Continuous Distributed Representations of Words as Input of LSTM Network Language Model

Daniel Soutner and Luděk Müller

University of West Bohemia, Faculty of Applied Sciences
New Technologies for the Information Society
Univerzitní 22, 306 14 Plzeň, Czech Republic
{dsoutner,muller}@ntis.zcu.cz

Abstract. The continuous skip-gram model is an efficient algorithm for learning quality distributed vector representations that are able to capture a large number of syntactic and semantic word relationships. Artificial neural networks have become the state-of-the-art in the task of language modelling whereas Long-Short Term Memory (LSTM) networks seem to be efficient training algorithm.
In this paper, we carry out experiments with a combination of these powerful models: the continuous distributed representations of words are trained with skip-gram method on a big corpora and are used as the input of LSTM language model instead of traditional 1-of-N coding. The possibilities of this approach are shown in experiments on perplexity with Wikipedia and Penn Treebank corpus.

Keywords: language modelling, neural networks, LSTM, skip-gram, word2vec

1 Introduction

Language modelling is a crucial task in many areas of natural language processing. Speech recognition, optical character recognition and many other areas heavily depend on the performance of the language model that is being used. Each improvement in the language modelling may also improve the particular job where the language model is used.

In a lot of applications we are not able to obtain enough amount of the in-domain data for a language model; for example in case of specific style of text or spontaneous speech. Sometimes, adding more out-domain data do not cover our performance demands.

We have focused in our work whether the state-of-the-art models could be improved by adding some information from an out-domain corpus. More specifically, we are exploring the possibilities of two currently favourite LM techniques: *word2vec* architecture for training word distributed representations and LSTM neural networks.

Recurrent neural networks (RNN) have attracted much attention in last years among other types of language models (LM) caused by their better performance [1] and their ability to learn on a smaller corpus than conventional n-gram models. Skip-gram and continuous bag of words (CBOW) [2] – sometimes deonted as *word2vec* – are recently developed technique for building a neural network that maps words to real-number vectors, with the desideratum that words with similar meanings will be mapped to the similar vectors.

There are described LSTM neural network language model and *word2vec* architecture for training continuous distributed representations of words in the Section 2; the introduction of our proposed language model which combines both of them follows. Section 3 deals with experiments and results and Section 4 summarizes the results and draws some conclusions.

2 Model Structure

In this Section the two main models we were working with are described: LSTM language model and the *word2vec* model for training continuous distributed representations of words from text. Afterwards we proposed our fused model, where we are using word vectors trained from *word2vec* as the input to LSTM neural net instead of standard 1-of-N coding inputs.

2.1 LSTM Language Model

The long-short term memory neural networks were successfully introduced to the field of language modelling by [3] and the basic scheme is shown in Figure 1. LM is based (similarly to conventional recurrent neural network) on these principals:

– The input vector is a word encoded in 1-of-N coding.
– By the training the output vector is also in 1-of-N coding.
– There is a softmax function used in output layer to produce normalized probabilities.
– The cross entropy is used as training criterion.

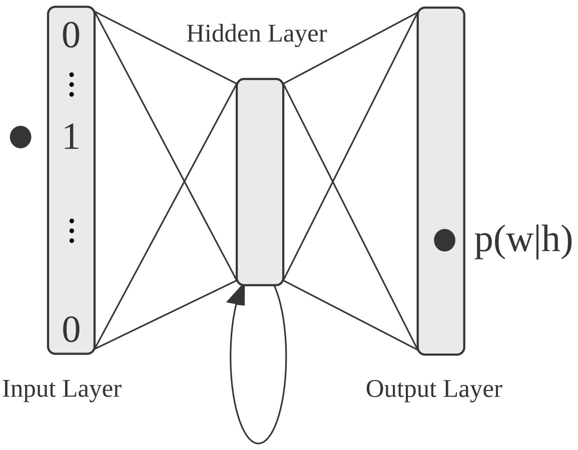

Fig. 1. LSTM language model scheme.

The vanishing gradient seemed to be problematic during the training of recurrent neural networks as is shown in [4]. This led authors to re-design the network unit, in

LSTM called as a *cell*. Every LSTM cell contains *gates* that determine when the input is significant enough to remember, when it should continue to remember or forget the value, and when it should output the value.

