Letter N-Gram-based Input Encoding for Continuous Space Language Models

Henning Sperr[†], **Jan Niehues**[⋆] **and Alexander Waibel**[⋆]
Institute of Anthropomatics
KIT - Karlsruhe Institute of Technology
Karlsruhe, Germany
[⋆] firstname.lastname@kit.edu
[†] henning.sperr@student.kit.edu

Abstract

We present a letter-based encoding for words in continuous space language models. We represent the words completely by letter n-grams instead of using the word index. This way, similar words will automatically have a similar representation. With this we hope to better generalize to unknown or rare words and to also capture morphological information. We show their influence in the task of machine translation using continuous space language models based on restricted Boltzmann machines. We evaluate the translation quality as well as the training time on a German-to-English translation task of TED and university lectures as well as on the news translation task translating from English-to-German. Using our new approach a gain in BLEU score by up to 0.4 points can be achieved.

1 Introduction

Language models play an important role in natural language processing. The most commonly used approach is n-gram-based language models (Chen and Goodman, 1999).

In recent years Continuous Space Language Models (CSLMs) have gained a lot of attention. Compared to standard n-gram-based language models they promise better generalization to unknown histories or n-grams with only few occurrences. Since the words are projected into a continuous space, true interpolation can be performed when an unseen sample appears. The standard input layer for CSLMs is a so called 1-of-n coding where a word is represented as a vector with a single neuron turned on and the rest turned off. In the standard approach it is problematic to infer probabilities for words that are not inside the vocabulary. Sometimes an extra unknown neuron is used in the input layer to represent these words (Niehues and Waibel, 2012). Since all unseen words get mapped to the same neuron, no real discrimination between those words can be done. Furthermore, rare words are also hard to model, since there is too few training data available to estimate their associated parameters.

We try to overcome these shortcomings by using subword features to cluster similar words closer together and generalize better over unseen words. We hope that words containing similar letter n-grams will yield a good indicator for words that have the same function inside the sentence. Introducing a method for subword units also has the advantage that the input layer can be smaller, while still representing nearly the same vocabulary with unique feature vectors. By using a smaller input layer, less weights need to be trained and the training is faster. In this work we present the letter n-gram approach to represent words in an CSLM, and compare it to the word-based CSLM presented in Niehues and Waibel (2012).

The rest of this paper is structured as follows: First we will give an overview of related work. After that we give a brief overview of restricted Boltzmann machines which are the basis of the letter-based CSLM presented in Section 4. Then we will present the results of the experiments and conclude our work.

2 Related Work

First research on neural networks to predict word categories has been done in Nakamura et al. (1990) where neural networks were used to predict word categories. Xu and Rudnicky (2000) proposed a language model that has an input consisting of one word and no hidden units. This network was limited to infer unigram and bigram statistics. There has been research on feed forward neural network language models where they achieved a decrease in perplexity compared to standard n-gram language models (Bengio et al., 2003). In Schwenk and Gauvain (2005) and later in Schwenk (2007) research was performed on training large scale neural network language models on millions of words resulting in a decrease of the word error rate for continuous speech recognition. In Schwenk et al. (2006) they use the CSLM framework to rescore n-best lists of a machine translation system during tuning and testing steps. Usually these networks use short lists to reduce the size of the output layer and to make calculation feasible. There have been approaches to optimize the output layer of such a network, so that vocabularies of arbitrary size can be used and there is no need to back off to a smaller n-gram model (Le et al., 2011). In this Structured Output Layer (SOUL) neural network language model a hierarchical output layer was chosen. Recurrent Neural Networks have also been used to try and improve language model perplexities in Mikolov et al. (2010), concluding that Recurrent Neural Networks potentially improve over classical n-gram language models with increasing data and a big enough hidden unit size of the model. In the work of Mnih and Hinton (2007) and Mnih (2010) training factored restricted Boltzmann machines yielded no gain compared to Kneser-Ney smoothed n-gram models. But it has been shown in Niehues and Waibel (2012), that using a restricted Boltzmann machine with a different layout during decoding can yield an increase in BLEU score. There has also been a lot of research in the field of using subword units for language modeling. In Shaik et al. (2011) linguistically motivated sub-lexical units were proposed to improve open vocabulary speech recognition for German. Research on morphology-based and subword language models on a Turkish speech recognition task has been done by Sak et al. (2010). The idea of Factored Language models in machine translation has been introduced by Kirchhoff and Yang (2005). Similar approaches to develop joint language models for morphologically rich languages in machine translation have been presented by Sarikaya and Deng (2007). In Emami et al. (2008) a factored neural network language model for Arabic was built. They used different features such as segmentation, part-of-speech and diacritics to enrich the information for each word.

