N-sized sub-word units: $D_N(w)$. For example, given the sub-word size $N = 3$:

$$D_N(language) = \{lan, ang, ngu, gua, uag, age\}$$

In our experiments, we worked with value of $N = 3$. However, it remains to be seen how using differently sized sub-word units or even morphemes affects the performance of this method. Note that our procedure does not formally require that sub-word units be of equal length, so linguistically-sensible morphemes may be used if the resource is available for that language.

Step 2: Index word using its sub-words
Pre-trained sets of embeddings can cover large numbers of words already (for example, *Polyglot* embeddings have 100K words in their vocabulary). So, performing substring searches and comparisons can become quite computationally expensive. To speed up the search for sub-word units, we create an inverted index on words. For each $w \in V$, we treat $D_N(w)$ as a document and feed it into a search engine-based indexer. In this work, we used Lucene[3] (McCandless et al., 2010) to index the words.

Step 3: Search for matches of a rare word
Next, we break down the rare word $w' \notin V$ into its sub-word units ($D_N(w')$) and search for $D_N(w')$ using the index. We restrict the search results set to the top K results, denoted by $R^K(w')$. $R^K(w')$ contains the words having similar sub-word units as w', hence, containing words which are sub-word similar to w'. In our experiments, we fixed $K = 10$.

Step 4: Generating rare-word embeddings
To estimate the word embedding of $w' \in RW$, we compute the weighted average of embeddings (v) of the rare-word matches. For this weighted average, we employ a string similarity function S, such that

$$v_{w'} = \sum_{w : D_N(w) \in R^K(w')} S(w', w) \times v_w$$

The above method particularly hinges on the third step, where we utilise sub-word similarity of morphologically similar words to search for rare word alternatives, leading to embedding combination in the fourth step. Hence, we refer to the above technique as **Sub-Word S**imilarity based **S**earch (**SWordSS**: pronounced swordz). The SWordSS embeddings ($\{v_{w'} : w' \in RW\}$) are used along with $\{v_w : w \in V\}$ to perform rare word-related tasks.

In the fourth step, we apply different string similarity functions (S), described in the list below, to average different embeddings of matches from the third step. These different similarity functions help provide a more morphologically-sensible scoring of matches and eventually are used to weight the inputs of the final rare word embeddings.

- Jaccard Index, Jaccard (1912) computes the size of the character intersection over the size of the character union. Therefore, order of characters is not considered by this metric. Frequent characters such as vowels lead to uninteresting intersections, and short words could possibly suffer from an unfair floor.

[3]`https://lucene.apache.org/`

- Jaro similarity, Jaro (1989) considers the number of matching characters in corresponding positions and the number of transpositions detected. So, order of characters does matter for this metric. Insertions and deletions are treated similarly, and the frequency and length effects from Jaccard could also affect this metric.
- Most frequent K Characters similarity, Seker et al. (2014) considers the counts of the top K characters in each string. Thus, if the "root morphemes" are long enough to create nontrivial count statistics, this metric may, too, favor a more linguistic similarity, but as before, shorter strings could have unwanted effects.
- Subsequence Kernels, Lodhi et al. (2002) create automatically-generated features based on sequences of characters within the strings to be compared. Therefore, those sequences that do not cross morpheme boundaries could be especially helpful for estimating morphological similarity.
- Tversky coefficient, Tversky (1977) breaks down the union in the Jaccard index, allowing different weights for the denominator intersection, those characters that only appear in the first string, and those characters that only appear in the second string. These metaparameters allow the metric some flexibility that the others do not.

In our experiments on rare word-related tasks, we mostly observed that using SWordSS led to high coverage rates, also presented in Table 1 and Table 2. We note that whenever words w' resulted in zero matches in our experiments, they were either removed completely (in case of word similarity tasks) or substituted with random vectors (in case of language modelling tasks, Section 4).

3 Word Similarity Task

To test the efficacy of SWordSS embeddings, we evaluated them on two standard word similarity tasks. In such tasks, the correlation between the human annotator ratings of word pairs and the scores generated using embeddings was calculated. A good set of embeddings would achieve a high correlation.

