Self Supervised Representation Learning

Imandi Bhargava Veera Laksmana Raju
SR-NO : 21190
MTech CSA
Indian Institue Of Science
bhargavav@iisc.ac.in

Abstract

Analysis on one of Self Supervised Representation Learning method : Efficient Estimation of Word Representations in Vector Space and comparing results of using Embeddings from Skip-gram model and Supervised Learning model

1 Introduction

1.1 Self Supervised Representation Learning:

Self-supervised representation learning (SSRL) methods aim to provide powerful, deep feature learning without the requirement of large annotated data sets.Supervised learning is good at solving tasks when there are enough labels available, but it can be difficult and expensive to collect manual labels. There is a vast amount of unlabelled data available that could be used, but unsupervised learning is not as efficient as supervised learning.However, self-supervised learning can solve this problem by using unlabelled data and framing a supervised learning task in a specific way to predict a subset of information using the rest. This approach provides all the required inputs and labels for training, resulting in better efficiency and less dependence on manually curated data sets.

1.2 Word2Vec :

As part of my term project i read the paper **"Efficient Estimation of Word Representations in Vector Space"** by Tomas Mikolov et al. [1] which is comes under (SSRL). I provide my understanding of the paper in this report.

Implemented the **Skip Gram neural network model** of word2vec from scratch in Pytorch to learn word vectors of the **tiny-shakespeare** data set and perform classification on **IMDB movie reviews** using that representations .Comparing the results with **Logistic Regression** with different ratios of test and train data.

2 Word2Vec

2.1 Word Representations :

Word representations are crucial for various natural language processing tasks, including document classification, named entity recognition, and sentiment analysis.The semantic and syntactic meaning contained in these vectors make them appropriate feature vectors for these tasks. The bag-of-words model is commonly used to construct feature vectors, which use the tf-idf scores of words as their representation. However, this approach loses word order and fails to capture semantic relationships among words. Distributed representations of words, which learn word representations from large data sets, have become popular. The neural network language model (NNLM) is a feedforward neural network that jointly learns the word vector and a statistical language model.

The word2vec family has two major models: the **Continuous Bag-Of-Words model** and the **Continuous Skip-Gram model**. These models minimize the computational complexity of the model i.e, parameters of the model to be learnt while maximizing accuracy in terms of syntactic and semantic integrity among the word vectors.

2.2 Continuous Bag-Of-Words Model:

The CBOW model aims to classify the middle word accurately using the k previous and k future words as context inputs, forming a context size of C. Unlike the feed forward NNLM, the CBOW model removes the non-linear hidden layer and uses one-hot encoded C words $x_1, x_2, .., x_c$ with 1 at the index of the word in the vocabulary in V dimensions(vocabulary size). The model's hidden layer is an N - dimensional vector h, and the output is the middle word y in one-hot encoding. The one-hot encoded inputs are projected into the hidden layer through a $V * N$ weight matrix W, representing the word vector of the i^th word in the vocabulary. As the input vectors are projected into the same position, the order of input words does not matter, and the hidden layer connects to the output layer via a $N * V$ weight matrix W with softmax as the activation function.The architecture of CBOW is presented in Figure 1

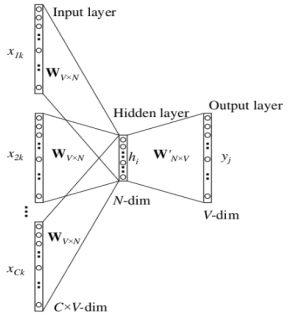

Figure 1: CBOW Model Architecture

2.3 Continuous Skip Gram Model :

The Continuous Skip Gram Model is similar to CBOW, but instead of predicting the current word given its context, the model predicts words in the context of the current word. The log-linear classifier takes each current word as input and predicts words within a context window before and after the current word. The input vector x_k is the one-hot encoded current word, and the output context words $y_1, ..., y_c$ are also one-hot encoded. The input word is projected to the hidden layer of N neurons using a weight matrix W of dimensions $V * N$, where the i^{th} row of W with N dimensions represents the word vector of the i^{th} word in the vocabulary. Each output vector y_i has an associated weight matrix $N * V$ represented as W with softmax as the activation function. The skip-gram model architecture is shown in figure 2.

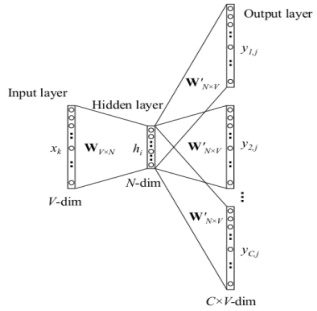

Figure 2: Skip Gram Model Architecture

3 Implementation :

3.1 Data Set :

As I mentioned previously i used **tiny-shakespeare** dataset for making representations of words.I pre-processed the sentences in that dataset and made tokenization on unique words present in dataset.Now iam makeing input to skip gram model. I took window size as 5 i.e, 2 past words and 2 future words.So for every word i make a skip gram input to give to model.

3.2 Networks:

Encoder Network : One linear layer with input dimensions N (Number of Words) and output dimensions D (Word embedding dimension).

Decoder Network : One linear layer with input dimensions D (Word embedding dimension) and output dimensions N (Number of Words) . Here output function is soft-max function

Skip Gram Model : Adding encoder,decoder to this model with same input and output dimensions.Made function for getting embedding of words.

