Assignment 1 - SkipGram with Negative Sampling

Natural Language Processing Course - Spring - 2019
Ecole CentraleSupelec, Paris

Benoit Laures, Paul Asquin, Ayush K. Rai
benoit.laures@student.ecp.fr, paul.asquin@student.ecp.fr, ayush.rai2512@student-cs.fr

July 5, 2019

In this short report we explain our experiments (varying the hyperparameters) in the computation of word vectors using skip-gram model and negative sampling **Section 1**. And in the **Section 2**, we describe some additional strategies that we have used for reducing the gender bias on trained word embedding.

Note : Due to computational power issues, the maximum number of lines we could use for training our skipgram model was only 1000.

1 Experiments on Word2Vec Computational Task

1.1 Overview

The SkipGram Word2Vec architecture is represented in figure 1. For every word in the Corpus of size V, SkipGram model learns an embedding of size N based on the context of the word. The model has 1 hidden layer (without any activation function) whose weight matrix is of size $V \times N$ and the model tries to predict its context words within the size of a window (another hyperparameter). Please go through the **README.md** for details of pipeline, optimization, initialization of parameters, negative sampling etc. Below we describe the results of some of our experiments.

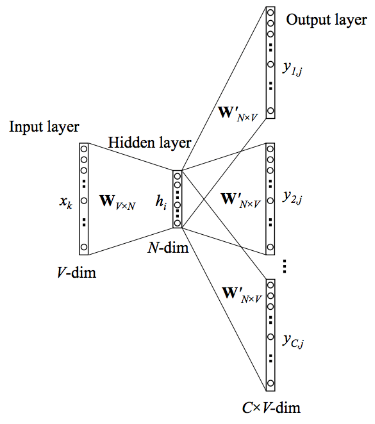

Figure 1: Word2Vec - SkipGram Architecture

1.2 Experimental Results

1.2.1 Experiment 1 (Vocabulary Size : 1083 words)

NUMBER_LINES = 100
N_EPOCHS = 150
DECAY_INTERVAL = 5
EMBEDDED_SIZE = 50
BATCH_SIZE = 128
NEGATIVE_SAMPLING_SIZE = 5
INITIAL_LEARNING_RATE = 1e-2
DECAY_FACTOR = 0.99
WINDOW_SIZE = 5

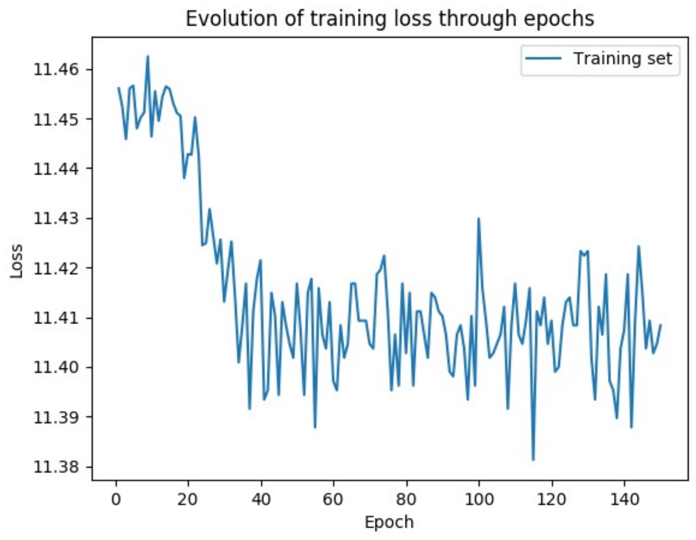

Figure 2: Experiment 1

Here we conclude that the training is proceeding in right direction as the training loss is decreasing but we need to increase the training dataset and reduce the learning rate.

