Outperforming Neural Readers Using Broad-Context Discriminative Language Modeling on the LAMBADA Word Prediction Task

Nima Nabizadeh, Mittul Singh, Dietrich Klakow
Spoken Language Systems, Saarland University,
Saarland Informatics Campus, Saarbrücken, Germany
{firstname.lastname}@lsv.uni-saarland.de*

Abstract

Since Hinton and Salakhutdinov published their landmark science paper in 2006 ending the previous neural-network winter, research in neural networks has increased dramatically. Researchers have applied neural networks seemingly successfully to various topics in the field of computer science. However, there is a risk that we overlook other methods. Therefore, we take a recent end-to-end neural-network-based work (Dhingra et al., 2018) as a starting point and contrast this work with more classical techniques. This prior work focuses on the LAMBADA word prediction task, where broad context is used to predict the last word of a sentence. It is often assumed that neural networks are good at such tasks where feature extraction is important. We show that with simpler syntactic and semantic features (e.g. Across Sentence Boundary (ASB) *N*-grams) a state-of-the-art neural network can be outperformed. Our discriminative language-model-based approach improves the word prediction accuracy from 55.6% to 58.9% on the LAMBADA task. As a next step, we plan to extend this work to other language modeling tasks.

1 Introduction

Neural networks (NN) have endured a long winter and had their spring a decade ago. As they are in full bloom, the community seems to suffer from an NN monoculture. Researchers spend tremendous effort in training NNs including intense hyperparameter tuning. However, we forget that classical methods like feature-based techniques also have their merits. We therefore want to take a recent NN architecture and compare it with a feature-based approach.

(Dhingra et al., 2018) explore neural readers on the LAMBADA word prediction task (Paperno et al., 2016) and establish the state-of-the-art. LAMBADA dataset is built from manually-chosen instances from the BookCorpus (Zhu et al., 2015), where the task is to predict the last word (*target*) of the last sentence in a paragraph and it has been shown that without the broad context humans are unable to predict the target. On this task, neural readers (Chu et al., 2017; Dhingra et al., 2017; Dhingra et al., 2018) have substantially outperformed recurrent, memory-based and other popular language models, which have close-to-zero accuracy on the same task.

In contrast, we develop simpler syntactic and semantic features based on both non-neural and neural methods used in a Discriminative Language Model (DLM) for this task. Specifically, **1)** We introduce features based on ASB word dependencies, on a model to learn the possible last words in a sentence and on similarity of word candidates to important words in the context in Section 2; **2)** we apply a DLM-based approach (Section 3) for word prediction and compare it with the neural reader approach; and **3)** we analyze the effect of different feature sets and present state-of-the-art results on the LAMBADA task (Section 4).

2 Features for the Word Prediction

In this section, we describe the various word features used in the DLM. We train all these features using the BookCorpus training set (the corpus without the LAMBADA instances).

The Across Sentence Boundary Model: The LAMBADA task relies mainly on capturing the long-term information from the paragraph (Paperno et al., 2016). To incorporate such informa-

*The work was supported by the Cluster of Excellence for Multimodal Computing and Interaction and the German Research Foundation (DFG) as part of SFB1102.

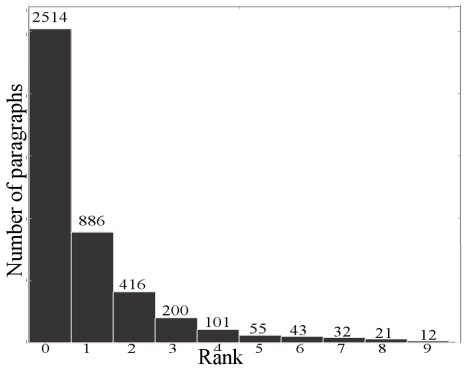

Figure 1: Frequency of correct words at different ranks in the LAMBADA development set

tion, we use the Across Sentence Boundary LM (ASB) (Momtazi et al., 2010), which captures the information of the words in the previous sentences $(s_{-1}, s_{-2}, \ldots)$, triggering the words in the current sentence (s_0) as follows:

$$p(W_{s_0}|W_{s_{-1}}) = \frac{N(W_{s_{-1}}, W_{s_0})}{N(W_{s_{-1}})} \quad (1)$$

where W_{s_0} and $W_{s_{-1}}$ are the words in the current and the previous sentence respectively, and N counts such occurrences. Finally, the score generated from a log-linear interpolation of the unigram probability for W_{s_0} and the trigger probability for all words in the paragraph is used as a feature.

