On Continuous Space Word Representations as Input of LSTM Language Model

Daniel Soutner and Luděk Müller

NTIS - New Technologies for the Information Society, Faculty of Applied Science, University of West Bohemia, Czech rep.
{dsoutner, muller}@ntis.zcu.cz

Abstract. 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 an efficient architecture. The continuous *skip-gram* and the *continuous bag of words* (CBOW) are algorithms for learning quality distributed vector representations that are able to capture a large number of syntactic and semantic word relationships. In this paper, we carried out experiments with a combination of these powerful models: the continuous representations of words trained with *skip-gram/CBOW/GloVe* method, word cache expressed as a vector using *latent Dirichlet allocation* (LDA). These all are used on the input of LSTM network instead of *1-of-N* coding traditionally used in language models. The proposed models are tested on Penn Treebank and MALACH corpus.

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

1 Introduction

In last years, recurrent neural networks (RNN) have attracted attention among other types of language models (LM) caused by their better performance [5] and their ability to learn on a smaller corpus than conventional n-gram models. Nowadays, they are considered as a state-of-the-art, especially the Long-short Term Memory (LSTM) variant. Skip-gram and continuous bag of words (CBOW) [6] 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. In this paper, we propose using these vectors on the input of LSTM language model and we observe an effectivity of this model.

There is described our proposed language model architecture in Section 2, including the description of sub-parts of the model such as continuously distributed representations of words. Section 3 deals with experiments and results and Section 4 summarizes the results and draws conclusions.

2 Model Architecture

2.1 Recurrent Neural Network Language Model

In standard back-off n-gram language models, words are represented in a discrete space – in the vocabulary. This prevents better interpolation of the probabilities of unseen n-grams because a change in this word space can result in an arbitrary change of the n-gram probability. The basic architecture of neural network model was proposed by Y. Bengio in [1]. The main idea is to understand a word not as a separate entity with no specific relation to other words, but to see words as points in a finite dimensional (separable) metric space. The recurrent neural networks – using the similar principle – were successfully introduced to the field of language modelling by T. Mikolov [5] and have become widely used language modelling technique.

In our work, we aimed to discover whether we are able to step further and move from words that are projected into the continuous space by network itself to the words projected to vectors with some more powerful techniques (as it is shown in [7]) and to learn the RNN model on this vectors afterwards.

2.2 Skip-gram and CBOW

Mikolov et al. in [6] introduced a new efficient architecture for training distributed word representations which belongs to the class of methods called "neural language models". Authors proposed two architectures: continuous skip-gram that tries to predict the context words given the input word and continuous bag-of-words (CBOW) that predicts the current word given the context (Figure 1). 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. We used publicly available *word2vec*[1] tool in our experiments.

2.3 Log-bilinear Variant of Skip-gram and CBOW

The log-bilinear variant (LBL) of both previously described architectures (CBOW-LBL and skip-gram-LBL) learned by noise-contrastive estimation (more could be found in [9]) seemed to perform slightly better on word analogy tasks. Thus, we decided to compare word vectors obtained with these architectures with the previous ones. We used the publicly available *LBL4word2vec* tool in experiments.[2]

2.4 GloVe (Global Vectors for Word Representation)

In essence, GloVe (Global Vectors for Word Representation) [10] is a log-bilinear model with a weighted least-squares objective. The main intuition which underlies this model is the simple observation that ratios of word-word co-occurrence