Specifically, we evaluated the SWordSS embeddings on Luong et al. (2013)'s English Rare Words dataset with 2034 word pairs (Luong2034) and also evaluated these embeddings on a German word similarity task (Gurevych, 2005) with 65 word pairs (Gur65).

3.1 Experimental Setup

For the German word similarity task, we used only *Polyglot* word embeddings, which are 64-dimensional vectors. For English along with *Polyglot* word embeddings, we used the Google News word2vec embeddings, which are 300-dimensional vectors.

As a baseline, we used the existing pre-trained word embeddings, which are compared to their augmented SWordSS versions. While augmenting the pre-trained set with the SWordSS embeddings, we also explored various string similarity functions to be used in the fourth step (Section 2.1), namely, Jaccard Index (SWordSS$_{ji}$), Jaro similarity (SWordSS$_{jaro}$), Most Frequent K Characters similarity (SWordSS$_{mfk}$), Subsequence Kernels (SWordSS$_{ssk}$) and Tversky Coefficient (SWordSS$_{tc}$).

To evaluate the effect of these string similarity functions, we also implemented a constant similarity function ($S(w, w') = 1$, where w and w' are words) used in the fourth step, denoting the corresponding embeddings by SWordSS$_1$. Finally, we also compared the SWordSS embeddings to SO2015 (Soricut and Och, 2015), which also applies morphological analysis to generate missing word embeddings quite similar to SWordSS embeddings.

3.2 Results

Using SWordSS embeddings definitely increased the correlation with humans in comparison to the original on the Gur65 task (shown in Table 3), though the different string similarity functions except the constant function (SWordSS$_1$) led to correlations in a very close range, showing that particularly for German, different similarity functions behave very similarly. Henceforth, we only report the best correlation coefficient after applying these functions.

Word Vectors	Gur65
Polyglot	28.5
Polyglot+SWordSS$_{ji}$	37.5
Polyglot+SWordSS$_{jaro}$	37.1
Polyglot+SWordSS$_{mfk}$	37.2
Polyglot+SWordSS$_{ssk}$	36.9
Polyglot+SWordSS$_{tc}$	**37.6**
Polyglot+SWordSS$_1$	35.8

Table 3: Spearman's rank correlation (%) based evaluation of various string similarity functions used to generate augmented word vectors for the German word similarity task (Gur65)

Task	Luong2034	
Word Vectors	*Polyglot*	*Google News*
SO2015 w/o morph	-	44.7
SO2015 w/ morph	-	**52.0**
w/o SWordSS	9.7	45.3
w/ SWordSS$_1$	28.9	51.3
w/ SWordSS$_{sim}$	30.4	51.4

Table 4: Spearman's rank correlation (%) based evaluation of techniques with and without morphological features used to generate representations for the word similarity task.

Next, we compared SWordSS versions of *Polyglot* embeddings and Google News Embeddings on the Luong2034 task. When the SWordSS versions were compared to the original (labelled w/o SWordSS) it led to a higher correlation, as shown in Table 4. However, for each set of embeddings, the difference between SWordSS$_1$ and SWordSS$_{sim}$ remained small. The correlations for the SWordSS version of *Polyglot* were still lower than the correlation rates reported by SO2015. This was due to the difference in initial quality of embeddings used by each method. As *Polyglot* embeddings trained on a lesser amount of data than SO2015, they were easily outperformed.

In Table 4, we addressed this lower performance issue by replicating our experiment using Google News `word2vec` embeddings to jump start the SWordSS versions for the Luong2034 task. Using these embeddings, trained on a larger dataset than used by *Polyglot*, led to SWordSS versions having on-par results with the SO2015 results for the Luong2034 task.

Overall the SWordSS technique was able to drastically improve pre-trained embeddings performance on the above word similarity tasks. Even though SWordSS-augmented Google News embeddings did not significantly outperform SO2015, this method provides a simpler sub-word search based alternative to the graph search over morphological relationships performed by SO2015. Furthermore, by applying sub-word search in the third step as shown in Section 2.1, SWordSS overcomes the need for creating and tuning the graph of morphological relationships as required by SO2015.