We also created such features by skipping one or more previous sentences (denoted as ASB-*d*, where d is number of skipped sentences). Moreover, we constructed ASB-*d* trigrams, but these LMs showed improvements for d = 1 only.

Similarity to Important words: The last word is semantically similar to one or more words in the paragraph. By identifying these words, we can better predict the last word. For simplicity, we assume that these semantically-similar words are contextually *important* words.

To rank the words based on their importance, we apply the method proposed in (Zhou and Slater, 2003), where the important or relevant words are assumed to appear in bursts i.e. close to each other in text. On the other hand, less relevant words occur randomly everywhere in text, therefore they do not form significant clusters. Unlike the co-occurrence counts-based methods, such as tf-idf, this method ranks the words on the variance of distances between their occurrences in the corpus, represented by $\Gamma(w)$ (Ventura and da Silva, 2008).

In our experiments, we observed that the method worked better by adding the total number of words (n) to each value and subtracting the unigram frequency (m) of the word candidate (w), in effect smoothing the scoring mechanism:

$$\Gamma^*(w) = \Gamma(w) + n - m \quad (2)$$

Based on these scores, we select the N most interesting words in the paragraph and pick the M most similar words for each such word in the vocabulary. We fixed the values of N and M to 50 and 200 respectively. To define the similarity between words, we use cosine similarity based on word embeddings (Mikolov et al., 2013) trained on the BookCorpus training set. The words collected in this manner are assigned with a score (denoted as **Sim2Imp**) that is the multiplication of its cosine similarity with respect to the important word (w) and importance of the word $(\Gamma^*(w))$.

Last Word Prediction: Conventional LMs predict words at every time step of the sentence, however, in the LAMBADA task we only have to predict the last word of the sentence. Based on this observation, we build a model to predict the probability of a word being the last word (labeled as 1) or not (labeled as 0). We use Long Short Term Memory-based (LSTM) neural network (Hochreiter and Schmidhuber, 1997) to label the word (w_i) from a sentence $(s = \{w_1, w_2, \ldots, w_n\})$, minimizing a weighted mean squared error between correct label and predicted label (L_i^p, also used as a feature) across all sentences (S) in the corpus:

$$\sum_{s \in S} \left\{ \alpha(1 - L_n^p)^2 + \sum_{w_i \in \{w \in s | w \neq w_n\}} (L_i^p)^2 \right\} \quad (3)$$

where L_n^p is the predicted label of last word (w_n).

We weigh the loss due to the last words more than the other words for this unbalanced classification problem. α is chosen as the average number of words per sentence in the training set. The LSTM model has two layers of recurrent cells with 300 neurons in each.

Paragraph embedding-based Feature: Aforementioned features focus on capturing explicit interactions between the last word and the paragraph words, lacking any topical interactions.

Thus, we compute the similarity between the vector representation of the paragraph and word candidates (denoted as **Sim2Para**). This feature can help capture the topical similarity of the words and the paragraph. To compute these paragraph embeddings, we utilize the distributed memory model (Le and Mikolov, 2014), which has been shown to

provide a well performing set of paragraph embeddings (Dai et al., 2015).

Recurrent Neural Network based LM: Recurrent Neural Network LMs (RNNLM) (Mikolov et al., 2010) cannot model short-term dependencies as well as N-grams. Moreover, (Le et al., 2012; Oualil et al., 2016) have shown that in RNNLMs the context changes rapidly over time and hence, they are not suitable for capturing longer correlations. Nevertheless, these models can be used to capture *mid-term dependencies* and we use their output probability as a feature.

We train an RNNLM with a layer of 300 hidden nodes and use this model to score words from the vocabulary given the sentence as the context.

***N*-grams Features**: We augment the above-described mid- and long-term features with local or short-term information from n-gram counts of the last word candidates as a feature. Previously, combinations with n-grams as local context have considerably improved the language modeling performance (Mikolov et al., 2011; Chelba et al., 2014; Józefowicz et al., 2016).

These counts for ***N*-gram** (N = 2, 3, 4, 5) are extracted from the BookCorpus training set. Additionally, we also use ***Google N*-grams** (Brants and Franz, 2006) counts for N = 2 and 3 .

Other Features: We define a few other features based on the characteristics of the LAMBADA dataset. As 83% of the last words appear in the given broader context (Paperno et al., 2016), we assign words from the context a high weight (denoted as **InContext**).