3 Restricted Boltzmann Machine-based Language Model

In this section we will briefly review the continuous space language models using restricted Boltzmann machines (RBM). We will focus on the parts that are important for the implementation of the input layers described in the next section. A restricted Boltzmann machine is a generative stochastic neural network which consists of a visible and a hidden layer of neurons that have unidirectional connections between the layers but no inner layer connections as shown in Figure 1.

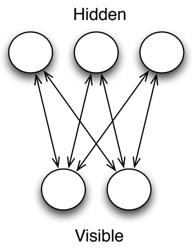

Figure 1: Restricted Boltzmann Machine.

The activation of the visible neurons will be determined by the input data. The standard input layer for neural network language models uses a 1-of-n coding to insert a word from the vocabulary into the network. This is a vector, where only the index of the word in the vocabulary is set to one and the rest to zero. Sometimes this is also referred to as a *softmax* layer of binary units. The activation of the hidden units is usually binary and will be inferred from the visible units by using sampling techniques. In Niehues and Waibel (2012) an n-gram Boltzmann machine language model is proposed using such a softmax layer for each context. In this work, we want to explore different ways of encoding the word observations in the input layer. Figure 2 is an example of the original model with three hidden units, two contexts and a vocabulary of two words. In this example the bigram *my house* is modeled.

To calculate the probability of a visible configuration v we will use the definition of the free energy in a restricted Boltzmann machine with binary stochastic hidden units, which is

$$F(v) = -\sum_i v_i a_i - \sum_j log(1 + \mathrm{e}^{x_j}) \qquad (1)$$

where a_i is the bias of the ith visible neuron v_i and

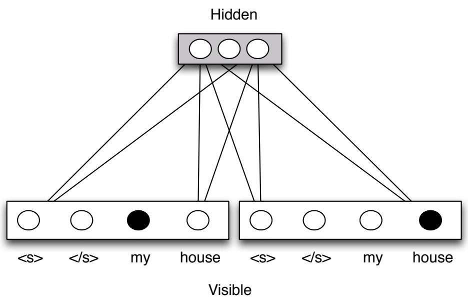

Figure 2: RBMLM with three hidden units and a vocabulary size of two words and two word contexts, where activated units are marked as black.

x_j is the activation of the jth hidden neuron. The activation x_j is defined as

$$x_j = b_j + \sum_i v_i w_{ij} \tag{2}$$

where b_j is the bias of the jth hidden neuron and w_{ij} is the weight between visible unit v_i and hidden unit x_j. Using these definitions, the probability of our visible configuration v is

$$p(v) = \frac{1}{Z} \mathrm{e}^{-F(v)} \tag{3}$$

with the partition function $Z = \sum_v \mathrm{e}^{-F(v)}$ being the normalization constant. Usually this normalization constant is not easy to compute since it is a sum over an exponential amount of values. We know that the free energy will be proportional to the true probability of our visible vector, this is the reason for using the free energy as a feature in our log-linear model instead of the true probability. In order to use it as a feature inside the decoder we actually need to be able to compute the probability for a whole sentence. As shown in Niehues and Waibel (2012) we can do this by summing over the free energy of all n-grams contained in the sentence.

3.1 Training

For training the restricted Boltzmann machine language model (RBMLM) we used the Contrastive Divergence (CD) algorithm as proposed in Hinton (2010). In order to do this, we need to calculate the derivation of the probability of the example given the weights

$$\frac{\delta \log p(v)}{\delta w_{ij}} = <v_i h_j>_{data} - <v_i h_j>_{model} \tag{4}$$

where $<v_i h_j>_{model}$ is the expected value of $v_i h_j$ given the distribution of the model. In other words we calculate the expectation of how often v_i and h_j are activated together, given the distribution of the data, minus the expectation of them being activated together given the distribution of the model, which will be calculated using Gibbs-Sampling techniques. Usually many steps of Gibbs-Sampling are necessary to get an unbiased sample from the distribution, but in the Contrastive Divergence algorithm only one step of sampling is performed (Hinton, 2002).