Regular back-propagation could be effective at training LSTM cell due to this specific topology of LSTM, especially because of a constant error flow. This leads to easier model training and in LM application these networks capture even a longer context then standard RNN as it is showed in [3] and perform better on dynamic evaluation [5].

2.2 Distributed Representations of Words

Representation of text is very important for performance in many real-world applications. The most commonly used techniques in these days are *n*-grams, bag-of-words, 1-of-N coding (which all belongs to local representations). Continuous representations of the words, which represent words as a vector with real numbers are for example LDA, LSA or distributed representations. Distributed representations of words can be obtained from various NN-based language models.

Mikolov et al. in [2] introduced new architecture for training distributed word representations which belongs to class of methods called ”neural language models”. Using a scheme that is much simpler than previous work in this domain, where neural networks with many hidden units and several non-linear layers were normally constructed (e.g., [6]), *word2vec* constructs a simple log-linear classification network [7]. There were proposed two architectures: continuous bag-of-words (cbow) predicts the context given the word, continuous skip-gram predicts the current word given the context. The basic scheme of model is shown in Figure 2. The input words are encoded in 1-of-N coding, the model is trained with hierarchical softmax, a context in interval 5-10 word is usually considered.

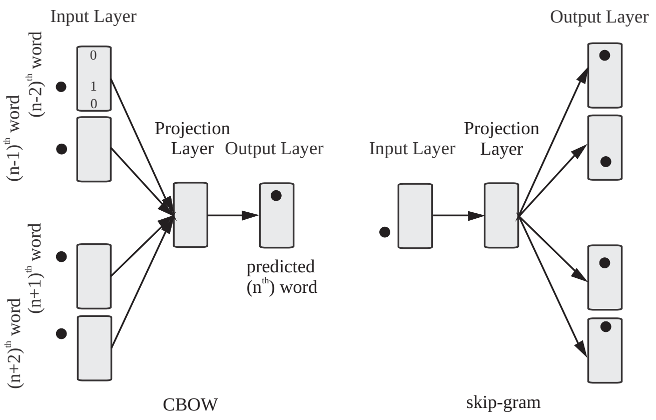

Fig. 2. *The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.*

This model provides more efficient word vectors than neural networks, the resulting distributed representations of words contain surprisingly a lot of syntactic and semantic information. There are multiple degrees of similarity among words:

- KING is similar to QUEEN as MAN is similar to WOMAN
- KING is similar to KINGS as MAN is similar to MEN

Simple vector operations (addition and subtraction) with the word vectors provide very intuitive results, $vector(KING) - vector(MAN) + vector(WOMAN) \approx vector(QUEEN)$ [8].

Word vectors from neural networks were previously criticized for their poor performance on rare words – performance of skip-gram on rare words has good results for the nearest neighbours rare words which are more correlated with human ratings [9]. For the reasons and properties described above we choose this model for our experiments, further we work only with skip-gram.

2.3 Our Architecture

The idea was to replace the word vectors (sometimes called as projections or feature vectors) which computes the recurrent neural network itself with better ones computed on more data, even out-of-domain. As it was described in Section 2.2, the skip-gram provides us – in some point of view – better word vectors. Moreover, the vectors form skip-gram are computationally cheaper. The LSTM neural net architecture provides very stable learning (we have already done some experiments with modifications of the input vector in [10]). The LSTM network should not have any difficulties with learning from this skip-gram word vectors.

The word vectors obtained by training skip-gram model are trained on bigger corpora and afterwards they are used as an input of LM. It means that on the input of the LSTM language model we do not put words in 1-of-N coding, but words represented by skip-gram word vectors. This architecture should allow to language model to discover more regularities in a language.

3 Experiments

We performed the experiments to evaluate the contribution of our model to the current state-of-the-art were performed on Penn Treebank and Wikipedia corpus. Continuous distributed representations of words were obtained with *word2vec* toolkit [1], training of LSTM model has been provided with our own toolkit.

3.1 Text Data

To maintain comparability with the other experiments, we chose well-known Penn Treebank (PTB) [11] portion of the Wall Street Journal corpus for testing our models. The following standard preprocessing was applied to the corpora: the vocabulary was

[1] https://code.google.com/p/word2vec/

short-listed to 10k most frequent words, all numbers were unified into N tag and a punctuation was removed. The corpus was divided into 3 parts (training, development and test) with 42k, 3.3k and 3.7k tokens.