1.2.2 Experiment 2

NUMBER_LINES = 1000 (Vocabulary Size : 6091 words)
N_EPOCHS = 50
DECAY_INTERVAL = 1
EMBEDDED_SIZE = 200
BATCH_SIZE = 256
NEGATIVE_SAMPLING_SIZE = 3
INITIAL_LEARNING_RATE = 5e-3
DECAY_FACTOR = 1
WINDOW_SIZE = 3

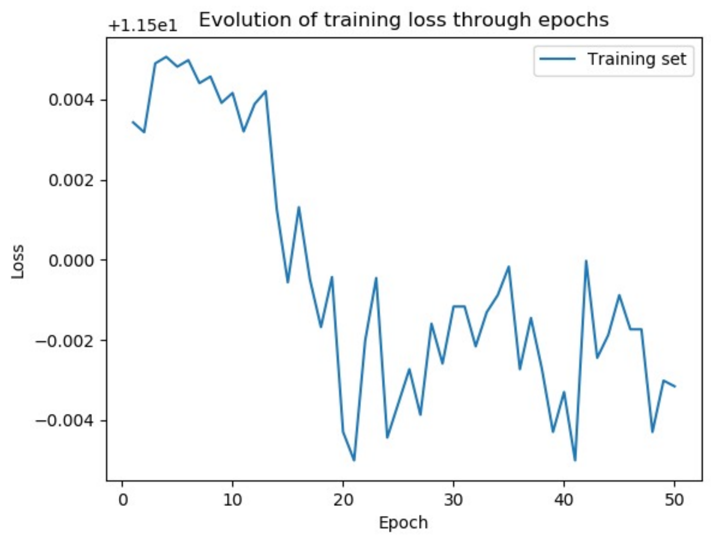

Figure 3: Experiment 2

Here we observe that the training loss is decreasing (almost zero) with increase in number of epochs.

1.2.3 Experiment 3

NUMBER_LINES = 1000 (Vocabulary Size : 6091 words)
N_EPOCHS = 70
DECAY_INTERVAL = 5
EMBEDDED_SIZE = 50
BATCH_SIZE = 128
NEGATIVE_SAMPLING_SIZE = 5
INITIAL_LEARNING_RATE = 1e-2
DECAY_FACTOR = 0.99

WINDOW_SIZE = 3

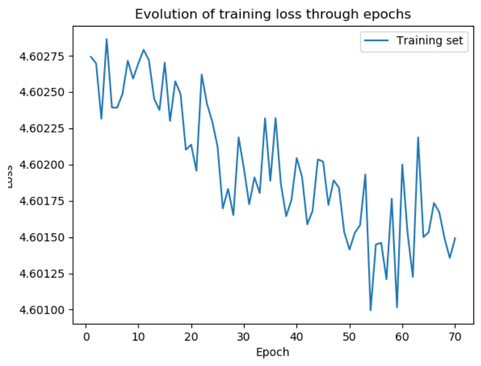

Figure 4: Experiment 3

Here we observe that even though most of the parameter from Experiment 2 are same but we initialized learning rate from $1e-2$ and this gives much bigger value of loss than previous experiment.

1.2.4 Experiment 4

NUMBER_LINES = 1000 (Vocabulary Size : 6091 words)
N_EPOCHS = 60
DECAY_INTERVAL = 5
EMBEDDED_SIZE = 300
BATCH_SIZE = 128
NEGATIVE_SAMPLING_SIZE = 8
INITIAL_LEARNING_RATE = 1e-2
DECAY_FACTOR = 0.99
WINDOW_SIZE = 5

Here we observe that the training is diverging. A major reason for this might that our embedding size (300) is too big in comparison to the size of the vocabulary and therefore this embedding size not a good choice of hyperparameter for given size of vocabulary.

1.3 Impact of Embedding Size

We observe that since the maximum number of lines that we used from the dataset were only 1000 i.e with vocabulary size of 6091, having too large embedding size makes it hard to learn properly.

1.4 Impact of Negative Sampling Size

We observe that the negative sampling size between 5-10 yield better results. This substantially reduces that time complexity of the algorithm.

