Natural Language Processing
Assignment 1 Report

Tsinghua University

Albert Millan – 2019280366

Contents

1 Introduction

This report explains the methodology involved to deploy Word2vec employing a skip-gram model in Pytorch. The first section shows the the derivation of the gradient of the softmax loss.

2 Softmax Loss

To simplify the gradient computation operation, we take the logarithm of the softmax.

$$\begin{aligned}
\frac{\partial L}{\partial v_c} &= -\log\Big(\frac{e^{u_o^T * v_c}}{\sum_{w=1}^{W} e^{u_w^T v_c}}\Big) \\
&= -\log\Big(e^{u_o^T v_c}\Big) + \log\Big(\sum_{w=1}^{W} e^{u_w^T v_c}\Big) \\
&= -\frac{1}{e^{u_o^T * v_c}} * e^{u_o^T v_c} * \frac{\partial}{\partial v_c}\Big(u_o^T v_c\Big) + \frac{1}{\sum_{w=1}^{W} e^{u_w^T v_c}} * \Big(\sum_{x=1}^{W} e^{u_x^T v_c}\Big) * \frac{\partial}{\partial v_c}\Big(u_x^T v_c\Big) \\
&= -u_o^T + \sum_{x=1}^{W} \frac{e^{u_x^T v_c}}{\sum_{w=1}^{W} e^{u_w^T v_c}} * u_x \\
&= -u_o + \sum_{x=1}^{W} p(x|c) * u_x
\end{aligned}$$

3 Word2Vec Implementation

3.1 Pre-processing

Gensim's library was used to import the corpus of the Wikipedia dataset and convert it from *XML* to *text* file, while also removing punctuation, one article per line of text document. In order to remove noise, the following operations are performed next:

1. Delete those word tokens that have a low frequency in the corpus (less than five words).

2. Remove word tokens that appear in excess using sub-sampling.

3. Delete word tokens that have less than three characters long.

4. Remove sequences of repeated word tokens within the corpus.

Upon successful completion of the aforementioned noise-removal operations, the word2id and id2word dictionaries, as well as the integer encoded word representation of the corpus are generated. A table containing randomly shuffled tokens, each in a proportion equivalent to their frequencies in the corpus is generated to sample the negative samples during training.

3.2 Skip-gram

I employ the skip-gram model to compute center and context word pairs, generating negative samples for each pair. To ensure this procedure is performed effectively, a custom torch *Dataset* and *Dataloader* is developed. When querying for new data element, the function returns an array containing the center word identifier (v_c) of the same size as the array containing the context words (u_o), the actual context word array and the matrix containing the negative samples (u_w). Thus, when taking a batch, we are taking as many distinct center word elements as the size of the batch and their corresponding context words, each appended together forming two single arrays, as well as a vertical stack of the negative samples.

The loss employed to train the model is shown in equation 1. Torch automatically computes the sparse gradients of the embeddings of the negative samples, center words and context words.

$$L = -\log \sigma(u_o v_c) - \sum_{k=1}^{K} \log(\sigma(u_k v_c)) \tag{1}$$

For each training iteration, we compute the spearman coefficient and loss. These are plotted in figures 1 and 2. The experiment involves using three embedding sizes (100, 200 & 300), with three distinct corpus sizes (10, 100 & 10000 articles). These are presented in table 2.

3.3 Results

The graphs in figure 1 indicate that models trained using larger embedding sizes take longer to converge and also converge to to more optimal solutions in terms of the loss. Notwithstanding, the maximum spearman coefficient obtained is marginally superior on models trained with larger embedding sizes. Graphs indicate this is the case irrespective of the size of the training dataset.

Figure 2 suggests that the dataset size significantly influences the capability of the model to draw meaningful relationships across words. For instance, the improvement in the spearman correlation on models trained with fixed embedding sizes and datasets of 10 and 100 article sizes increases by up to $\tilde{0}.2$. This increase is marginal between dataset sizes of 100 and 10000. Despite this, smaller datasets converge better according. Thus, there is a minimum threshold of words in a dataset required to better capture meaningful relationships across words. The greater the exposure of center word tokens to context word tokens, the better the model can define the relevant relationships.

4 Enhancements on Word2Vec

4.1 Incorporating Wordsense

I attempted to incorporate word sense by including sense embeddings. The intuition involves initializing sense embeddings based on existing center word embeddings obtained during the training of skip-gram model as aforementioned and replacing the context words by words present in their sense definition should they yield a cosine similarity beyond a certain threshold, for any given center word. I failed to implement this within the time restrictions.