Continuous distributed representations of words need to be computed on big corpora, so we decided for free and easy available Wikipedia corpus (Wiki) [12]. In our experiments we used database dump from the year 2009, which has been preprocessed and the vocabulary was limited to 225k words (words that occurs more than 50 times), it contains 930M tokens. This data could be assumed as out-domain data if referred to PTB.

As the in-domain data we have used Wall Street Journal (WSJ) corpus [13] with 38M tokens.

3.2 Evaluated Models

All LSTM models used in our experiments were with 100 cells in the hidden layer. The input vocabulary was short-listed to 10k words in case of PTB, to 230k in case of PTB+Wiki; the output vocabulary is short-listed to 10k words from PTB.

Models denoted as **KN5** is a standard 5-gram language model with Knesser-Ney discounting [14], the **LSTM-100** means conventional LSTM neural network models with 100 cells in the hidden layer and **LSTM-100&skip1200** means our model fusion where word vectors of the width 1200 are on input of LSTM network.

For network training we used the PTB-development dataset as validation data and as test data is always used the PTB-test set.

3.3 Results

First, we evaluated how the amount of in-domain data impacts the resulting perplexity. This was done by adding increasing part of WSJ portion (by step of 10%) to the skip-gram model. The word vectors by *skip-gram* were computed from combined PTB and portion of WSJ corpus, the LSTM model was trained on PTB-train. As it could be seen in Figure 3, the model performs better with bigger amount of data for continuous distributed representations of words. But for word vectors computed only on PTB (for *portion of WSJ = 0* in the graph), model performs worse (perplexity 211) then KN5 baseline (perplexity 141.4, ($ID = 4$) in Table 1).

Figure 4 shows the influence of the word vector width on the results, the best results we achieved with the width 1200. The skip-gram word vectors were computed from combined PTB and Wiki corpus, the LSTM model was trained on PTB-train.

Table 1 shows the results for various models, all tested on PTB-test. It could be seen, that Wiki corpus is, as we expected, out-of-domain for PTB-test ($ID = 2$). The performance of LSTM-100 and KN5 are comparable, when trained on PTB, resulting perplexities are 145.0 and 141.4 ($ID = 3$ and $ID = 4$). As the baseline model we have chosen linear combination of KN5 models from PTB-train and Wiki corpus (tuned with EM-algorithm on PTB-dev data) with perplexity 139.5 ($ID = 5$). Our proposed model, even with word vectors trained on out-of-domain Wiki corpus, notable outperforms the baseline models with perplexity 130.4 ($ID = 6$).

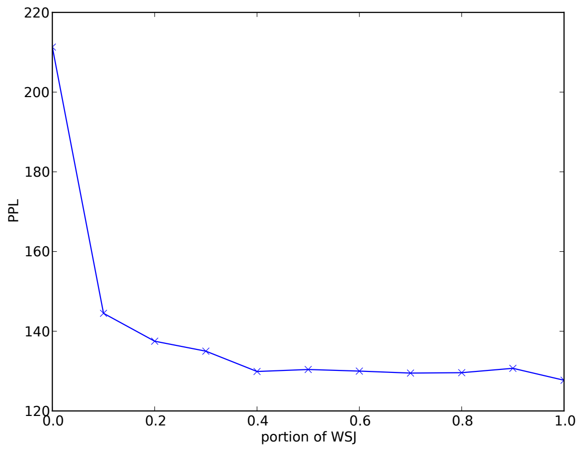

Fig. 3. Perplexity results with an increasing portion of WSJ added to PTB for training word2vec, measured on PTB-test.

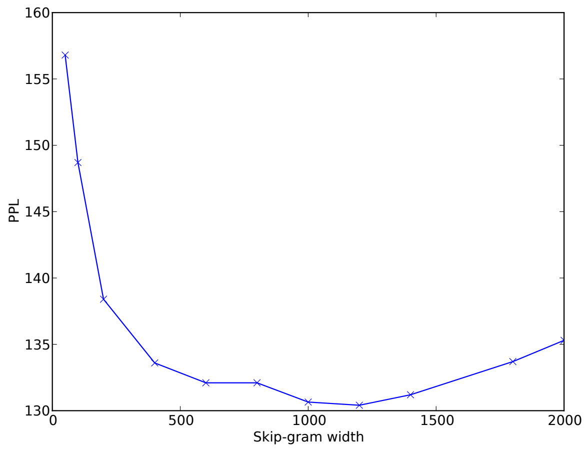

Fig. 4. Perplexity results with a various width of word vectors computed on PTB+Wiki corpus, measured on PTB-test.