4 Letter-based Word Encoding

In this section we will describe the proposed input layers for the RBMLM. Compared to the word index-based representation explained above, we try to improve the capability to handle unknown words and morphology by splitting the word into subunits.

4.1 Motivation

In the example mentioned above, the word index model might be able to predict *my house* but it will fail on *my houses* if the word *houses* is not in the training vocabulary. In this case, a neuron that classifies all unknown tokens or some other techniques to handle such a case have to be utilized.

In contrast, a human will look at the single letters and see that these words are quite similar. He will most probably recognize that the appended *s* is used to mark the plural form, but both words refer to the same thing. So he will be able to infer the meaning although he has never seen it before.

Another example in English are be the words *happy* and *unhappy*. A human speaker who does not know the word *unhappy* will be able to know from the context what *unhappy* means and he can guess that both of the words are adjectives, that have to do with happiness, and that they can be used in the same way.

In other languages with a richer morphology, like German, this problem is even more important. The German word *schön* (engl. *beautiful*) can have 16 different word forms, depending on case, number and gender.

Humans are able to share information about words that are different only in some morphemes like *house* and *houses*. With our letter-based input encoding we want to generalize over the common word index model to capture morphological information about the words to make better predictions for unknown words.

4.2 Features

In order to model the aforementioned morphological word forms, we need to create features for every word that represent which letters are used in the word. If we look at the example of *house*, we need to model that the first letter is an h, the second is an o and so on.

If we want to encode a word this way, we have the problem that we do not have a fixed size of features, but the feature size depends on the length of the word. This is not possible in the framework of continuous space language models. Therefore, a different way to represent the word is needed.

An approach for having a fixed size of features is to just model which letters occur in the word. In this case, every input word is represented by a vector of dimension n, where n is the size of the alphabet in the text. Every symbol, that is used in the word is set to one and all the other features are zero. By using a sparse representation, which shows only the features that are activated, the word *house* would be represented by

$$w_1 = \text{e h o s u}$$

The main problem of this representation is that we lose all information about the order of the letters. It is no longer possible to see how the word ends and how the word starts. Furthermore, many words will be represented by the same feature vector. For example, in our case the words *house* and *houses* would be identical. In the case of *houses* and *house* this might not be bad, but the words *shortest* and *others* or *follow* and *wolf* will also map to the same input vector. These words have no real connection as they are different in meaning and part of speech.

Therefore, we need to improve this approach to find a better model for input words. N-grams of words or letters have been successfully used to model sequences of words or letters in language models. We extend our approach to model not only the letters that occur in the in the word, but the letter n-grams that occur in the word. This will of course increase the dimension of the feature space, but then we are able to model the order of the letters. In the example of *my house* the feature vector will look like

$$w_1 = \text{my <w>m y</w>}$$
$$w_2 = \text{ho ou se us <w>h e</w>}$$

We added markers for the beginning and end of the word because this additional information is important to distinguish words. Using the example of the word *houses*, modeling directly that the last letter is an s could serve as an indication of a plural form.

If we use higher order n-grams, this will increase the information about the order of the letters. But these letter n-grams will also occur more rarely and therefore, the weights of these features in the RBM can no longer be estimated as reliably. To overcome this, we did not only use the n-grams of order n, but all n-grams of order n and smaller. In the last example, we will not only use the bigrams, but also the unigrams.

This means *my house* is actually represented as

$$w_1 = \text{m y my <w>m y</w>}$$
$$w_2 = \text{e h o s u ho ou se us <w>h e</w>}$$

With this we hope to capture many morphological variants of the word *house*. Now the representations of the words *house* and *houses* differ only in the ending and in an additional bigram.

$$\textit{houses} = \text{... es s</w>}$$
$$\textit{house} = \text{... se e</w>}$$

The beginning letters of the two words will contribute to the same free energy only leaving the ending letter n-grams to contribute to the different usages of *houses* and *house*.

The actual layout of the model can be seen in Figure 3. For the sake of clarity we left out the unigram letters. In this representation we now do not use a softmax input layer, but a logistic input layer defined as

$$p(v_i = \text{on}) = \frac{1}{1 + \mathrm{e}^{-x_i}} \qquad (5)$$

where v_i is the ith visible neuron and x_i is the input from the hidden units for the ith neuron defined as

$$x_i = a_i + \sum_j h_j w_{ij} \qquad (6)$$

with a_i being the bias of the visible neuron v_i and w_{ij} being the weight between the hidden unit h_j and v_i.