4 Word Embeddings in Language Models

Training language models (LMs) using an expanded vocabulary (having more word types than contained in the training corpus) requires assigning probabilities to words which are not present in the training set. Traditionally, these rare words are assigned a default value of probability in conventional N-gram and long short term memory (LSTM)-based reccurrent neural network LMs (Sundermeyer et al., 2012). This is usually not beneficial for spoken term detection and automatic speech recognition systems made for low resourced languages, since presence of rare words in speech queries is high (Logan et al., 1996; Logan et al., 2005).

To avoid this misrepresentation of rare words, we apply SWordSS embeddings in a language modelling framework. Specifically, a log-bilinear language model (LBL) (Mnih and Hinton, 2007). In our experiments, when the SWordSS embeddings were used to initialise an LSTM's input layer, the system obtained the same perplexity values as the LSTM initialised with random embeddings. This observation suggests that the LBL framework is better suited than LSTMs for this naïve way of initialising neural language models with SWordSS embeddings and improving perplexity on rare words.

LBL predicts the next word vector $p \in \mathbb{R}^d$, given a context of $n-1$ words, as a transformed sum of context word vectors $q_j \in \mathbb{R}^d$, as:

$$p = \sum_{j=1}^{n-1} q_j C j$$

where $C_j \in \mathbb{R}^{d \times d}$ are position-specific transformation matrices. p is compared with the next word w's representation r_w. This comparison is performed using the vector dot product and then is used in a softmax function to obtain the probability of the next word as follows:

$$p(w_i|w_{i-n+1}^{i-1}) = \frac{\exp(p \cdot r_w + b_w)}{\sum_{v \in V} \exp(p \cdot r_v + b_v)}$$

where b is the bias term encoding the prior probability of word type w.

First, Q the collection of context word vectors (q_j) and R the collection next word representations (r_w) are initialised with the pre-trained word embeddings. Thereafter, we train the LBL using stochastic gradient descent.

Previously, extensions to class based and factor based formulations have provided impressive improvements over regular N-gram LMs for morphological languages (Botha and Blunsom, 2014). But, these LMs do not provide straightforward ways of incorporating pre-trained word embeddings, so we use the original LBL because of the ease with which it incorporates pre-trained embeddings in its formulation.

4.1 Data

To evaluate the SWordSS embeddings for language modelling, we used the Europarl-v7 corpus of German (de) language as processed by Botha and Blunsom (2014). We also performed language modelling experiments with the SWordSS embeddings on Tagalog (tl), Turkish (tr) and Vietnamese (vi) corpora, which include transcriptions of phone conversations collected under the IARPA Babel Program language collection releases babel106b-v0.2f, babel105-v0.5 and babel107b-v0.7 respectively.

The German corpus was processed to have no out-of-vocabulary words (OOVs), however, it still had a lot of low frequency words (see Table 2). Contrastingly, the Babel corpora have OOVs as well as other low frequency words.

Statistics	de	tl	tr	vi
Train	1000K	585K	239K	985K
Dev	74K	30K	5K	65K
Test	73K	31K	6K	60K
Voc Size	37K	22K	25K	6K

Table 5: Statistical summary of corpora used for the language modelling experiments. Information corresponding to a language is presented in a column.

The Babel corpora were provided with training and development sets. We divided the existing development set into two halves to use one as the test set and the other half as the new development set. The statistics on these corpora are summarised in Table 5.

In Tables 1 & 2, we had shown that even though a lot of rare-word embeddings are missing from the pre-trained set, SWordSS was able to generate and obtain high coverage rates for such words, giving this method added benefit in the context of rare words.

4.2 Experimental Setup

Before evaluating the SWordSS embeddings for predicting rare words, we used all the OOVs to expand the corresponding vocabulary. SWordSS embeddings for all the words in the expanded vocabulary were used to initialise LBL framework as described in Section 4. A bigram version of this LBL ($\text{LBL2}_{SWordSS}$) was further trained on language corpora before being evaluated.

