Implementing word2vec: Efficient Estimation of Word Representations in Vector Space

2017 - CS/DS 866 Machine Learning II

Sneha A.* IMT2013045

Nigel Fernandez* IMT2013027

{Sneha.A, NigelSteven.Fernandez}@iiitb.org

IIIT-Bangalore

May 19, 2017

Contents

*Both authors contributed equally to this work.

List of Figures

1 Problem Statement

As part of our project, we have completed:

1. Read the paper "Efficient Estimation of Word Representations in Vector Space" by Tomas Mikolov et al. [1]. We provide our understanding of the paper in this report.

2. Implemented the skip gram neural network model of word2vec from the paper in Keras to learn word vectors of the top 2000 most frequent words in a $100MB$ manuscript [2] from 10^5 generated training samples.

3. Validated our results by providing a t-SNE visualization of the word vectors to demonstrate their effectiveness in keeping syntactic and semantic regularities among words. We also demonstrate the preservation of linear regularities among words by word2vec by forming semantic word relationship examples.

2 word2vec

2.1 Word Representations

The representation of words play an important role in many natural language processing tasks like 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 most common approach for constructing a feature vector of texts is using the bag-of-n-grams approach due to its simplicity, efficiency and acceptable accuracy. The bag-of-words (BOW) model uses the tf-idf scores of words as their representation. This causes the word order to be lost as we consider order independent frequency counts. The bag-of-n-grams model also considers word order in short context while suffering from data sparsity and high dimensionality. The formed representations of words fail to hold semantic relationships among words.

Distributed representation of words paved the way for the design of machine learning and neural models to learn word representations from large data sets. A distributed representation of a word is so called since each vector element contributes to the representation of multiple words and each word is represented by contributions (real numbers not equal to 0) from multiple vector elements (as opposed to a sparse one-hot representation). A popular model for learning continuous vector representations of words is the neural network language

model (NNLM). A NNLM is a feed forward neural network with a linear projection layer and a non-linear hidden layer to jointly learn the word vector and a statistical language model.

Mikolov et al. propose two model architectures for learning distributed representations of words that minimize the computational complexity of the model (parameters of the model to be learnt) while maximizing accuracy in terms of syntactic and semantic integrity among the word vectors. The key observation is that their model is similar to NNLM with the exception of the removal of the non-linear hidden layer. A simpler model for reduced complexity is constructed by using only linear hidden layers. The two major models in the word2vec family are the continuous bag-of-words (CBOW) model and the continuous skip-gram model.

3 Continuous Bag-of-Words Model

The CBOW model has to correctly classify the current (middle) word given the context window as input. The context window consists of k previous words and k future words to form a context size of C. The CBOW model is similar to the feed forward NNLM with the modification of the removal of the non-linear hidden layer. The input to the CBOW model is C words $x_1, \ldots, x_c$ as one-hot encodings. With the vocabulary size as V, the one-hot encodings will be of V dimensions with 1 at the index of the word in the vocabulary. The hidden layer is a N dimensional vector h. The output of the model is the middle word y in one-hot encoding form. The one-hot encoded inputs are projected into the hidden layer through a $V \times N$ weight matrix W. The ith row of W with N dimensions represents the word vector of the ith word in the vocabulary. Observe that this operation projects all input vectors into the same position resulting in averaging their encodings. Since an averaging is performed, the order of the input words in the context window does not matter. The hidden layer is connected to the output layer through a $N \times V$ weight matrix W' with softmax as the activation function.. The architecture of CBOW is depicted in fig 1.

4 Continuous Skip Gram Model

4.1 Model Description

The second model is similar to CBOW, instead of predicting the current word given its context, the model has to predict the words in the context of the current word. Each current word is an input to the log-linear classifier and the model predicts words within a context window (neighbourhood) before and after the current word. The input x_k represents the one-hot encoded vector corresponding to the current word in the training sample. The words $y_1, \ldots, y_C$ are the one-hot encodings of the output context words. The input word is projected to the hidden layer of N neurons through a weight matrix W of dimensions $V \times N$. The ith row of W with N dimensions represents the word vector of the ith word in the vocabulary. Each output vector y_i has an associated weight matrix $N \times V$ represented as W' with softmax as the activation. The projection matrix acts as a lookup table since we are using one-hot encodings. The architecture of skip gram is depicted in fig 2.