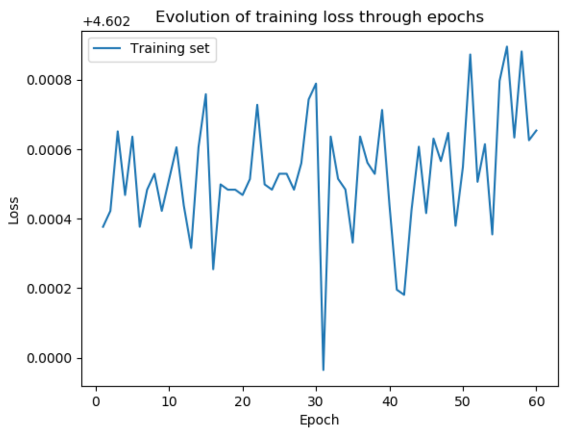

Figure 5: Experiment 4

1.5 Impact of Window Size

We observe that having a small window size doesn't capture context information of the word properly, whereas having too large window size captures general information instead of word's context specific information.

1.6 Implementation Tricks

See also on README

- **Gradient Checking** : We performed gradient checking by comparing the analytical gradient to the numerical gradient in order to ensure that gradient flow was normal while training.
- **Gradient Clipping** : Since there is no activation function in the model, exploding gradient leads to exponential overflow problem even with negative sampling, which we tackled using gradient clipping.

1.7 Weakness of our Model

A big issue we faced and that we spent a very long time on was the exploding gradient problem which made the exponential computation in the softmax overflow, even with negative sampling. Thus, managed to solve this problem by putting a saturation before the exponential (values are in [-10, 10]) and we performed gradient clipping to prevent it from exploding.

Besides, the model has to be trained on an enormous amount of data to display good results, thus it must undergo a very long phase of training (more than a day) which is very hard to perform analysis, optimizing parameters or just tests.

2 Reducing Gender Bias on Word Embeddings

Word embeddings capture semantics in the language, and are used in variety of NLP and machine learning applications like sentiment analysis, sentence classification etc. However despite of having many useful properties, word embeddings exhibit gender biases as shown in [1]. In our work in we first explain the direct and indirect gender biases. We also perform post processing on learnt word embeddings to mitigate gender biases in word embeddings, which can be further utilized in many downstream tasks.

2.1 Understanding Gender Bias

Here we present some advance work on reducing gender bias on our word embeddings.

- **Direct Bias**: It is measured as the relationship between a gender-neutral word like doctor, computer programmer and a gendered word pair like as (she,he), (female,male), (woman,man). Direct Bias is given by:

$$DirectBias = \frac{1}{|N|} \sum_{n \in N} |Cos(\vec{w}, g)|$$

  which represents the the average of the cosine similarity between the word vectors in N, a set of gender neutral words and the gender direction g. In the next section we explain how to compute the gender bias direction g.

  Indirect Bias: Direct bias fail to capture word receptionist is more closer to word softball than compared to word softball. This observation is largely a consequence of the respective relationships between receptionist and softball to words such as woman and she. [1] suggests that in order to measure the indirect bias we first need to decompose all of our normalized word vectors into v_g and $v_\perp$, where $v_g = (v.g)g$ represents the gender component and $v_\perp = v - v_g$. Indirect Bias β between any two gender neutral word pair (w, v) is given by:

$$InDirectBias = \beta(w, v) = \frac{w.v - \dfrac{w_\perp v_\perp}{||w_\perp||_2 ||v_\perp||_2}}{w.v}$$

2.2 Strategy to Reduce Gender Bias

- **Identify Gender Bias Direction**

  In order to compute Gender Bias direction g, we first compute $g_1 = e_{woman} - e_{man}$, $g_2 = e_{mother-father}$ and $g_3 = e_{girl} - e_{boy}$ and then take the average of them to get gender bias direction g. The paper referred uses more complicated method in involving Single Value Decomposition to find gender bias direction however our method is good for naive implementation.

