Generating Embeddings on the Fly

Samrat Halder (sh3970)
Data Science Institute, Columbia University

Supervisors: Dr. Yassine Benajiba and Dr. Daniel Bauer
Department of Computer Science, Columbia University

Abstract

In recent years the NLP community has shown a tremendous amount of interest in language models and embeddings. There has been a lot of advancement in this direction. However, there have also been criticisms about the language models because of their lack of interpretability, huge size, environmental cost. In the past couple of years, a parallel research area has emerged and gained a lot of interest which promises to leverage locally sensitive hash functions to generate embeddings. In this research, we have implemented and experimented with such an architecture based on Neural Projection Skip-Gram (NP-SG) model and Dense Average Network (DAN) for a downstream classification task. We have learned about certain engineering challenges in this framework and overcome some of them to build an end-to-end pipeline. We lay down the path for future research in this direction.

1. Introduction

Pre-trained word embeddings are at the core of natural language processing. In the recent past, researchers have come up with various models to generate embeddings such as word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014). The biggest challenge of these language models is they require a lookup and large memory footprint as one has to store a d-dimensional array for each word in the vocabulary. Also, another issue is that these models cannot handle out-of-vocabulary words. In parallel to this, there has been an increasing demand for deploying these models on edge devices for inferences to address privacy concerns. This has led to new research towards developing small models with lower environmental footprint.

Ravi and Kozareva, 2018 developed an on-device neural model for text classification. They proposed to reduce the memory footprint of large neural networks by replacing the input word embeddings with projection-based representations. They used n-gram features to generate binary LSH randomized projections on the fly surpassing the need to store word embedding tables and reducing the memory size. The projection models reduce the memory occupied by the model from O (|V |) to O(nP), where |V | refers to the vocabulary size and nP refers to number of projection operations. This has two main advantages: (1) they are fixed and have low memory size; (2) they can handle out of vocabulary words. In recent months Google Research has also come up with some advanced neural projection-based architectures for language modeling such as PRADO.

In Sankar et al., 2019, the authors propose to combine the best of both worlds by learning transferable neural projection representations over randomized LSH projections. They do this by introducing a new neural architecture inspired by the skip-gram model of (Mikolov et al., 2013) combined with a deep MLP on top of LSH projections.

In this project, we were interested in exploring and understanding the efficiency and applicability of the neural projection-based embeddings and compare them with other embeddings from language models. However, the major challenge was there was no available open-source codebase that demonstrates neural projections. Therefore, our primary goal was to build an end-to-end pipeline that is not only capable of training a neural projection skip-gram model but also test the model for a standard downstream task such as training a classification model on an SST-fine dataset. For this task, we followed the Deep Averaging Network (DAN) architecture as demonstrated in Iyyer et al., 2015.

2. Methodology

The end-to-end pipeline discussed in the research project consists of two components: 1. A neural projection skip-gram model 2. A Deep Average Network. We first train a two-layer skip-gram network with LSH projections. Thereafter we use the skip-gram embeddings to train a Deep Averaging network consisting of three dense layers to train the classification model on the SST-fine dataset.

2.1. Objectives and Technical Challenges

Our major objective for this project was two-fold: 1. Build an open-sourced end-to-end pipeline for generating neural projection-based 2. Test the embeddings on a downstream task for which we used SST-fine corpus. The major challenge for this project is more related to

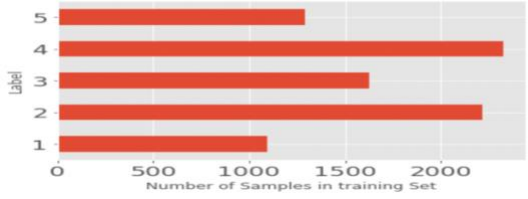
engineering and computing resources. While training the skip-gram models we came across some of the nuisances of large language models and related engineering challenges. Initially, we tried to train our skip-gram model with some example corpora such as Bible Dataset (nltk), SST-fine training corpus however after realizing the model performance may not be optimum with such small datasets, we moved on to wiki9 dataset which contains 140,000 sentences (each of which is often of the length 400-800 words) scrapped from Wikipedia. Even with the wiki9 dataset, the current implementation needs further optimization in terms of speed. In our current implementation, we used up to 30,000 sentences from wiki9.