4 Conclusions

In this paper, we have explored a novel language modelling technique where the long-short term memory neural network language model was combined with the skip-gram method of training of continuous distributed word representations.

Table 1. Perplexity results with various models, measured on PTB-test.

ID	Model Type	Train data			PPL
		***n*-gram**	**LSTM**	***word2vec***	
1	**LSTM-100**	-	PTB-train+ Wiki	-	308.0
2	**KN5**	Wiki	-	-	188.0
3	**LSTM-100**	-	PTB-train	-	145.0
4	**KN5**	PTB-train	-	-	141.4
5	**KN5 (lin.combination)**	PTB-train + Wiki	-	-	139.5
6	**LSTM-100&skip-1200**	-	PTB-train	PTB-train + Wiki	**130.4**

We empirically evaluated the proposed design against the conventional LSTM model with perplexity on Penn Treebank corpus. The experiments revealed that the combination of LSTM and skip-gram word vectors trained on huge out-of-domain data from Wikipedia notable outperforms baseline models.

The advantages of the LSTM&skip solution are that we are able to improve performance of a neural network model using a words vectors trained on an out-domain text. Secondly, the input layer of LSTM language model is fixed and allows eventually adding a new unseen word (for LSTM model) to the input, if we are able to compute its continuous distributed representation. However, the output vocabulary is fixed as in the conventional model. Another difficulty is that the learning must be done in two phases.

These model properties should be considered in our next experiments, which we would like to lead to even more general language model, with not fixed input and output dictionary and with an ability to learn new words and connections.

Acknowledgements

This research was supported by the Ministry of Culture Czech Republic, project No.DF12P01OVV022.

References

1. Mikolov, T., Kombrink, S., Deoras, A., Burget, L. and Černocký, J.: RNNLM - Recurrent Neural Network Language Modeling Toolkit. 2011.
2. Mikolov, T., Chen, K., Corrado, G. and Dean, J.: Efficient Estimation of Word Representations in Vector Space, in Proceedings of Workshop at ICLR, 2013.
3. Sundermeyer, M., Schlüter, R., Ney, H.: LSTM Neural Networks for Language Modeling, INTERSPEECH 2012.
4. Hochreiter, S., Schmidhuber, J.: Long Short-term Memory, in Neural Computation 9(8), pages 1735–1780, 1997.
5. Graves, A.: Generating sequences with recurrent neural networks., arXiv:1308.0850 [cs.NE]. 2013.
6. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. 3 (March 2003) 1137–1155
7. Mnih, A., Hinton, G.: Three new graphical models for statistical language modelling. In: Proceedings of the 24th International Conference on Machine Learning. ICML '07, New York, NY, USA, ACM (2007) 641–648
8. Mikolov, T., Yih, W. and Zweig, G.: Linguistic Regularities in Continuous Space Word Representations, in Proceedings of NAACL HLT, 2013.
9. Mikolov, T., Sutskever, I., Chen, K., Corrado G., and Dean, J.: Distributed Representations of Words and Phrases and their Compositionality, in Proceedings of NIPS, 2013.
10. Soutner, D. and Müller, L. : Application of LSTM Neural Networks in Language Modelling. Lecture Notes in Computer Science, p. 105-112, 2013.
11. Charniak, E. et. al. BLLIP 1987-89 WSJ Corpus Release 1, Linguistic Data Consortium, Philadelphia, 2000.
12. Wikimedia Foundation: Wikipedia, The Free Encyclopedia, `http://en.wikipedia.org`, 2009.
13. Garofalo, J. et. al.: CSR-I (WSJ0) Complete, Linguistic Data Consortium, Philadelphia. 2007.
14. Kneser, R. and Ney, H.: Improved backing-off for M-gram language modeling. Acoustics, Speech, and Signal Processing, 1:181, may 1995.