We also consider a position-based feature (**Position**) that is the distance between the word candidate in a paragraph and the last sentence. This distance is also scaled by the target word's frequency that appears at the same distance in development set.

On the LAMBADA set, a preference for certain POS tags as the last words has been noted earlier (Paperno et al., 2016). Thus, we weigh word candidates based on their **POS** tag. The POS tag distribution is estimated on the development set and a word is weighed with a score proportional to their POS tag probability mass.

3 Word Prediction Pipeline

The word prediction is set up in two stages. In the first stage, we *select word candidates* and in the second stage, we *learn a classifier* to predict the target word, shown in Figure 2.

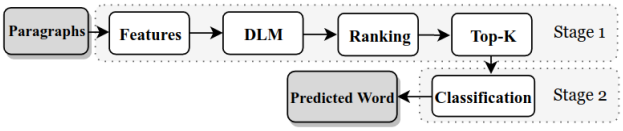

Figure 2: The two stage pipeline used for word prediction on the LAMBADA dataset

To select the word candidates, we first use features extracted in Section 2 to train a DLM. The DLM optimizes the word prediction accuracy by creating a linear model using the features. The DLM parameters are optimized on the LAMBADA development set. For further details about DLMs, we direct readers to a comprehensive review of DLM techniques (Saraclar et al., 2015).

By applying the DLM, we obtain a ranked list of word candidates for the last position of the paragraph. Out of this ranked list, we select the top-K words, which are then passed to the next stage. The ranks of a target word for development set paragraphs are plotted in Figure 1 and we observe that 83% target words occur within the first 10 words. Also, setting $K = 10$ the word prediction pipeline performed best on the development set.

Choosing from top-K candidates is set up as a binary classification task over the features, where the classifier predicts if a word candidate is the target word or not. This classification task is unbalanced due to larger non-target samples to target samples ratio per paragraph and is compensated by penalizing the misclassification of a correct target word. The training is performed using a Multi-Layer Perceptron (MLP) with two hidden layers of size 24 and 12. In our experiments, we also tried using Support Vector Machines for this classification task, but MLP obtained the best results.

4 Experiments and Results

In this section, we evaluate our DLM-based word prediction pipeline against the present state-of-the-art techniques on the LAMBADA test set.

4.1 Stage 1: Word Candidate Selection

DLM results for first stage are reported in the second column of Table 1. Note, all results reported in this table use the InContext feature.

Earlier in (Paperno et al., 2016), RNNLM and N-gram LMs perform with zero accuracy on the LAMBADA task, but their accuracy improves when used

Models	Accuracy	
GA reader w/ features (Chu et al., 2017)	49.0	
GA reader w/ MAGE (Dhingara et al., 2017)	51.6	
GA reader w/ C-GRU (Dhingara et al., 2018)	55.6	
DLM	Stage 1	Stage 2
All Features	48.8	**58.9**
RNNLM	4.5	4.5
N-grams (N = 2, 3, 4, 5)	16.8	18.1
ASB	34.8	36.0
- RNNLM	45.3	55.6
- Sim2Para	45.1	55.4
- Last Word	45.4	55.6
- Sim2Imp	45.6	55.9
- *Google N*-grams	45.9	56.2
- Position & POS	46.3	57.6
- *N*-grams (*N* = 2, 3, 4, 5)	36.8	42.7
- ASB-d *2*-gram & *3*-gram	22.3	28.2

Table 1: Word prediction accuracy on the LAMBADA test set, plus previous benchmarks. GA stands for Gated Attention Neural Network, MAGE stands for Memory as Acyclic Graph Encoding and C-GRU stands for Coref-GRU

alongside with InContext in the DLM as reported in rows 4 and 5 of Table 1. Using all the ASB features (ASB includes ASB-d 2-grams for d = 1 to 5 and ASB-1 3-grams) significantly improves performance compared to RNNLM and *N*-gram features by leveraging more information. Combining all the features (labelled as **All Features**), we get the best results for the first stage.

The subsequent rows report DLM's performance when the feature(s) in the row are not used to infer the target word. Singleton features when removed lead to substantial losses in the accuracy. Again, ASB-based features are noted as the most important feature set, as removing this group leads to the largest loss in accuracy.

4.2 Stage 2: Target Word Prediction

Learning to re-rank in this stage helps improve the overall accuracy by approximately 10%. Choosing from only top 10 words reduces data sparsity and the second stage classifier helps improve the performance in this less-sparse space.

Similar trends as the first stage are observed with respect to different features in the second stage. Overall, we are able to outperform the state-of-the-art system (Dhingra et al., 2018), showing the benefits using a two-stage DLM.