We compare our $\text{LBL2}_{SWordSS}$ model with the conventional Modified-Kneser-Ney five-gram LM (MKN5) (Kneser and Ney, 1995; Chen and Goodman, 1996) and also with the bigram (LBL2) based log-bilinear LM. As a more powerful baseline, we also trained an LSTM based RNN LM to compare with $\text{LBL2}_{SWordSS}$. Moreover, we compare the $\text{LBL2}_{SWordSS}$, with a character aware language model (Kim et al., 2015), denoted as CCNN-LSTM. The CCNN-LSTMs were chosen for comparison because of their ability to use character-based features to implicitly handle OOVs and rare words. For training each of these LMs, we used the expanded vocabulary as used by $\text{LBL2}_{SWordSS}$. In training neural network-based language models, we restricted the number of parameters to have a similar number of parameters as $\text{LBL2}_{SWordSS}$.

4.3 Perplexity Experiments

We compare the language models described in Section 4.2 using perplexity values calculated on test sets of different languages, shown in the Table 6.

Language Model	German		Tagalog		Turkish		Vietnamese	
	PPL	RW1PPL	PPL	RW1PPL	PPL	RW1PPL	PPL	RW1PPL
MKN5	364.2	559K	162.6	420K	478.9	139K	120.8	174K
LBL2	391.1	404K	171.4	204K	649	**94K**	137.6	**100K**
LSTM[4]	323.1	596K	134.7	343K	489.8	110K	**102.1**	457K
CCNN-LSTM	**315.7**	636K	**117.4**	354K	**408.7**	168K	182.7	516K
LBL2$_{SWordSS}$	369.4	**260K**	167.2	**167K**	513.2	110K	136.4	143K
#PAR	4.7 M		2.9 M		3.2 M		0.8 M	

Table 6: Perplexities on test set (PPL), RW1 perplexities (RW1PPL) in thousands and number of parameters (#PAR) for LBL and LSTM LMs in millions, presented on four language corpora

As shown in Table 6, LBL2$_{SWordSS}$ was able to outperform the conventional LBL2 comfortably on all the corpora except Vietnamese. For Vietnamese, LBL2$_{SWordSS}$ and LBL2 performed comparably. Due to SWordSS' low coverage of Vietnamese vocabulary, initialising LBL2 with SWordSS embedding led to only a marginal performance gain.

Overall in terms of test set perplexity, CCNN-LSTM outperformed LBL2$_{SWordSS}$ comfortably on most language corpora. However, on Vietnamese (in which characters represent meaning units rather than sounds) CCNN-LSTM suffered and the LSTM outperformed the other language models. In comparison to LSTM and CCNN-LSTM, LBL2$_{SWordSS}$'s lower performance on test data was expected as the former are more non-linearly complex language models.

However, for tasks like spoken term detection, having low perplexities on most frequent set of words is not good enough and hence, we compare LMs on the perplexity of a rare-word based test set. To perform this comparison, we computed perplexity only on rare words (RW1PPL), i.e. with training-set frequency of one, present in the test set. As shown in Table 6, we observe that LBL2$_{SWordSS}$ performed better than the LSTM-based LMs across various languages in terms of RW1PPL.

We note that CCNN-LSTM model cannot include SWordSS embeddings easily. Hence, they are not directly comparable to LBL2$_{SWordSS}$, as the latter has more information at its disposal.

4.4 Performance on OOVs and Rare Words

To further compare the performance of the aforementioned language models on rare words, we analyse perplexities of such words (RWPPL) in the test set as a variation of the frequency classes of these words in the training set. This variation is displayed in Figure 1.

For OOVs (rare words with zero training-set frequency), LBL2$_{SWordSS}$ outperformed the other language models built with similar number of parameters, on the Tagalog and Turkish corpora. In these cases, LBL2$_{SWordSS}$ reduced rare-word perplexities by a factor of two over the character-feature rich CCNN-LSTM, whose design allows it to implicitly handle rare words.