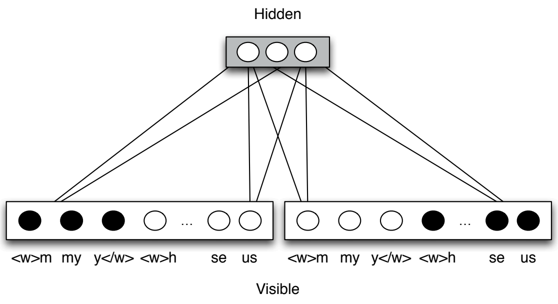

Figure 3: A bigram letter index RBMLM with three hidden units and two word contexts, where activated units are marked as black.

4.3 Additional Information

The letter index approach can be extended by several features to include additional information about the words. This could for example be part-of-speech tags or other morphological information. In this work we tried to include a neuron to capture capital letter information. To do this we included a neuron that will be turned on if the first letter was capitalized and another neuron that will be turned on if the word was written in all capital letters. The word itself will be lowercased after we extracted this information.

Using the example of *European Union*, the new input vector will look like this

$$w_1 = \text{a e n o p r u an ea eu op pe ro ur}$$
$$\text{<w>e n</w><CAPS>}$$
$$w_2 = \text{u i n o un io ni on}$$
$$\text{<w>u n</w><CAPS>}$$

This will lead to a smaller letter n-gram vocabulary since all the letter n-grams get lowercased. This also means there is more data for each of the letter n-gram neurons that were treated differently before. We also introduced an all caps feature which is turned on if the whole word was written in capital letters. We hope that this can help detect abbreviations which are usually written in all capital letters. For example *EU* will be represented as

$$w_1 = \text{e u eu <w>e u</w><ALLCAPS>}$$

5 Evaluation

We evaluated the RBM-based language model on different statistical machine translation (SMT) tasks. We will first analyze the letter-based word representation. Then we will give a brief description of our SMT system. Afterwards, we describe in detail our experiments on the German-to-English translation task. We will end with additional experiments on the task of translating English news documents into German.

5.1 Word Representation

In first experiments we analyzed whether the letter-based representation is able to distinguish between different words. In a vocabulary of 27,748 words, we compared for different letter n-gram sizes how many words are mapped to the same input feature vector.

Table 1 shows the different models, their input dimensions and the total number of unique clusters as well as the amount of input vectors containing one, two, three or four or more words that get mapped to this input vector. In the word index representation every word has its own feature vector. In this case the dimension of the input vector is 27,748 and each word has its own unique input vector.

If we use only letters, as done in the unigram model, only 62% of the words have a unique representation. Furthermore, there are 606 feature vectors representing 4 or more words. This type of encoding of the words is not sufficient for the task.

When using a bigram letter context nearly each of the 27,748 words has a unique input representation, although the input dimension is only 7% compared to the word index. With the three letter vocabulary context and higher there is no input vector that represents more than three words from the vocabulary. This is good since we want similar words to be close together but not have exactly the same input vector. The words that are still clustered to the same input are mostly numbers or typing mistakes like “YouTube” and “Youtube”.

5.2 Translation System Description

The translation system for the German-to-English task was trained on the European Parliament corpus, News Commentary corpus, the BTEC corpus and TED talks[1]. The data was preprocessed and compound splitting was applied for German. Afterwards the discriminative word alignment approach as described in Niehues and Vogel (2008) was applied to generate the alignments between source and target words. The phrase table was

[1]http://www.ted.com

				#Vectors mapping to			
Model	Caps	VocSize	TotalVectors	1 Word	2 Words	3 Words	4+ Words
WordIndex	-	27,748	27,748	27,748	0	0	0
Letter 1-gram	No	107	21,216	17,319	2,559	732	606
Letter 2-gram	No	1,879	27,671	27,620	33	10	8
Letter 3-gram	No	12,139	27,720	27,701	10	9	0
Letter 3-gram	Yes	8,675	27,710	27,681	20	9	0
Letter 4-gram	No	43,903	27,737	27,727	9	1	0
Letter 4-gram	Yes	25,942	27,728	27,709	18	1	0

Table 1: Comparison of the vocabulary size and the possibility to have a unique representation of each word in the training corpus.

built using the scripts from the Moses package described in Koehn et al. (2007). A 4-gram language model was trained on the target side of the parallel data using the SRILM toolkit from Stolcke (2002). In addition, we used a bilingual language model as described in Niehues et al. (2011). Reordering was performed as a preprocessing step using part-of-speech (POS) information generated by the TreeTagger (Schmid, 1994). We used the reordering approach described in Rottmann and Vogel (2007) and the extensions presented in Niehues et al. (2009) to cover long-range reorderings, which are typical when translating between German and English. An in-house phrase-based decoder was used to generate the translation hypotheses and the optimization was performed using the MERT implementation as presented in Venugopal et al. (2005). All our evaluation scores are measured using the BLEU metric.