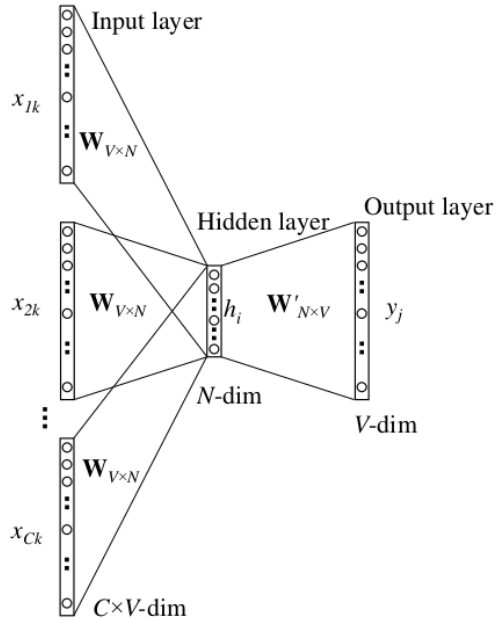


Figure 1: CBOW Model Architecture [3]

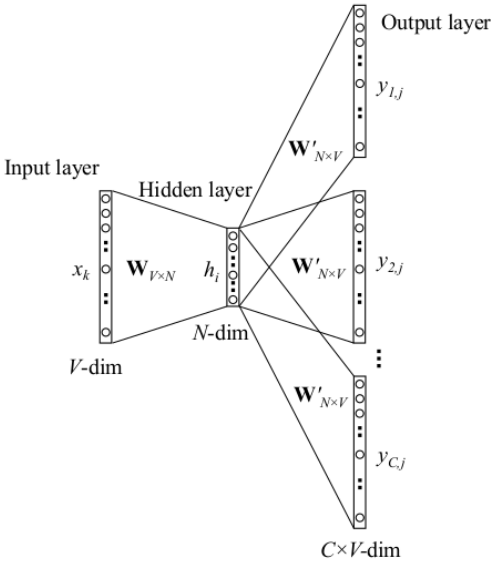


Figure 2: Skip Gram Model Architecture [3]

4.2 Intuition of word2vec

Unsupervised Learning: Both word2vec models are trained in an unsupervised fashion, i.e., from unlabelled data. Essentially we train our neural network to perform an artificially constructed classification task but are not going to use the trained neural network further. Instead we care about the hidden representations of the inputs which the neural network learns which are the word vectors we require. This method is normally applied in unsupervised feature learning. A major advantage of unsupervised learning is the creation of virtually unlimited data samples to train on.
Semantic Integrity: From the architecture and working of both the models, we can observe the role of the context window in maintaining similar word representations for words with similar meanings. If two words are semantically the same, for example 'happy' and 'joyful', they will be used in the same context. The model has to output similar predictions with these words as the input. This is possible if both the words have similar hidden representations (where similarity between vectors is defined using cosine similarity) in the weight matrix or equivalently similar word representations.

4.3 Unsupervised Learning in the Skip Gram Model

The skip gram model is a language modelling problem with the aim to get an estimate of the following conditional probability:

$$p(w_{-c}, w_{-c+1}, ..., w_{c-1}, w_c | w_t) \tag{1}$$

where c denotes the context window and w_t is the target word vector for the log-linear classifier. The objective function $\mathcal{J}'(\theta)$ for the skip gram model is to maximize the probability of occurrence of any context word given the target word w_t. The conditional probability distribution can be written as a negative log-likelihood function $\mathcal{J}(\theta)$ with its derivative as follows:

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

where T is the number of words in the sentence, $t = 1...T$ are the indices of the target words and θ represents all the trainable parameters. The loss function for the above probability distribution over the context words is given by the cross-entropy loss function.