Even for rare words with training set frequency up to one, LBL2$_{SWordSS}$ reduced perplexity up to a factor of 2.5 times with respect to CCNN-LSTM, on the German, Tagalog and Turkish corpora. Interestingly on these particular language corpora, Figure 1 shows that LBL also performed better than both the LSTM-based LMs in modelling OOV and rare words of frequency up to ten.

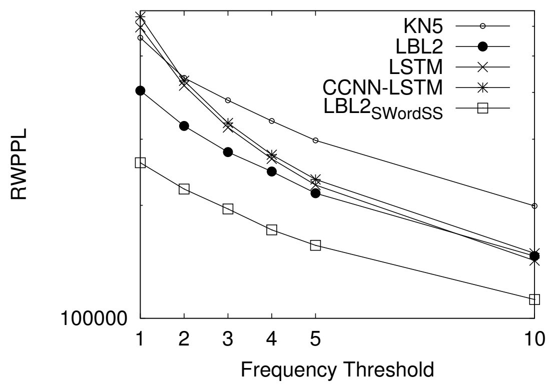
For Vietnamese, LBL alone was able to improve OOV and RW1 words over the other LMs. We attribute this to lower coverage of Vietnamese rare words by SWordSS than for other languages. Instead adding SWordSS embeddings harmed the prediction of OOV and RW1 words.

These perplexity improvements stared to wane when higher frequency words were included into the rare word set, across the different languages. Nevertheless, for languages with rich morphology, initialising LBL with SWordSS embeddings reduced perplexities on rare words.

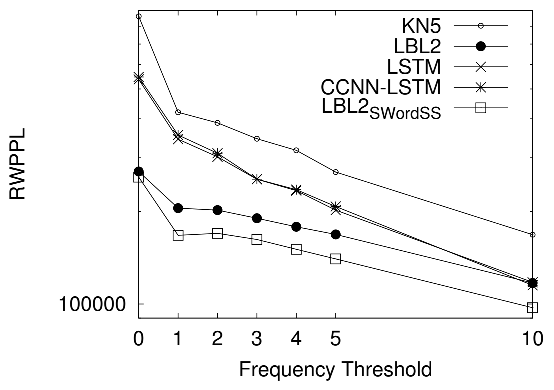
5 Conclusion

In this paper, we introduced SWordSS, a novel sub-word similarity based search for generating rare word embeddings. It leverages the sub-word similarity in morphologically rich languages to search for close

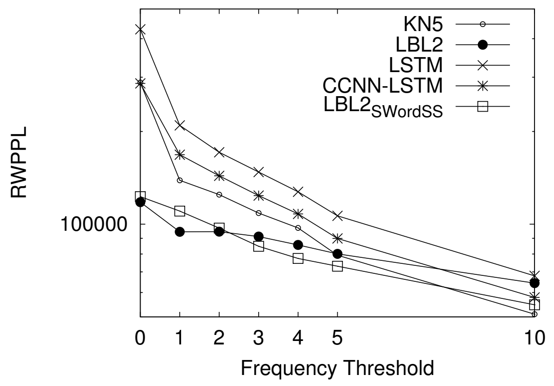
[4]when initialised with SWordSS embeddings it obtained the same perplexity values



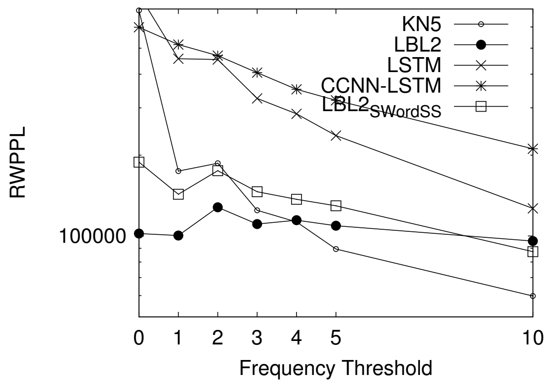
(a) German



(b) Tagalog



(c) Turkish



(d) Vietnamese

Figure 1: Variation of rare-word perplexity versus threshold on frequency of training-set words on German, Tagalog, Turkish and Vietnamese corpora

matches of a rare word, and then combines these close matches to estimate the embedding of a rare word.