We trained the RBMLM models on 50K sentences from TED talks and optimized the weights of the log-linear model on a separate set of TED talks. For all experiments the RBMLMs have been trained with a context of four words. The development set consists of 1.7K segments containing 16K words. We used two different test sets to evaluate our models. The first test set contains TED talks with 3.5K segments containing 31K words. The second task was from an in-house computer science lecture corpus containing 2.1K segments and 47K words. For both tasks we used the weights optimized on TED.

For the task of translating English news texts into German we used a system developed for the Workshop on Machine Translation (WMT) evaluation. The continuous space language models were trained on a random subsample of 100K sentences from the monolingual training data used for this task. The out-of-vocabulary rates for the TED task are 1.06% while the computer science lectures have 2.73% and nearly 1% on WMT.

5.3 German-to-English TED Task

The results for the translation of German TED lectures into English are shown in Table 2. The baseline system uses a 4-gram Kneser-Ney smoothed language model trained on the target side parallel data. We then added a RBMLM, which was only trained on the English side of the TED corpus.

If the word index RBMLM trained for one iteration using 32 hidden units is added, an improvement of about 1 BLEU can be achieved. The letter bigram model performs about 0.4 BLEU points better than no additional model, but significantly worse then the word index model or the other letter n-gram models. The letter 3- to 5-gram-based models obtain similar BLEU scores, varying only by 0.1 BLEU point. They also achieve a 0.8 to 0.9 BLEU points improvement against the baseline system and a 0.2 to 0.1 BLEU points decrease than the word index-based encoding.

System	Dev	Test
Baseline	26.31	23.02
+WordIndex	**27.27**	**24.04**
+Letter 2-gram	26.67	23.44
+Letter 3-gram	26.80	23.84
+Letter 4-gram	26.79	23.93
+Letter 5-gram	26.64	23.82

Table 2: Results for German-to-English TED translation task

Using the word index model with the first baseline system increases the BLEU score nearly as much as adding a n-gram-based language model trained on the TED corpus as done in the baseline of the systems presented in Table 3. In these experiments all letter-based models outperformed the baseline system. The bigram-based language model performs worst and the 3- and 4-gram-based models perform only slightly worse than the word index-based model.

System	Dev	Test
Baseline+ngram	27.45	24.06
+WordIndex	**27.70**	**24.34**
+Letter 2-gram	27.45	24.15
+Letter 3-gram	27.52	24.25
+Letter 4-gram	27.60	24.30

Table 3: Results of German-to-English TED translations using an additional in-domain language model.

A third experiment is presented in Table 4. Here we also applied phrase table adaptation as described in Niehues et al. (2010). In this experiment the word index model improves the system by 0.4 BLEU points. In this case all letter-based models perform very similar. They are again performing slightly worse than the word index-based system, but better than the baseline system.

To summarize the results, we could always improve the performance of the system by adding the letter n-gram-based language model. Furthermore, in most cases, the bigram model performs worse than the higher order models. It seems to be important for this task to have more context information. The 3- and 4-gram-based models perform almost equal, but slightly worse than the word index-based model.

System	Dev	Test
BL+ngram+adaptpt	28.40	24.57
+WordIndex	**28.55**	**24.96**
+Letter 2-gram	28.31	24.80
+Letter 3-gram	28.31	24.71
+Letter 4-gram	28.46	24.65

Table 4: Results of German-to-English TED translations with additional in-domain language model and adapted phrase table.

5.3.1 Caps Feature

In addition, we evaluated the proposed caps feature compared to the non-caps letter n-gram model and the baseline systems. As we can see in Table 5, caps sometimes improves and sometimes decreases the BLEU score by about ±0.2 BLEU points. One reason for that might be that most English words are written lowercased, therefore we do not gain much information.

