Language Model Enhanced Glove

Daniel Campos dacampos@uw.edu

06/10/2019

Abstract

Large Neural Language Models(NLM) have shown to be quite effective at nearly all tasks related to language such as question answering, sentiment analysis, search, etc. Despite the impressive model performance, most applications are unlikely to see a full Language Model used in production because they model speed and latency can be prohibitive. In this paper, we propose an approach for improving traditional word embedding by leveraging word cooccurrence probability from NLMs. These new NLM enhanced word embedding outperform traditional embedding on word similarity benchmarks and in downstream tasks NLM enhanced models can learn faster than traditional models.

1 Introduction

Text representation mechanism like word vectors(e.g. Mikolov et al. (2013)) and sentence vectors(e.g. Kiros et al. (2015)) have been known to provide a stable, efficient and accurate representation of language for use in Neural Networks and other traditional language models. More recently, contextualized word representations (Peters et al. (2018) and other Large Neural Language Models(e.g. Devlin et al. (2018) and Radford et al. (2019)) have shown to better represent text by leveraging the contextual information and continued expansion of training corpora. The aforementioned NLM's have become central to NLPs watershed moment because they have brought such improvements in accuracy in downstream task it becomes difficult to see why more traditional word representations would be useful. Despite the tremendous accuracy gains deploying these massive NLM can prove to be more difficult than traditional word representations and in many NLP tasks they can be overkill. Text representations such as Glove(Pennington et al. (2014)) have become the defacto norm in many natural language processing (NLP) tasks due to the ease of which they can be implemented in downstream tasks and the small size of the models which allows they to be deployed for inference in latency and speed sensitive environments. In this paper, we will describe how we leveraged some aspects of NLMs in order to enhance the more traditional Glove and by doing so saw a large improvement in word similarity benchmarks and saw faster convergence in downstream classifications tasks.

2 Enhancement Procedure

Traditional word embedding like Glove are trained by performing large scale factorization on a matrix of word cooccurrence. This method is very efficient at distilling large scale word properties but is highly dependent on the text corpus the model is trained upon. Since the model is trained off cooccurrence the training procedure is not able to focus on cases that are very common in text when compared to more rare cooccurrence. One can consider two essential steps as part of the Glove training procedure: create a word cooccurrence matrix and factorize that matrix to create our learned word embedding. In the first step, the corpus is read over and at each step a conceptual text window is created. A training hyper parameter is set to how big the context window with the goal to have all items in the windows be part of the larger conceptual meaning. The matrix is then updated for each words cooccurrence with the increment being dependent on how far away the words are from each other. The increment is set to 1 divided by the distance of the two words. This increment is set to provide more weight to words that co-occur closer together because they likely have more effect on each other than words far away. OpenAI's GPT-2(Radford et al. (2019) is a large NLM that is trained to predict the next word in a sequence given the proceeding N words. Thus, this model provides a method of calculating the $P(Y|X)$ where X is an existing context window and Y is the target word. In our experiments, we explored the effect of integrating millions of word cooccurrence probabilities into the word cooccurrence matrix and the various changes in quality of learned representations depending on how the probabilities are integrated. Our method is simple and is designed to be as fast as possible. Our NLM enhanced Glove has 3 stages. In the first stage, the text corpus is iterated over and much like in glove or word2vec word cooccurrence pairs are generated. Using these pairs, formulated as (x,y) the GPT-2 model is loaded and the $P(X|Y)$ and $P(Y|X)$ is calculated. This is the most time consuming step but only need to be done once on a sufficiently large and diverse corpus. In the second stage, much like Glove a cooccurrence matrix is generated. Unlike the traditional model, we modify the cooccurrence increment to be dependent on an activated version of $P(Y|X)$. There are various ways to activate the probabilities but as shown in our experiments, we found the most efficient to be $2^{e^{(P(Y|X)+P(X|Y))}}$. The third and final step is borrowed entirely from the original glove involves training word vectors by performing matrix factorization. By only having these minor deviations from the original Glove model we can quickly and efficiently generate new embedding and do so in almost any setting with no changes to other text processing infrastructure.

3 Experiments

In this section, we first introduce data sets and basic settings used to learn embedding. We then discuss additional settings and present experimental results of the two tasks we ran to explore how glove could be modified to make use of the conceptual information: word similarity and text classification.

3.1 Embedding Training

For our experiments we used the enwik8 (Hunter et al. (2012)) data set. The corpus was then cleaned of all web page like elements to represent 200,000 unique tokens and being 12,577,299 words long. For our training we set the learning rate to 0.05, models trained for 25 iterations, the context window size was set to 10 and all words are kept(min occurrence was set to 0), and we produced 100 dimensional embedding. These hyperparameters generate a vocabulary of 213,000 words and 41,000,000 word pairs for which we generate NLM probabilities, this will be refereed to as the conditional probability dictionary. To generate our conditional probability dictionary we built upon HuggingFace's implementation of Pytorch Pretrained BERT. This allowed up to build on top of OpenAI's GPT-2 Small Model(115M Params) and generate a program that creates the probability of word Y given the context X. We ran this for all word pairs on 3 Nvidia 2080 TI over 4 days. Once the probabilities were generated they can be reused in a wide variety of settings. Since the enwik8 corpus is fairly diverse, our conditional probability dictionary will be a relevant resource for any other corpus wishing to leverage this information.