[1] code.google.com/p/word2vec
[2] github.com/qunluo/LBL4word2vec

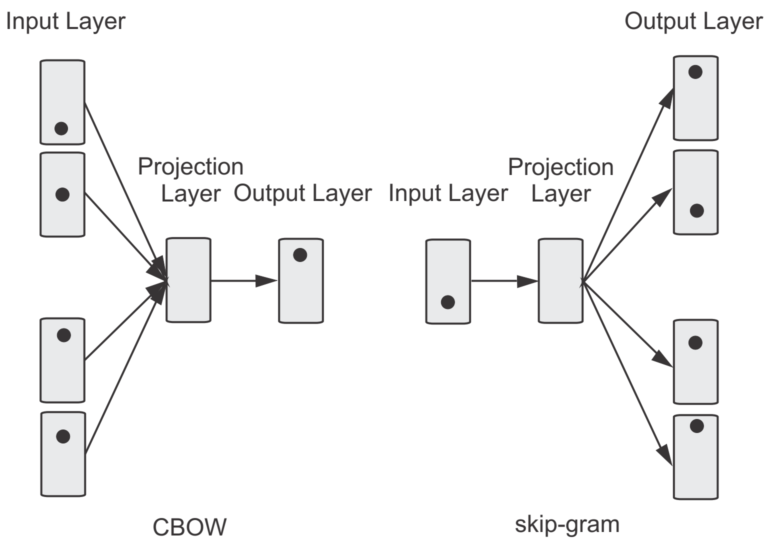

Fig. 1. The scheme of CBOW and skip-gram architecture. CBOW predicts the current word given the context and skip-gram that predicts the context words given the input word.

probabilities have the potential for encoding some form of meaning. The training criterion of GloVe is to estimate word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. Owing to the fact that the logarithm of a ratio equals the difference of logarithms, this objective associates (the logarithm of) ratios of co-occurrence probabilities with vector differences in the word vector space. Because these ratios can encode some form of meaning, this information gets encoded to the vector differences as well. Details about this architecture and implementation are published in [10]. We used the publicly available *GloVe* tool in our experiments.[3]

2.5 LDA

In addition to the word embeddings – to exploit more information from the long span context – we decided to use the latent Dirichlet allocation (LDA) [2] in our experiments as we already did in our previous work [13].

The LDA process converts word representation of a document to a low-dimensional vector which represents a probability of the topic. It represents documents as mixtures of topics that split out words with certain probabilities. In our experiments, we fixed the length of the word cache while computing the topic distribution.

[3] nlp.stanford.edu/projects/glove

We divided text to documents of 10 non-overlapping sentences for Penn Treebank corpus. For MALACH corpus, we stacked sentences to documents so that they are not longer than 200 words. The input vector of NN is modified as word embedding extended with this additional LDA feature computed from cache with the length of 50 words. The models were created with *gensim* tool [12]. We explored several configurations of trained models with a different number of topics.

2.6 Our Model Architecture

The motivation for our model and the background of the work was already lightly sketched out in Section 2.1. The main idea was to replace the word projections, which are computed by the recurrent neural network itself, with better ones. By the training RNN language model, the network learns own word embeddings, but if we look closer to they are estimated only from the past word co-occurrences (which comes from recurrentness of the network).

While all previously described models (in Sections 2.2, 2.3 and 2.4) produce better word vectors – in some point of view and measured in some applications. We assume, that this is caused by their architecture since they are all taking into account both words in past and words in the future context. This, in our opinion, makes the resulting vectors more accurate. The projections perform better while they are trained on more text or trained in more iterations, thus we run experiments with a various number of iterations over the training text.

We chose for our experiments the Long-short Term Memory (LSTM) [4] networks architecture due its better ability of learning [14].The scheme of one LSTM cell is shown in Figure 2. The simpler “vanilla” RNN network showed to be unstable while we were changing the input of the model in experiments.

To be able to catch the longer context, which is also crucial for accurate language model, we added LDA context features. This architecture, as we expect, should allow to discover more regularities in a language.

More technically, first the word vectors are computed for every word in vocabulary from training text (by different techniques mentioned above). Afterwards, in the training phase of the language model we give to the input of LSTM network these vectors instead of 1-of-N encoding of words. The input vector could be moreover extend simply by appending with the LDA vector – computed for every context.

3 Experiments and Results

In this section, we describe experiments which we did with proposed models and presenting the obtained results. The data are described in first part, the details about the training and the results follow.