System	Dev	Test
Baseline	26.31	23.02
+Letter 3-gram	**26.80**	23.84
+Letter 3-gram+caps	26.67	**23.85**
Baseline+ngram	27.45	24.06
+Letter 3-gram	27.52	24.25
+Letter 3-gram+caps	**27.60**	**24.47**
BL+ngram+adaptpt	28.40	24.57
+Letter 3-gram	28.31	**24.71**
+Letter 3-gram+caps	**28.43**	24.66

Table 5: Difference between caps and non-caps letter n-gram models.

5.4 German-to-English CSL Task

After that, we evaluated the computer science lecture (CSL) test set. We used the same system as for the TED translation task. We did not perform a new optimization, since we wanted so see how well the models performed on a different task.

The results are summarized in Table 6. In this case the baseline is outperformed by the word index approach by approximately 1.1 BLEU points. Except for the 4-gram model the results are similar to the result for the TED task. All systems could again outperform the baseline.

System	Test
Baseline	23.60
+WordIndex	**24.76**
+Letter 2-gram	24.17
+Letter 3-gram	24.36
+Letter 4-gram	23.82

Table 6: Results the baseline of the German-to-English CSL task.

The system with the additional in-domain language model in Table 7 shows that both letter n-gram language models perform better than the baseline and the word index model, improving the baseline by about 0.8 to 1 BLEU. Whereas the word index model only achieved an improvement of 0.6 BLEU points.

The results of the system with the additional phrase table adaption can be seen in Table 8. The

System	Test
Baseline+ngram	23.81
+WordIndex	24.41
+Letter 2-gram	24.37
+Letter 3-gram	24.66
+Letter 4-gram	**24.85**

Table 7: Results on German-to-English CSL corpus with additional in-domain language model.

word index model improves the baseline by 0.25 BLEU points. The letter n-gram models improve the baseline by about 0.3 to 0.4 BLEU points also improving over the word index model. The letter bigram model in this case performs worse than the baseline.

System	Test
BL+ngram+adaptpt	25.00
+WordIndex	25.25
+Letter 2-gram	24.68
+Letter 3-gram	**25.43**
+Letter 4-gram	25.33

Table 8: Results on German-to-English CSL with additional in-domain language model and adapted phrase table.

In summary, again the 3- and 4-gram letter models perform mostly better than the bigram version. They both perform mostly equal. In contrast to the TED task, they were even able to outperform the word index model in some configurations by up to 0.4 BLEU points.

5.5 English-to-German News Task

When translating English-to-German news we could not improve the performance of the baseline by using a word index model. In contrast, the performance dropped by 0.1 BLEU points. If we use a letter bigram model, we could improve the translation quality by 0.1 BLEU points over the baseline system.

System	Dev	Test
Baseline	16.90	17.36
+WordIndex	16.79	17.29
+Letter 2-gram	**16.91**	**17.48**

Table 9: Results for WMT2013 task English-to-German.

5.6 Model Size and Training Times

In general the letter n-gram models perform almost as good as the word index model on English language tasks. The advantage of the models up to the letter 3-gram context model is that the training time is lower compared to the word index model. All the models were trained using 10 cores and a batch size of 10 samples per contrastive divergence update. As can be seen in Table 10 the letter 3-gram model needs less than 50% of the weights and takes around 75% of the training time of the word index model. The four letter n-gram model takes longer to train due to more parameters.

Model	#Weights	Time
WordIndex	3.55 M	20 h 10 min
Letter 2-gram	0.24 M	1h 24 min
Letter 3-gram	1.55 M	15 h 12 min
Letter 4-gram	5.62 M	38 h 59 min

Table 10: Training time and number of parameters of the RBMLM models.

6 Conclusions

In this work we presented the letter n-gram-based input layer for continuous space language models. The proposed input layer enables us to encode the similarity of unknown words directly in the input layer as well as to directly model some morphological word forms.

We evaluated the encoding on different translation tasks. The RBMLM using this encoding could always improve the translation quality and perform similar to a RBMLM based on word indices. Especially in the second configuration which had a higher OOV rate, the letter n-gram model performed better than the word index model. Moreover, the model based on letter 3-grams uses only half the parameters of the word index model. This reduced the training time of the continuous space language model by a quarter.