5 Related work

Language modeling has become the happy hunting grounds for neural network LMs by performing substantially better than other methods (Józefowicz et al., 2016). Skip-gram-based features, similar to long-term dependency features used in this work, have been preferred over such neural networks due to their computational prowess and scalability to large datasets (Chelba et al., 2017; Pelemans et al., 2016; Chelba and Shazeer, 2015; Shazeer et al., 2015). But, such efforts have been concentrated and have slipped into a very *deep* winter elsewhere.

On the LAMBADA word prediction task, neural readers (Chu et al., 2017) have shown substantial performance gains against recurrent and memory-based neural network models. (Dhingra et al., 2018) have augmented this neural reader using external knowledge-based annotations to capture long-term information and further improved the performance on this task.

These neural readers are trained on automatically-extracted data that requires the paragraph to have certain length and contain the last word. This constraint restricts the amount of data leveraged to learn long-term triggering information, which is critical to this task. In contrast, the features used in our work are not restricted by such constraints and can learn triggering information over a larger data, which is larger by an order of magnitude to the neural reader's training set.

6 Concluding Remarks

Advancements in deep learning have shown impressive gains across a variety of research problems. These improvements are so dramatic that we forget to look at other methods. It is essential that we focus an equal amount of effort on non-neural methods to allow for a fair comparison between these approaches. The method presented in this paper is a step in this direction.

Here, we presented a DLM-based approach for the LAMBADA word prediction task, which is trained on various syntactic and semantic features. Unlike the neural readers, this feature-oriented method is easy to scrutinize and understand. Overall, we outperformed the previous best of 55.6% on the LAMBADA task to establish a new state-of-the-art of 58.9%. As a next step, we plan to extend our work to more classical language modeling tasks on the Penn Treebank and the one-billion-word corpora.

References

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram corpus version 1.1. *Google Inc*.

Ciprian Chelba and Noam Shazeer. 2015. Sparse non-negative matrix language modeling for geo-annotated query session data. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, 2015, Scottsdale, AZ, USA, December 13-17, 2015*, pages 8–14.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639.

Ciprian Chelba, Diamantino Caseiro, and Fadi Biadsy. 2017. Sparse non-negative matrix language modeling: Maximum entropy flexibility on the cheap. In *18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 2725–2729.

Zewei Chu, Hai Wang, Kevin Gimpel, and David McAllester. 2017. Broad context language modeling as reading comprehension. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 52–57. Association for Computational Linguistics.

Andrew M. Dai, Christopher Olah, and Quoc V. Le. 2015. Document embedding with paragraph vectors. *CoRR*, abs/1507.07998.

Bhuwan Dhingra, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Linguistic knowledge as memory for recurrent neural networks. *CoRR*, abs/1703.02620.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. 2:42–48.

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

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1188–1196.

Hai Son Le, Alexandre Allauzen, and François Yvon. 2012. Measuring the influence of long range dependencies with neural network language models. In *Proceedings of the Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT, WLM@NAACL-HLT 2012, Montrèal, Canada, June 8, 2012*, pages 1–10.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukás Burget, and Jan Cernocký. 2011. Empirical evaluation and combination of advanced language modeling techniques. In *12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*, pages 605–608.

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

Saeedeh Momtazi, Friedrich Faubel, and Dietrich Klakow. 2010. Within and across sentence boundary language model. In *11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1800–1803.

Youssef Oualil, Mittul Singh, Clayton Greenberg, and Dietrich Klakow. 2016. Long-short range context neural networks for language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1473–1481.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Joris Pelemans, Noam Shazeer, and Ciprian Chelba. 2016. Sparse non-negative matrix language modeling. *Transactions of the Association for Computational Linguistics*, 4:329–342.

Murat Saraclar, Erinc Dikici, and Ebru Arisoy. 2015. A decade of discriminative language modeling for automatic speech recognition. In *International Conference on Speech and Computer*, pages 11–22. Springer.

Noam Shazeer, Joris Pelemans, and Ciprian Chelba. 2015. Sparse non-negative matrix language modeling for skip-grams. In *16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 1428–1432.

Joao Ventura and Joaquim Ferreira da Silva. 2008. Ranking and extraction of relevant single words in text. In *Brain, Vision and AI*. InTech.

Hongding Zhou and Gary W Slater. 2003. A metric to search for relevant words. *Physica A: Statistical Mechanics and its Applications*, 329(1-2):309–327.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.