Equation (1) can be reformulated for each target word w_{j+t} as:

$$p(w_{j+t} | w_t) = \frac{exp(w_{j+t}^T w_t)}{\sum_{\substack{-c \leq j \leq c \\ j \neq 0}} exp(w_{j+t}^T w_t)} \tag{2}$$

Equation (2) resembles the softmax classifier equation. We use softmax classifier when we have n output classes in the last layer of a neural network model. In the skip gram model we try to get the probability distribution for the c context words using the softmax classifier. We can view the softmax function as trying to minimize the cross-entropy between the

predicted and the true label vectors. The back-propagation equations for the skip gram model are written in terms of cross-entropy equations as follows:

$$\begin{aligned}\mathcal{L} &= -\log p(w_{-c}, w_{-c+1}, ..., w_{c-1}, w_c | w_t) \\ &= -\log \prod_{c=1}^{C} \frac{exp(w_{j+t}^T w_t)}{\sum_{\substack{-c \le j \le c \\ j \neq 0}} exp(w_{j+t}^T w_t)} \quad ... \text{ using (2)} \\ &= -\sum_{c=1}^{C} w_{j+t}^T w_t + C. \log \sum_{\substack{-c \le j \le c \\ j \neq 0}} exp(w_{j+t}^T w_t)\end{aligned}$$

For calculating the back-propagation error we take the gradient of $\mathcal{L}$ with respect to w_t and update the trainable variables in θ which includes the weights of the hidden and the output layer.

5 Implementation in Keras

We implemented the skip gram model of word2vec in Keras using Theano as the backend.

5.1 Generating Training Data Set

To generate training samples for the skip gram model we used the same approach as followed in the paper. After tokenizing our data set into words we formed a vocabulary over the words retaining the top 2000 words. We iterated over all words with x denoting the current word and randomly sampled words from its context window of size 10 with the sampled words denoted by y. Each random sampling from the context window formed a training sample (x, y) illustrating the advantage of unsupervised learning in producing multiple training samples. The context window consists of 5 previous words and 5 future words. We sampled 50% of the words from a context window to form training samples for the current word to prevent overfitting. If either x or y is not part of our vocabulary, we discard the word. The training samples are used in the form of one-hot encodings with dimension 50 where 50 is the dimensions of the word vectors finally formed also. The pseudocode of our algorithm to generate training samples is given in 1. An illustration of the training sample generation process is shown in fig 3.

5.2 Experimental Settings

We formed $200,000$ training samples from a manuscript of 100MB. The skip gram neural model architecture in Keras consists of an input layer, a linear hidden layer and a softmax layer with no bias in all layers. We initialized the weight matrices using truncated normal with mean as 0 and standard deviation as 0.005. We used stochastic gradient descent with a learning rate of 0.01 and momentum of 0.9. Since we use one-hot encodings to represent the output words with softmax at the output layer our loss function is categorical cross entropy. We ran this model for 1918 epochs with early stopping monitoring the validation loss. The

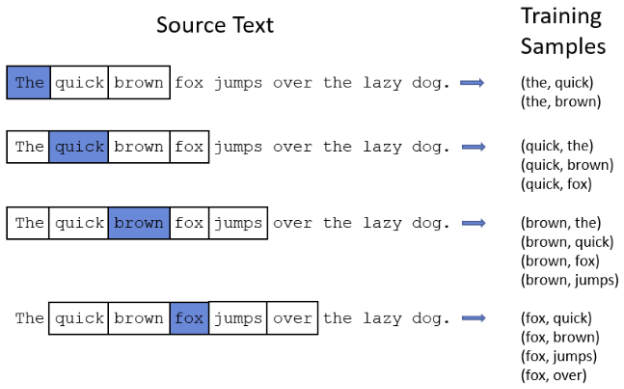


Figure 3: Generation of Training Samples[4]

Algorithm 1 Generate Training Data Set

procedure GENERATETRAINDATA($vocab\_size, vector\_dim, context, random\_percentage$)
    tokenize input data into word sequence S
    build vocabulary dictionary over words with max words as $vocab\_size$
    for word x in S **do**
        for repeat $random\_percentage * context$ times **do**
            randomly pick target word y in context neighbourhood of size $context$
            discard y from future picks
            store (x, y) as training sample

batch size was 256. We used the `get_weights` function of Keras to obtain the weights of matrix W. The ith row of W with N dimensions represents the word vector of the ith word in the vocabulary.