2.2. Formulation and Design

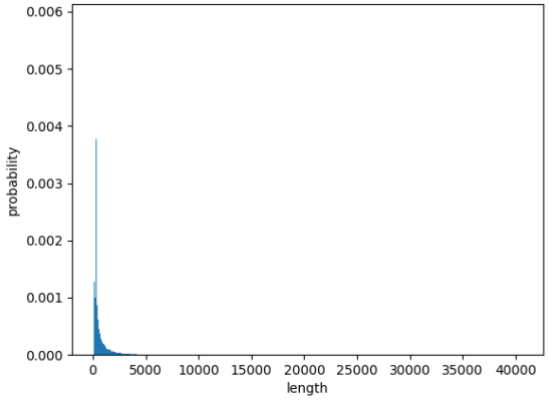
As briefly discussed earlier, we broke down the entire pipeline into two steps, 1. Extracting LSH projections tuned with skip-gram from the neural projection skip-gram model 2. Using these embeddings for training the Deep Averaging Network model with SST-Fine dataset. We fix the number of random projections to 80 and the projection dimension to 14. We use a 2- layer MLP (sizes: [2048, 100]) regularized with dropout (with probability of 0.65) to transform the binary random projections to continuous word representation. For the vanilla skip-gram model, we fix the embedding size to 100. We use different window sizes and negative sampling to generate skip-grams. We learn the parameters using Adam optimizer with a default learning rate of 0.001. We initialize the weights of the MLP using Xavier (Glorot) initialization. For the Deep Average Network, we use these 100-dimensional embeddings. We implement a three-layer dense network of size 500 hidden units with dropout (0.4). We use Relu as the activation function and Adam as optimizer.

3. Dataset

We use three different datasets for training the skip-gram model 1. SST-fine 2. Wiki9 (https://cs.fit.edu/~mmahoney/compression/textdata.html) 3. Bible corpus and SST-fine for the classification task. The SST-fine is a standard 5 class sentiment classification dataset annotated by humans. The dataset is fairly balanced consisting of 8,544 training and 2,210 examples in the test dataset.



We further explored the wiki9 dataset. This dataset contains a very varied length of sentences with a maximum length of 40,591, minimum length of 50, average length of 680 and standard dev. of 977.



4. Implementation

We use Tensorflow2.3 framework for implementing the integrated pipeline. The two modeling frameworks are 1. NP-SG 2. DAN

The NP-SG is built as a skip-gram model on top of a neural projection layer.

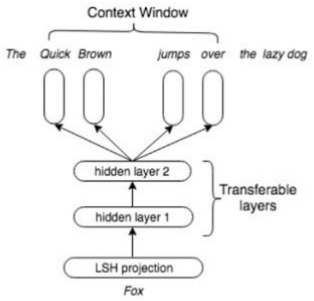


Figure 1: Neural Projection Skip-gram (NP-SG) model

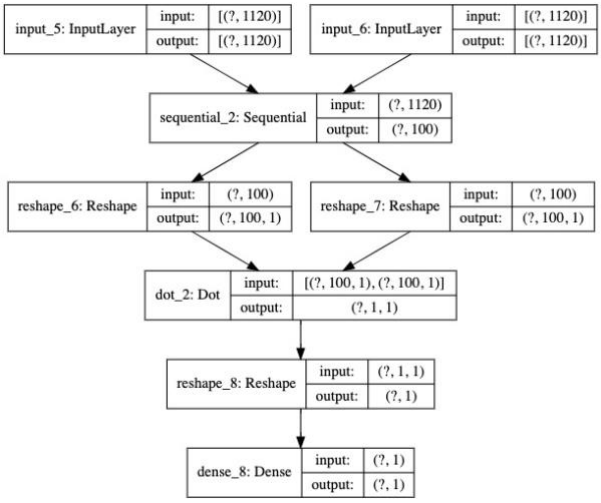


Fig. 2 NP-SG Model built with TF

The DAN model averages input embeddings and passes them through a dense network for the classification task.