Even though SWordSS is an unsupervised approach like Soricut and Och (2015), it differs from latter in the way it utilises the morphological information. The latter automatically induces morphological rules and transformations to build a morphological word graph. This graph is then tuned and used to induce embedding of a rare word. Instead, SWordSS replaces the overhead of induction of rules and creation of graph by searching a sub-word inverted index to find rare-word matches and combining their embeddings to estimate rare-word embedding.

To test the SWordSS technique, we augmented pre-trained embeddings and then evaluated them on word similarity tasks. The augmented embeddings outperformed the initial set of embeddings drastically. However, it lagged behind the state-of-the-art performance of Soricut and Och (2015). But, by employing embeddings trained on larger corpora, SWordSS was able to perform comparably on a rare-word task.

We also investigated the effects of using SWordSS augmented embeddings for modelling rare words. To perform this experiment, we trained LBL$_{SWordSS}$ LM and compared it with language models like the character aware LM, LSTM-based RNN LM restricted to similar size. On almost all datasets, the character aware LM outperformed the other LMs with respect to perplexity on complete test sets. But on rare words, SWordSS showed up to 50 % reduced perplexity values in comparison to other LMs. Hence, SWordSS embeddings contributed substantially in modelling rare-word tasks.

In future work, we plan to incorporate SWordSS embeddings into more complex LMs than LBL and further analyse the different string similarity functions used in SWordSS's formulation.

Acknowledgments

We would like to thank anonymous reviewers for their comments, which helped improve this paper. We are also immensely grateful to Rose Hoberman for her comments on an earlier version of this manuscript.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

Samy Bengio and Georg Heigold. 2014. Word embeddings for speech recognition. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association*, pages 1053–1057, Singapore, September.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA. ACM.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. *CoRR*, abs/1405.4273.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA, June. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*.

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)*, pages 1651–1660, Berlin, Germany, August. Association for Computational Linguistics.

M. Creutz and K. Lagus. 2005. Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0. Technical report, Helsinki University of Technology.

Iryna Gurevych, 2005. *Natural Language Processing – IJCNLP 2005: Second International Joint Conference, Jeju Island, Korea, October 11-13, 2005. Proceedings*, chapter Using the Structure of a Conceptual Network in Computing Semantic Relatedness, pages 767–778. Springer Berlin Heidelberg, Berlin, Heidelberg.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. *CoRR*, abs/1404.4641.

Paul Jaccard. 1912. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, February.

Matthew A. Jaro. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444.

Beth Logan, Pedro Moreno, Jean-Manuel Van Thong, et al. 1996. An experimental study of an audio indexing system for the web. In *Proceedings of the 4th International Conference of Spoken Language Processing*. Citeseer.

B. Logan, J. M. Van Thong, and P. J. Moreno. 2005. Approaches to reduce the effects of oov queries on indexed spoken audio. *IEEE Transactions on Multimedia*, 7(5):899–906, Oct.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Michael McCandless, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co., Greenwich, CT, USA.

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

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 641–648, New York, NY, USA. ACM.

Sadi Evren Seker, Oguz Altun, Ugur Ayan, and Cihan Mert. 2014. A novel string distance function based on most frequent K characters. *CoRR*, abs/1401.6596.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D. Manning, and Andrew Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, July. Association for Computational Linguistics.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637, Denver, Colorado, May–June. Association for Computational Linguistics.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *INTERSPEECH*, pages 194–197.

Amos Tversky. 1977. Features of similarity. *Psychological review*, 84(4):327.

Yiming Yang, Hanxiao Liu, Jaime Carbonell, and Wanli Ma. 2015. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 159–168, New York, NY, USA. ACM.