4.2 Fasttext

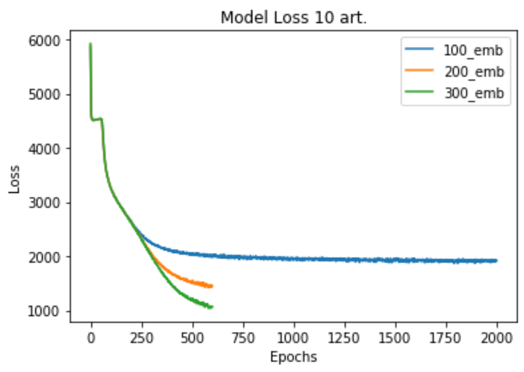
A script implementing *fasttext* was developed and tested on the dataset involving 10 wikipedia articles and embeddings of size 300. Fasttext yields superior performance than the implemented word2vec model.

Emb. Sz.	**Dataset Sz.**	**Word2vec**	**Fasttext**
300	10	44.71%	49.1%

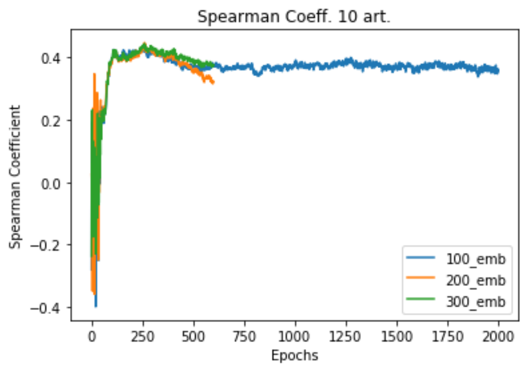
Table 1: Results table

Emb. Size	**Dataset Size**	**Window Size**	**Iters**	**Batch Size**	**Lrn. Rate**	**Sp. Coeff. (%)**
100	10	5	600	512	3e-3	42.85
100	100	5	300	512	3e-3	44.71
100	10000	3	300	512	3e-3	44.40
200	10	5	600	512	3e-3	56.49
200	100	5	300	512	3e-3	59.41
200	10000	3	300	512	3e-3	58.94
300	10	5	600	512	3e-3	59.83
300	100	5	300	512	3e-3	60.67
300	10000	3	300	512	3e-3	62.61

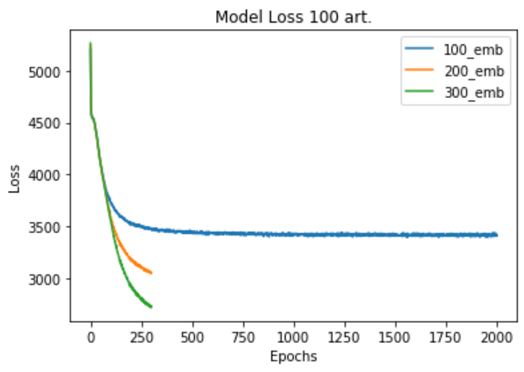
Table 2: Hyperparameters employed to train the skip-gram model and results obtained as per *spearman coefficient*.



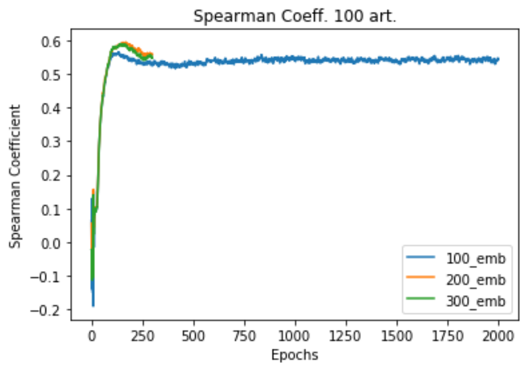
(a) Model loss trained on 10 wikipedia articles



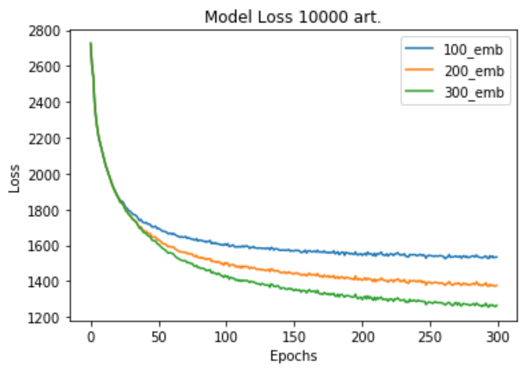
(b) Model spearman coeff. trained on 10 wikipedia articles



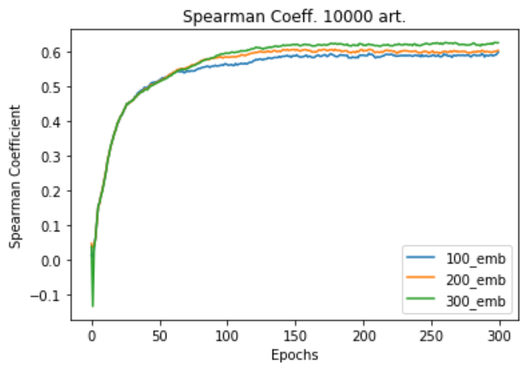
(c) Model loss trained on 100 wikipedia articles