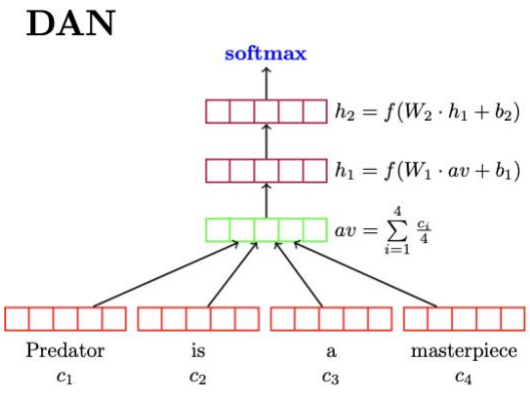


Fig. 3 DAN model overview

5. Results

5.1. Project Results

We did an exhaustive literature review of various contextual and non-contextual language models. We focused our research on neural-projection-based embeddings because in the recent past NLP community has shown significant interest in this direction. We built an end-to-end pipeline to train and test LSH projection-based embeddings generated from a skip-gram model with a Deep Averaging Network. Our initial results with the SST-fine dataset (with 7000 examples to train the NP-SG) model resulted in the best accuracy of 30.9% on the validation data for the 5-class classification task after 10 epochs and with trainable embedding layer the best result we could achieve was 37.68% after 34 epochs. We observed there was no significant change in the loss after the second epoch in the NP-SG training (we trained for 5 epochs). We ran our initial experiments with a very small subset (1000 sentences) from wiki9 dataset. We could reach 27.88% accuracy with non-trainable embedding layer and 37.51% accuracy with trainable embeddings. We experimented with various drop-out rates and set the optimal values as per our findings. A summary of our experiments can be found below:

Trainable Embedding?	NP-SG train Dataset	Skip-gram train Size	Test Acc. (SST-Fine)
No	SST-Fine	7,000	30.9%
Yes	SST-Fine	7,000	37.68%
No	enWiki9	1,000	27.88%
Yes	enWiki9	1,000	37.51%
No	enWiki9	5,000	29.7%
Yes	enWiki9	5,000	38.1%
No	enWiki9	30,000	30.43%
Yes	enWiki9	30,000	38.42

We ran the NP-SG model training for 1 epoch with 30,000 sentences and it took about 12 Hrs on NVIDIA Tesla V100 GPU.

5.2. Discussion of Insights Gained

The major insights we gained from this research was that although projection-based embeddings require lesser environmental footprint and memory, they still need huge corpora of text to incorporate the semantic information of the language. This brings us to certain engineering challenges while training the skip-gram model starting from gathering and preparing the cleaned dataset to set up a robust pipeline and significant computational resources that can enable us to train these models. Our research was focused more on building the proto-type pipeline for data preparation and running some experiments on a small scale. Our initial results seem to be promising and we hope the same codebase can be further improved in the future to run experiments at a large scale by further optimizing speed and leverage the LSH embeddings in various downstream tasks.

6. Future Work

We believe current experiments are promising to carry on further work in this direction. Also, we have open-sourced our codebase for anyone to start experimenting with neural projections more easily. However, we think our codebase needs some optimization to make it feasible to run faster and needs some experiments with the model

hyperparameters to achieve better convergence. Also going forward, we might want to try other classification architectures to test our neural projection-based embeddings.

6. Acknowledgment

I want to take this opportunity to sincerely thank Dr. Yassine Benajiba and Dr. Daniel Bauer to give me the opportunity to carry on this research and spend their valuable time mentoring me throughout this project. I want to thank my colleagues Jake Stamell and Param Popat for collaborating with me on this work. Finally, I would like to express my gratitude to Dr. Yassine Benajiba for introducing me to the fascinating world of NLP during my master's program at Columbia University. I am very excited to learn more about this in my professional career in the coming years and hopefully do more work in the future!

7. References

[1] (Mohit Iyyer, 2015) Deep Unordered Composition Rivals Syntactic Methods for Text Classification

[2] (Chinnadhurai Sankar, 2019) Transferable Neural Projection Representations

[3] (Ravi, 2017) ProjectionNet: Learning Efficient On-Device Deep Networks Using Neural Projections

[4] (Chinnadhurai Sankar S. S., 2019) Do Neural Dialog Systems Use the Conversation History Effectively? An Empirical Study

[5] (Chinnadhurai Sankar S. R., 2019) On the Robustness of Projection Neural Networks For Efficient Text Representation: An Empirical Study

[6] (Sujith Ravi, 2018) Self-Governing Neural Networks for On-Device Short Text Classification

[7] (Prabhu Kaliamoorthi, 2019) PRADO: Projection Attention Networks for Document Classification On-Device

[8] (John Wieting, 2019) No Training Required: Exploring Random Encoders for sentence classification

[9] GitHub Codes[1]

[1] All codes are available here https://github.com/samrat-halder/Neural-Projection-Skip-Gram-DAN