- **Neutralization Step**

  Given that we have an embedding e for a gender neutral word like receptionist. The neutralization step removes the gender bias of gender neutral word by projecting it on the space, which is orthogonal to the gender bias axis.

$$e^{bias\_component} = \frac{e \cdot g}{||g||_2^2} * g$$

$$e^{debiased} = e - e^{bias\_component}$$

Here $e^{bias\_component}$ as the projection of e onto the direction g and then we subtract this term from e to get $e^{debiased}$. This is equivalent to orthogonal projection with respect to g.

- **Equalization Step:** By applying neutralizing to "computer" we can reduce the gender-stereotype related with it. But this does not guarantee that word pair ("he","she") are equidistant from "computer". Therefore apart from neutralizing gender neutral words, we also need to apply Equalization to word pairs like (grandfather,grandmother) and (actor,actress) to ensure that such word-pairs differ only in the gender property. The main idea behind equalization is to make sure that such particular pair of words are equi-distant from $g_\perp$, where $g_\perp$ represent a vector perpendicular to gender bias direction g. This step also ensures that the equalized word pair eg. ("male","female") are now the same distance from debiased gender neutral word like $e^{debiased}_{computer}$, $e^{debiased}_{professor}$, $e^{debiased}_{scientist}$.The equations governing equalization step are given by following equations as mentioned in [1]:

Compute the mean μ of e_{w1} and e_{w2} corresponding to the embedding both the gendered words to be neutralized.

$$\mu = \frac{e_{w1} + e_{w2}}{2}$$

Compute the projections of μ over the gender bias direction and the axis orthogonal to gender bias direction to get μ_B

$$\mu_B = \frac{\mu \cdot \text{bias\_axis}}{||\text{gender\_bias\_axis}||_2^2} * \text{gender\_bias\_axis}$$

$$\mu_\perp = \mu - \mu_B$$

Similarly compute the projections of e_{w1} and e_{w2} over the gender bias direction and the axis orthogonal to gender bias direction to get e_{w1B} and e_{w2B}

$$e_{w1B} = \frac{e_{w1} \cdot \text{gender\_bias\_axis}}{||\text{gender\_bias\_axis}||_2^2} * \text{gender\_bias\_axis}$$

$$e_{w2B} = \frac{e_{w2} \cdot \text{gender\_bias\_axis}}{||\text{gender\_bias\_axis}||_2^2} * \text{gender\_bias\_axis}$$

Adjust the gender bias part of e_{w1B} and e_{w2B} to get corrected projections $e^{corrected}_{w1B}$ and $e^{corrected}_{w2B}$

$$e^{corrected}_{w1B} = \sqrt{|1 - ||\mu_\perp||_2^2|} * \frac{e_{\text{w1B}} - \mu_B}{|(e_{w1} - \mu_\perp) - \mu_B)|}$$

$$e^{corrected}_{w2B} = \sqrt{|1 - ||\mu_\perp||_2^2|} * \frac{e_{\text{w2B}} - \mu_B}{|(e_{w2} - \mu_\perp) - \mu_B)|}$$

Debias by equalizing e_1 and e_2 to the sum of their corrected projections

$$e_1 = e^{corrected}_{w1B} + \mu_\perp$$

$$e_2 = e^{corrected}_{w2B} + \mu_\perp$$

The analysis of gender debiasing algorithms is done in detail in the **Debiasing_Word_Vectors.ipynb** attached in the zip file. We test our model on the standard 50-dim glove model and also on our trained word2vec skipgram model.

3 Conclusions

SkipGram word2vec models captures context information of the words. It's various hyperparameters depend on the size of corpus and on the underlying extrinsic NLP task. Word embeddings are highly vulnerable to gender biases, which can be removed by identifying the gender bias direction and performing neutralization of genderless words and equalization of gendered word pairs.

4 References

1. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, Adam Kalai [Link]

2. Distributed Representations of Words and Phrases and their Compositionality. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean [Link]