Topic-Enhanced High-Rank Language Model

Ao Liu **Yichi Zhang**
University of Massachusetts Amherst
College of Information and Computer Science
{aoliu, yiczhang}@cs.umass.edu
March 30th, 2018

Abstract

Using language models to help learning the word embeddings has dominated the natural language processing (NLP) community since first introduced by Bengio et al. (2003). Many recent researches focus on disambiguating word meanings by surrounding contexts (McCann et al., 2017; Peters et al., 2018). We assume that such contexts form some abstract topics of sentences, so that topic of a sentence can affect the word semantics. Thus, we tend to use topic model to enhance the language model to solve the problem of word ambiguity. Meanwhile, models based on conventional softmax decoding lack of expressiveness due to the softmax bottleneck, whereas Mixture of softmax (MoS) enables the probability matrix of a language model to have high-rank so as to increase the expressiveness (Yang et al., 2017). We attempt to adapt the idea of MoS to help us build a topic-enhanced high-rank language model.

1 Introduction

Statistical language modelling has evolved from traditional N-gram language models to neural language models over the past two decades (Bengio et al., 2003; Mikolov et al., 2010). Traditional N-gram approaches rely on Markov assumption, which usually has a high cost to learn long-distance dependency. The introduction of RNN-based language model relieves such burden by computing the conditional probability of a word given all previous words using hidden state(Mikolov et al., 2010). Despite the difference of model architectures, statistical language models factorize the probability of joint probability of a given sequence $S = (w_1, ..., w_N)$ into conditional probabilities: $P(S) = \prod_{t=1}^{N} P(w_t|w_{m:t})$, where $w_{m:t}$ is the sequence $w_m, w_{m+1}, ..., w_t$. Notice that when $m = t - N + 1$, the factorization is for N-gram models, whereas when $m = 1$, the conditional probability is based on all previous words.

Although RNN-based language models have higher generalizability by capturing long-distance information, they may in practice encounter overfitting issues (Srivastava et al., 2014). A general goal of language model is to capture both the syntax and the semantic coherence in the language. Syntactic structures mostly depend on local context, and semantic dependencies can have arbitrary long distance.

Probabilistic topic model is one way to learn such semantic coherence (Blei and Lafferty, 2009). A general goal of probabilistic topic models is to learn abstract topics formed by groups of words and to express documents as combinations of those topics. By doing so, such abstract topics are then embedded with semantic dependencies over the words and could potentially be utilized by language model to capture long-distance information without overfitting.

To further enlarge the expressiveness of our model, we incorporate MoS to our decoder. The reason is that MoS provides a high-rank probability matrix, whereas conventional softmax is low-rank and can only approximate the true distribution Yang et al. (2017).

2 Related Works

In this section, we only describe previous works that are most related to our approach and describe some different decisions we make.

2.1 TopicRNN (Dieng et al., 2016)

Dieng et al. (2016) introduce TopicRNN to provide a way of incorporating topic model into language model to learn better contextualized word embeddings. Instead of passing the learned topic representation of a sentence through the RNN as the hidden states, they separate global semantics from local syntax representations and directly add the topic representation to non-stop words. By doing so, they exclude the effects of stop words. We will try encrypting the topics using a softmax-based method, so that each word in a sentence may be related to different topics, because a complex sentence usually contains different topics.

2.2 Embeddings from Language Models (ELMo)(Peters et al., 2018)

Peters et al. (2018) propose a method to add softmax weights on the embeddings learned from bidirectional language model:

$$\mathbf{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j} \qquad (1)$$

where k is the position of the token, L is the number of layers, $\mathbf{h}$ is the hidden states, $\mathbf{s}^{task}$ are softmax-normalized weights and γ^{task} is the scalar parameter. The bidirectional language model can be unsupervisedly trained on large corpus. And then to adapt the learned word embeddings to a downstream task, they first fine-tune the word embeddings on the target dataset and then simultaneously train the task model parameters and the softmax parameters to give weights for word embeddings learned by different layer of the language model. Instead of fine tuning the model on supervised tasks to get task-specific word embeddings, we tend to let the model learn both syntactic and semantic contextual informations and learn the best contextualized word embeddings all by itself in an unsupervised manner without the help of labelled data.

2.3 Mixture of Softmax (Yang et al., 2017)

Yang et al. (2017) prove that to better model language, we should allow the probability matrix to have high rank in order to increase the expressiveness of a model. They suggest a technique called Mixture of Softmax has the form:

$$P(w_i|w_{1:t}) = \sum_{k=1}^{K} \pi_{w_{1:t},k} \frac{\exp(\mathbf{h}_{w_{1:t},k}^T)\mathbf{w}_t}{\sum_{t'=1}^{N} \exp(\mathbf{h}_{w_{1:t'},k}^T)\mathbf{w}'_t} \qquad (2)$$

, where $\pi_{w_{1:t},k}$ is the prior or mixture weights of the k-th component and $\mathbf{h}_{w_{1:t},k}$ is the k-th context vector associated with context $w_{1:t}$. By doing so, MoS improves the expressiveness over conventional softmax by allowing the probability matrix to be high-rank. They adapt MoS to AWD-LSTM by Merity et al. (2017) and achieves the state-of-the-art performance on several language modelling datasets. Their model does not incorporate much topical information and merely uses a latent discrete decision as the topic to predict the next word. To improve on that, we will use a topic model to provide abstract representation to enhance the topical information in the model.

3 Dataset

The two datasets listed below are the ones that we will train and test our proposed language model. Due to the purpose of language model, no annotations would be needed.

3.1 Penn Treebank (PTB)

The Penn Treebank (PTB) project selected 2,499 stories from a three year Wall Street Journal (WSJ) collection of 98,732 stories for syntactic annotation (Marcus et al., 1993). It is a standard benchmark for assessing new language models. We will use the standard split, where sections 0-20 (930K tokens) are used for training, sections 21-22 (74K tokens) for validation, and sections 23-24 (82K tokens) for testing.

3.2 WikiText-2

The WikiText language modeling dataset is a collection of over 100 million tokens extracted from the set of verified good and featured articles on Wikipedia (Merity et al., 2016). Compared to the preprocessed version of PTB, WikiText-2 (WT2) is over 2 times larger. The WikiText dataset also features a far larger vocabulary and retains the original case, punctuation and numbers - all of which are removed in PTB. As it is composed of full articles, the dataset is well suited for models that can take advantage of long term dependencies.

We will use the splits built in the dataset for training, development and testing respectively.

4 Software and Packages

Since we will implement a deep neural model, either Tensorflow or PyTorch will be used to help building the model architecture.

4.1 Tensorflow

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Googles AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

4.2 PyTorch

PyTorch is a Python package that provides two high-level features: 1) Tensor computation (like NumPy) with strong GPU acceleration and 2) Deep neural networks built on a tape-based autograd system.

5 Preliminary Experiment

To evaluate our model, we will compare the model perplexities over the two datasets we use. We will also test on the effectiveness of our topic-enhancing method and the generalizability of high-rank MoS-based decoding to our model.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

David M Blei and John D Lafferty. 2009. Topic models. *Text mining: classification, clustering, and applications*, 10(71):34.

Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2016. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2017. Breaking the softmax bottleneck: a high-rank rnn language model. *arXiv preprint arXiv:1711.03953*.