(d) Model spearman coeff. trained on 100 wikipedia articles

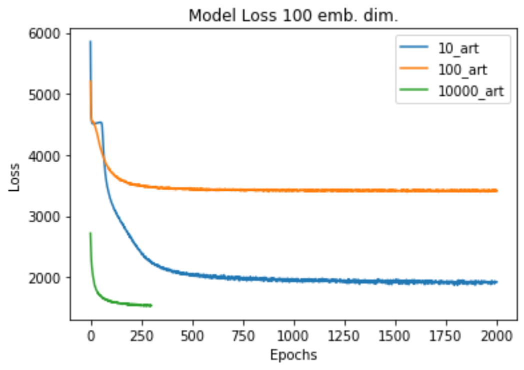


(e) Model loss trained on 1000 wikipedia articles

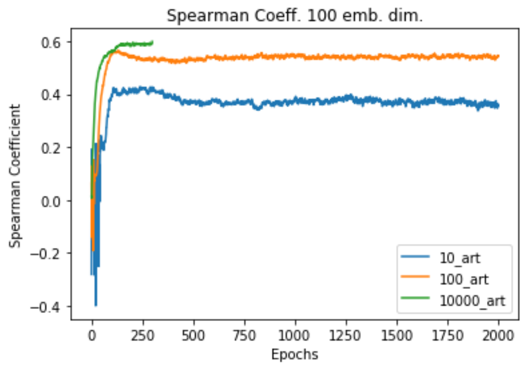


(f) Model spearman coeff. trained on 1000 wikipedia articles

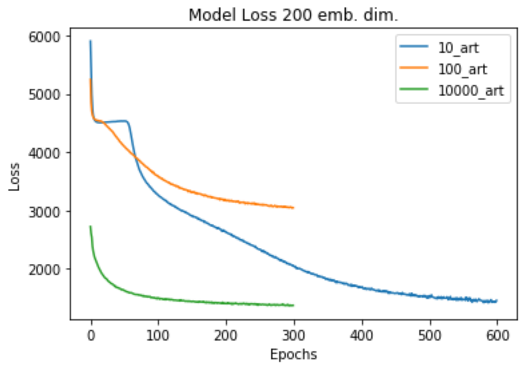
Figure 1: Comparison of loss and spearman coefficient across distinct embedding size on fixed article size.



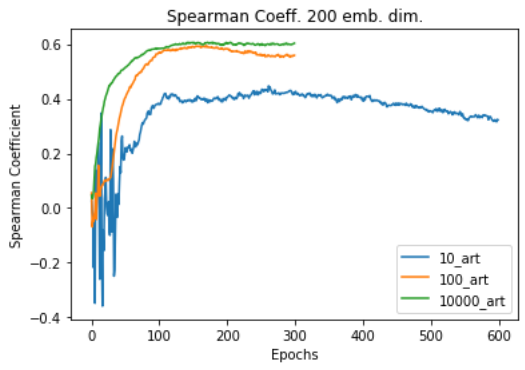
(a) Model loss trained on 10 wikipedia articles



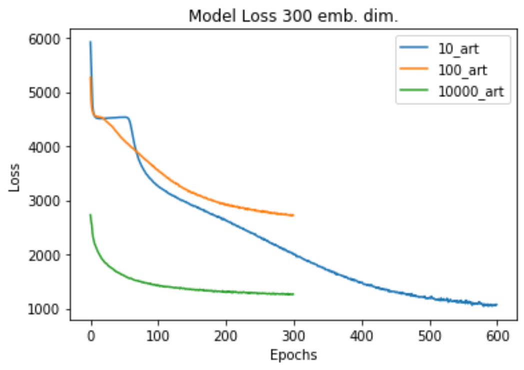
(b) Model spearman coeff. trained on 10 wikipedia articles



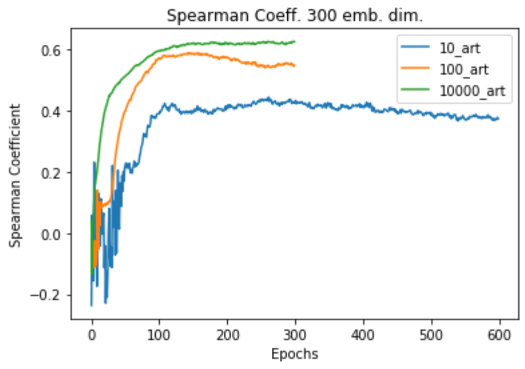
(c) Model loss trained on 100 wikipedia articles



(d) Model spearman coeff. trained on 100 wikipedia articles



(e) Model loss trained on 100 wikipedia articles



(f) Model spearman coeff. trained on 100 wikipedia articles

Figure 2: Comparison of loss and spearman coefficient across fixed embedding size on distinct article size.