If a word did not exist in out conditional probability then it was assumed to be zero. Its worth noting that for most conditional pairs the probability is so low that $e^{P(Y|X)} = 0$ which means for most pairs, the conditional probability matrix is almost unchanged from the original Glove method.

3.2 Word Similarity

To explore the quality of our learned word embedding we trained various embedding models and then used well established word similarity benchmarks to compare these embedding. We chose three commonly used word similarity data set compromised of a diverse array of nouns, verbs and adjectives. The benchmarks we used were MEN3k(Bruni et al., 2012) , SimLex999(Hill et al., 2015), and Wordsim353(Finkelstein 2002). Each of these data set contain two words and an associated similarity score assigned by human judges. Measures of correlation indicate how well word vectors are able to

Table 1: Word Similarity by varying the cooccurrence increment function

Cooccurrence Increment	MEN3K	SimLex999	Wordsim353	Average
1(Regular Glove)	0.3144 (0.00%)	0.1465 (0.00%)	0.3840 (0.00%)	0.2816 (0.00%)
$e^P(Y\|X)$	0.3206 (1.97%)	0.1452 (-0.85%)	0.3961 (3.17%)	0.2873 (2.03%)
$e^{(P(Y\|X)+P(X\|Y)}$	0.3214 (2.22%)	0.1368(-6.60%)	0.3932(2.41%)	0.2838(0.77%)
$2e^{(P(Y\|X)+P(X\|Y)}$	0.3445(9.58%)	0.1537(4.94%)	0.4095(6.64%)	0.3026(7.44%)
$2^{e^{(P(Y\|X)+P(X\|Y)}}$	0.3489(10.99%)	**0.1602(9.40%)**	**0.4137(7.74%)**	**0.3076(9.24%)**
$e^{e^{(P(Y\|X)+P(X\|Y)}}$	**0.3602(14.57%)**	0.1584(8.15%)	0.4036(5.10%)	0.3074(9.15%)

predict the similarity judgments. Spearman correlation specifically measures how well word vectors are able to predict the correct rankings of similarity judgments. For our experiments we use the mean spearman correlation for each evaluation benchmark. In the first experiment, detailed in table 1, we modified the increment update function for the cooccurrence matrix generation. Various functions were used to explore the effect that big and small changes had on the word embedding. This experiment shows that when a traditional word embedding is able to incorporate information from a NLM it is able to learn a better, more representative embedding. In the second experiment, detailed in table 2, we dispensed with the notion of increment the cooccurrence matrix and instead set the cooccurrence matrix value to specific values. Surprisingly, all of our iterations of this method performed quite badly and were unable to pass on any of the understanding of semantics and syntax that the NLM had captured. This experiment shows that to apply the knowledge learned in NLM a successful model should modify an existing corpus to increase importance of certain terms, not factorize based on conditional probability.

3.3 Text Classification

To explore the effectiveness of these different embedding we focused on the downstream task of text categorization on the Reuters-21578 data set. We build a logistic regression model based on global average pooling of the embedding of the input. Since both embedding were able to learn a perfect classifier in less than one training epoch we focused on how different embedding affected how much data a model needed to learn the data. To do so, we shuffled the training data and only allowed the models to learn on a various samples of the training data. As shown in table 3 below, the NLM enhance word embedding is able to learn with significantly less data and in training on small data, performance is 3x the regular embedding. We believe this is because NLM have been shown to be universal task learners and some of this ability has been passed on to the enhanced embedding.

4 Conclusion

Our experiments present a novel technique for leveraging contextual information learned by NLM in more traditional word vectors. By leveraging the contextual information form NLMs the new learned embedding out perform the non enhanced embedding by a significant margin in word similarity benchmarks and downstream classification tasks can train on fewer examples. In the future, we would like to explore how NLM can improve other existing word embedding such as Word2Vec and Sent2Vec. Additionally, we would like to explore how NLM knowledge can be used to improve the performance of any word vector after it has been learned.

Table 2: Word Similarity by varying the setting cooccurrence matrix to conditional probability

Cooccurrence Increment	MEN3K	SimLex999	Wordsim353	Average
1(Regular Glove)	**0.3144 (0.00%)**	**0.1465 (0.00%)**	**0.3840 (0.00%)**	**0.2816 (0.00%)**
e^1	0.1761(-43.98%)	0.1761(20.24%)	0.1322(-65.57%)	0.1615(-42.66%)
1	0.0532(-83.08%)	0.0368(-74.89%)	0.0330(-91.42%)	0.0401(-85.45%)
$e^{(P(Y\|X)+P(X\|Y)}$	0.1758(-44.08%)	0.1243(-15.13%)	0.1898(-50.57%)	0.1633(-42.01%)
$(P(Y\|X)+P(X\|Y)$	0.0499(-84.13%)	0.0335(-77.14%)	0.0404(-89.48%)	0.0413(-85.35%)

Table 3: Training data set size and test set accuracy by model

# of Samples	Regular Glove	NLM Enhanced Glove
15	0.0006	0.0074
147	0.0585	0.1823
221	0.2745	0.9940
265	0.3093	0.8809
394	0.9463	1
441	1	1