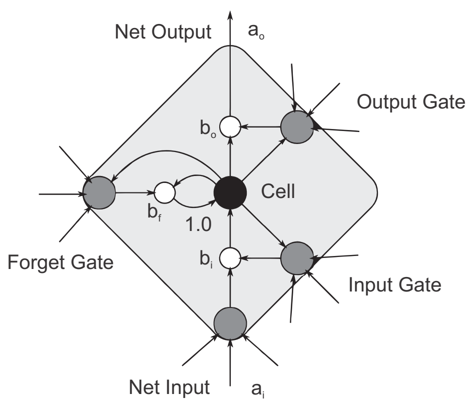

Fig. 2. The scheme of LSTM cell.

3.1 Data

Penn Treebank To maintain comparability with the other experiments in literature, we chose the well-known and widely used Penn Treebank (PTB) [3] portion of the Wall Street Journal corpus for testing our models. This is quite rare in the language modelling field and allows us to compare direct performance of different techniques and their combinations. Following common preprocessing was applied to the corpora:

- words outside the 10K vocabulary were mapped to a special token (unknown word)
- all numbers were unified into $\langle N \rangle$ tag
- punctuation was removed.

The corpus was divided into tree parts: sections 0-20 were used as the training data, sections 21-22 as the validation data and sections 23-24 as the test data.

MALACH We also wanted to carry out results on some real-world problem, so we decided to evaluate our models on the MALACH corpus [11]. Steven Spielberg (inspired by his experience making *Schindlers List*) established the Survivors of the Shoah Visual History Foundation (1994) to gather video testimonies from survivors and other witnesses of the Holocaust. It contains approximately 375 hours of interviews with 784 interviewees along with transcripts and other documentation. The original release includes transcripts of the first 15 minutes of each interview, which makes in a textual form circa 2M tokens, the vocabulary consists of 21.7k words. We split this data to *train* (70%), *development* and *test* folds (13% and 17%).

3.2 Training

For obtaining word embeddings, we used context window size of 10 words and the other parameters we anchored to default settings because tuning more parameters would be exhausting. For the language model – as remarked above – we used LSTM network, to speed up experiments the simplified version. During the training phase, the gradients were clipped into the interval $< -1, 1 >$; starting with learning rate at $\alpha = 0.16$ and while not achieving perplexity decrease we halved learning rate. The width of a hidden layer is fixed to 100 neurons.

3.3 Results

First, we evaluated, which word embeddings are suitable for this task; we employed algorithms described above. The performance of word embeddings also strongly depends on a number of training iterations, hence we produced results with various number of them – they are shown in Figure 3. If we employ the LDA

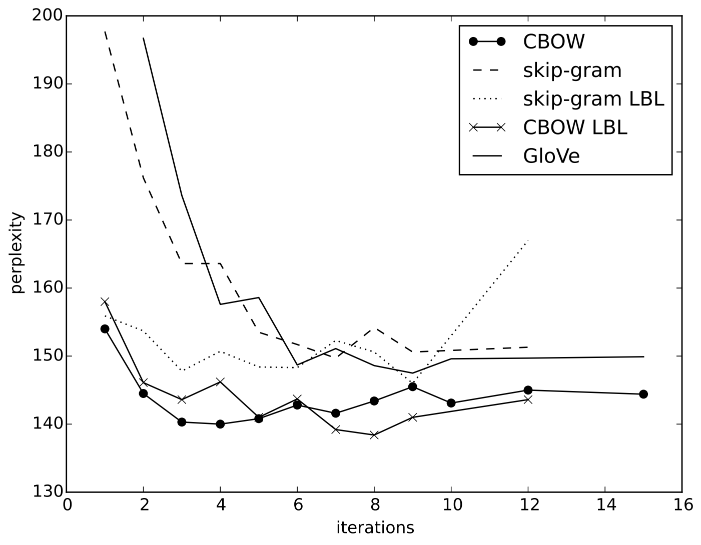

Fig. 3. Various types of word vectors, performance measured on PTB while using them in language model

extension, we obtain a bit better results, as we supposed. The best results on PTB are shown in Table 1.