3.3 Loss Function :

The skip-gram model is designed to estimate the conditional probability of a sequence of context words given a target word vector for a log-linear classifier. Specifically, the model seeks to estimate the probability of observing context words within a window of size c around the target word wt, as expressed in equation (1). The objective function $J'(\theta)$ aims to maximize the probability of any context word given wt. This probability distribution can be expressed as a negative log-likelihood function $J(\theta)$, whose derivative is given by:

$$J'(\theta) = \prod_{t=1}^{T} \prod_{-c \leq j \leq c, j \neq 0} p(w_{t+j} \mid w_t; \theta) \tag{1}$$

In this expression, $J'(\theta)$ is the objective function used in the skip-gram model for learning word embeddings. It is defined as the product of the probabilities of observing the context words w_{t+j} given the target word w_t and the model parameters θ. The outer product is taken over all target words in the training corpus, and the inner product is taken over all context words within a fixed window of size $2c$. The notation "$p(w_{t+j} \mid w_t; \theta)$" represents the probability of observing the context word w_{t+j} given the target word w_t and the model parameters θ.In equation (1) can be reformulated for each target word w_{t+j} as:

$$p(w_{j+t} \mid w_t) = \frac{\exp(\mathbf{w}j + t^\top \mathbf{w}t)}{\sum -c \leq j \leq c, j \neq 0 \exp(\mathbf{w}j + t^\top \mathbf{w}_t)} \tag{2}$$

In equation (2) is similar to that of the softmax classifier, used when there are multiple output classes in a neural network's last layer. The skip-gram model uses the softmax classifier to get a probability distribution for the context words within a window. The softmax function minimizes the cross-entropy between the predicted and true label vectors. Back-propagation in the skip-gram model is expressed as cross-entropy equations.

$$L = -\log p(w_{-c}, w_{-c+1}, \ldots, w_{c-1}, w_c | w_t) \tag{3}$$

$$L = -\log \left(\prod_{c=1}^{C} \frac{\exp({w_j}^T w_t)}{\sum_{-c \leq j \leq c, j \neq 0} \exp({w_j}^T w_t)} \right) \tag{4}$$

Intuition behind Loss function:To maximize the probability of the context words and minimize the probability of non-context words for the given target word. I trained model with this loss function on given data set for 5 epochs only due it takes so much of time. I will train model for 15 epochs before the final report submission.

3.4 Classification :

After getting the vector representation of words in corpus .I have implemented sentiment analysis on **IMDB movie reviews**. I took mean of all word vectors present in a review to make a vector for review. Similarly i made for every sentence in reviews. Now passing this vectors to Logistic regression to separate positive and negative reviews. Additionally i have implemented the Logistic regression on original data set using simple count vectorizer for making representation of reviews and compare the results between two models.

3.5 Results :

This is my Github repo for implementation of Skip Gram

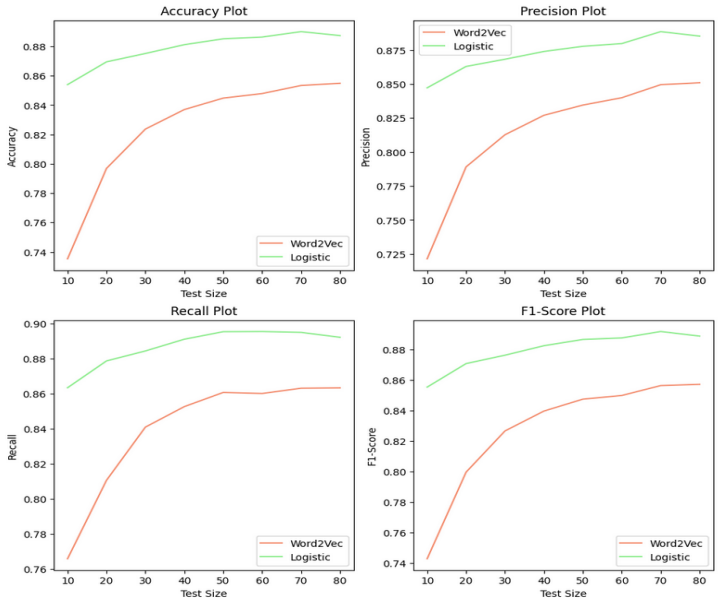

Figure 3: Skip Gram Model Architecture

3.5.1 Analysis:

Accuracy of SSRL is less compared to Supervision models. May be reason behind this is i made a linear classifier on mean of all word2vectors in a each review.That's why this accuracy is low.There are two classes only and SSRL works very good on large data sets. Here iam using only 50000 samples for classification.

3.5.2 Future Work:

Now i wanted to add one more network with 2 layers of Bi-LSTM and 2 fully connected layers with sigmoid function as output on top the model for classification analysis. So that iam thinking it improves the accuracy of SSRL comparing with Logistic regression .I want to train the Skip Gram Model for 15+epochs. As we know **BERT** is very large model compared to this. So I will study on BERT and its features and try to include BERT features in this Word2Vec model if it satisfies.

4 References :

1 . **"Efficient Estimation of Word Representations in Vector Space"** by Tomas Mikolov et al.

2.Xin Rong. "Word2vec Parameter Learning Explained". In: CoRR, abs/1411.2738 (2014) figures for architecture models are taken from this paper

3.https://github.com/ni9elf/Word2VecImplementation took the equations for loss function from his work.

4.https://github.com/kawinm/Deep-Learning-for-NLP taken help from his implementation.