6 Visualization

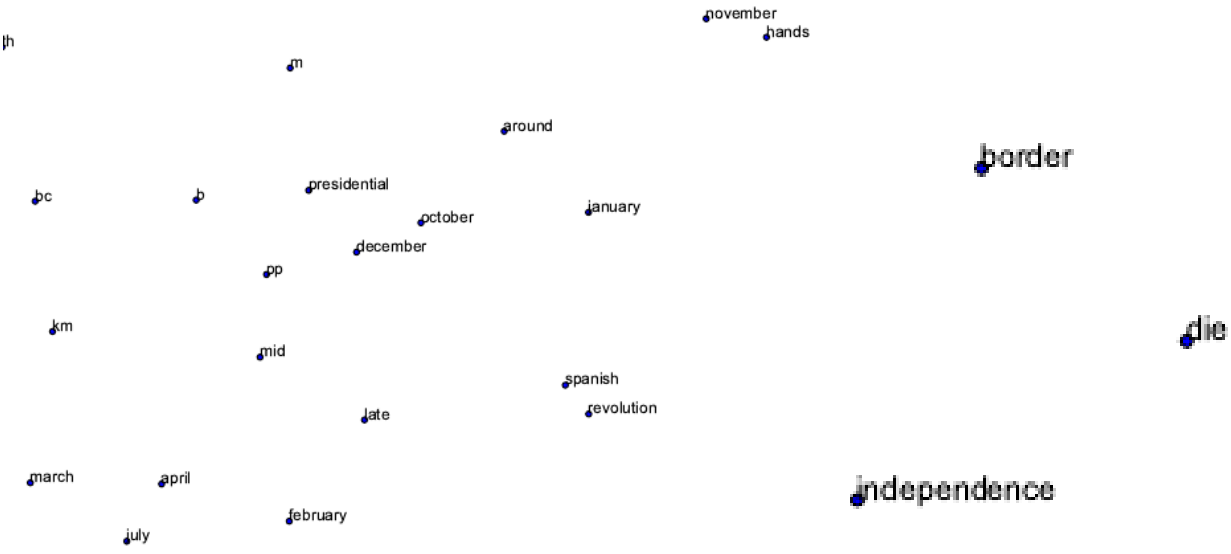
We visualize the word vectors of 50 dimensions learnt by the skip gram model using a t-Distributed Stochastic Neighbour Embedding (t-SNE) plot. As stated earlier, word2vec maintains the semantic relationships between words by incorporating context based learning with the assumption that similar words will occur in a similar context. The t-SNE plot over our vocabulary of 2000 words illustrates the formation of clusters of similar words which appear in the same context. We have shown such sample clusters in fig 4 and 5.

7 Conclusion

We have provided our understanding of the paper "Efficient Estimation of Word Representations in Vector Space" by Tomas Mikolov et al. [1]. We provide an implementation of the skip gram model of the paper along with visualizations to show the preservation of semantic relationships between words. As future work, we are reading about *paragraph vectors* from the paper "Distributed Representations of Sentences and Documents" [5]. In conclusion,

(a) Cluster of Numbers (b) Cluster of Seasons

Figure 4: Sample Clusters of Semantically Similar Words 1



(a) Cluster of Months (b) Cluster of Patriotic Words

Figure 5: Sample Clusters of Semantically Similar Words 2

word2vec forms high quality word representations from unlabelled data in a computationally fast manner.

Bibliography

[1] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *ICLR Workshop* (2013).

[2] Matt Mahoney. *Data Compression Programs*. URL: http://mattmahoney.net/dc/.

[3] Xin Rong. "Word2vec Parameter Learning Explained". In: *CoRR, abs/1411.2738* (2014). URL: http://arxiv.org/abs/1411.2738.

[4] Alex Minnaar. *Word2Vec Tutorial Part 1: The Skip-Gram Model*. URL: http://mccormickml.com/assets/word2vec/Alex_Minnaar_Word2Vec_Tutorial_Part_I_The_Skip-Gram_Model.pdf.

[5] Quoc Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *ICML* (2014).