The results for the MALACH task are in Table 2, where we compared our model (LSTM-100&CBOW LBL-100) with RNN models (RNN-100 and RNN-400 with 100 respectively 400 neurons in hidden layer). The CBOW LBL embeddings were used as the best option from experiments before. The RNN models

Model	PPL
LSTM-100	144.0
LSTM-100&CBOW LBL-100	138.4
LSTM-100&CBOW LBL-100 + LDA-50	133.6

Table 1. Perplexity results on Penn Treebank corpus.

were trained with RNNLM Toolkit [4], number denotes a size of a hidden layer. We also added results achieved with a linear combination of neural models with more conventional models Knesser-Ney 5-gram (KN5) and maximum entropy on 5-gram features (ME5). For the completeness the result with LDA (with 50) extension is added and also the model mixed together with n-gram models (with linear interpolation, which coefficients were tuned on development data with EM algorithm).

Model	PPL
RNN-100	107
RNN-400	100
LSTM-100	99
RNN-100 + KN5 + ME5	94
LSTM-100&CBOW LBL-100	**94**
LSTM-100&CBOW LBL-100 + LDA-50	91
LSTM-100&CBOW LBL-100 + LDA-50 + KN5 + ME5	83

Table 2. Perplexity results on MALACH corpus.

4 Conclusion and Future Work

We proposed language model using continuous word representation as input and extended this input with information from context vectors. We employed techniques that are assumed as a state-of-the-art such as LDA, skip gram and a log-bilinear continuous bag of words. The experiments showed the improvement of 4-5% in perplexity over standard LSTM/RNN models measured on Penn Treebank and MALACH corpus and with LDA extension circa 7%-8%.

We shown, that using continuous word vectors computed outside of neural network could improve LSTM-LM performance and even more if we add additional context information. Nevertheless it is good to notice, that the type and the quality matter i.e. number of train iterations of word vectors and algorithm. In our next work, we would like to further verify our approach on other corpora

[4] rnnlm.org

(such as Wikipedia Text8 or "One billion word benchmark") and on the speech recognition or automatic translation.

Acknowledgements

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

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

References

1. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. 3 (March 2003) 1137–1155
2. Blei, D. M., Ng, Y. A.,Jordan, M. I. and Lafferty, J.: Latent dirichlet allocation, Journal of Machine Learning Research, 2003, vol.3.
3. Charniak, E. et. al. BLLIP 1987-89 WSJ Corpus Release 1, Linguistic Data Consortium, Philadelphia, 2000.
4. Hochreiter, S., Schmidhuber, J.: Long Short-term Memory, in Neural Computation 9(8), pages 1735–1780, 1997.
5. Mikolov, T., Kombrink, S., Deoras, A., Burget, L. and Černocký, J.: RNNLM - Recurrent Neural Network Language Modeling Toolkit. 2011.
6. Mikolov, T., Chen, K., Corrado, G. and Dean, J.: Efficient Estimation of Word Representations in Vector Space, in Proceedings of Workshop at ICLR, 2013.
7. 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.
8. Mikolov, T., Yih, W. and Zweig, G.: Linguistic Regularities in Continuous Space Word Representations, in Proceedings of NAACL HLT, 2013.
9. Mnih, A. and Kavukcuoglu, K.: Learning word embeddings efficiently with noise-contrastive estimation, Advances in Neural Information Processing Systems 26 (NIPS 2013).
10. Pennington, J., Socher, J. and Manning., C. D.: GloVe: Global Vectors for Word Representation. Empricial Methods in Natural Language Processing (EMNLP), 2014.
11. Ramabhadran, Bhuvana, et al. USC-SFI MALACH Interviews and Transcripts English LDC2012S05. Web Download. Philadelphia: Linguistic Data Consortium, 2012.
12. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora, in Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks. Valletta, Malta: University of Malta, 2010. p. 4650, 5 pp. ISBN 2-9517408-6-7.
13. Soutner, D. and Müller, L. : Application of LSTM Neural Networks in Language Modelling. Lecture Notes in Computer Science, p. 105-112, 2013.
14. Sundermeyer, M., Schlüter, R., Ney, H.: LSTM Neural Networks for Language Modeling, INTERSPEECH 2012.