Acknowledgments

This work was partly achieved as part of the Quaero Programme, funded by OSEO, French State agency for innovation. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287658.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Ahmad Emami, Imed Zitouni, and Lidia Mangu. 2008. Rich morphology based n-gram language models for arabic. In *INTERSPEECH*, pages 829–832. ISCA.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August.

Geoffrey Hinton. 2010. A Practical Guide to Training Restricted Boltzmann Machines. Technical report, University of Toronto.

Katrin Kirchhoff and Mei Yang. 2005. Improved Language Modeling for Statistical Machine Translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 125–128, Ann Arbor, Michigan, USA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL 2007, Demonstration Session*, Prague, Czech Republic.

Hai Son Le, Ilya Oparin, Abdelkhalek Messaoudi, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. 2011. Large vocabulary soul neural network language models. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy.*

Tomas Mikolov, Martin Karafiát, Lukas 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.*

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.

Andriy Mnih. 2010. *Learning distributed representations for statistical language modelling and collaborative filtering*. Ph.D. thesis, University of Toronto, Toronto, Ont., Canada. AAINR73159.

Masami Nakamura, Katsuteru Maruyama, Takeshi Kawabata, and Kiyohiro Shikano. 1990. Neural network approach to word category prediction for english texts. In *Proceedings of the 13th conference on Computational linguistics - Volume 3*, COLING '90, pages 213–218, Stroudsburg, PA, USA.

Jan Niehues and Muntsin Kolss. 2009. A POS-Based Model for Long-Range Reorderings in SMT. In *Fourth Workshop on Statistical Machine Translation (WMT 2009)*, Athens, Greece.

Jan Niehues and Stephan Vogel. 2008. Discriminative Word Alignment via Alignment Matrix Modeling. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 18–25, Stroudsburg, PA, USA.

Jan Niehues and Alex Waibel. 2012. Continuous Space Language Models using Restricted Boltzmann Machines. In *Proceedings of the International Workshop for Spoken Language Translation (IWSLT 2012)*, Hong Kong.

Jan Niehues, Mohammed Mediani, Teresa Herrmann, Michael Heck, Christian Herff, and Alex Waibel. 2010. The KIT Translation system for IWSLT 2010. In *Proceedings of the Seventh International Workshop on Spoken Language Translation (IWSLT)*, Paris, France.

Jan Niehues, Teresa Herrmann, Stephan Vogel, and Alex Waibel. 2011. Wider Context by Using Bilingual Language Models in Machine Translation. In *Sixth Workshop on Statistical Machine Translation (WMT 2011)*, Edinburgh, UK.

Kay Rottmann and Stephan Vogel. 2007. Word Reordering in Statistical Machine Translation with a POS-Based Distortion Model. In *TMI*, Skövde, Sweden.

Hasim Sak, Murat Saraclar, and Tunga Gungor. 2010. Morphology-based and sub-word language modeling for Turkish speech recognition. In *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 5402–5405.

Ruhi Sarikaya and Yonggang Deng. 2007. Joint Morphological-Lexical Language Modeling for Machine Translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 145–148, Rochester, New York, USA.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing*, Manchester, UK.

Holger Schwenk and Jean-Luc Gauvain. 2005. Training neural network language models on very large corpora. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 201–208, Stroudsburg, PA, USA.

Holger Schwenk, Daniel Dchelotte, and Jean-Luc Gauvain. 2006. Continuous Space Language mModels for Statistical Machine T ranslation. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 723–730, Stroudsburg, PA, USA.

Holger Schwenk. 2007. Continuous Space Language Models. *Comput. Speech Lang.*, 21(3):492–518, July.

M. Ali Basha Shaik, Amr El-Desoky Mousa, Ralf Schlüter, and Hermann Ney. 2011. Hybrid language models using mixed types of sub-lexical units for open vocabulary german lvcsr. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy.*

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *7th International Conference on Spoken Language Processing, ICSLP2002/INTERSPEECH 2002, Denver, Colorado, USA.*

Ashish Venugopal, Andreas Zollman, and Alex Waibel. 2005. Training and Evaluation Error Minimization Rules for Statistical Machine Translation. In *Workshop on Data-driven Machine Translation and Beyond (WPT-05)*, Ann Arbor, MI, USA.

Wei Xu and Alex Rudnicky. 2000. Can artificial neural networks learn language models? In *Sixth International Conference on Spoken Language Processing, ICSLP 2000 / INTERSPEECH 2000, Beijing, China*, pages 202–205. ISCA.