Understanding and Explanation of Word2Vec

UIUC | CS 410: Text Information System | Fall 2022
Raghavendran Ramasubramanian (**RR26**)

Introduction:

In this technical review, I have explained how I comprehend the Word2Vec research paper by T. Mikolov and his colleagues. The research paper can be found in this location [1]. In many applications for natural language processing, words are encoded using a single-hot method, which does not take into account their interrelationships. "*simplicity, robustness, and the observation that simple models trained on huge amounts of data outperform complex systems trained on fewer data*" are the reasons that one-hot encoding was selected as the optimal method. [T. Mikolov et al.]

The N-gram model is an example of a relatively straightforward model. These models are called Markov models, and they are based on the assumption that the word in the i-th position depends on the history of words from the previous i-(n-1) words up until the word that comes before it (i-1). A frequency-based approach that was derived from the training corpus is basically what this is. However, these straightforward models call for high-quality training data, which is not always accessible, and they do not generalize particularly well to data that has not been seen before. However, as advancements have been made in machine learning, more complicated algorithms trained on much larger datasets have demonstrated superior performance to simpler models. Both syntactic and semantic similarities between the words are taken into account by the Word2vec model. One of the most famous illustrations of the application of vector algebraic on trained word2vec vectors is

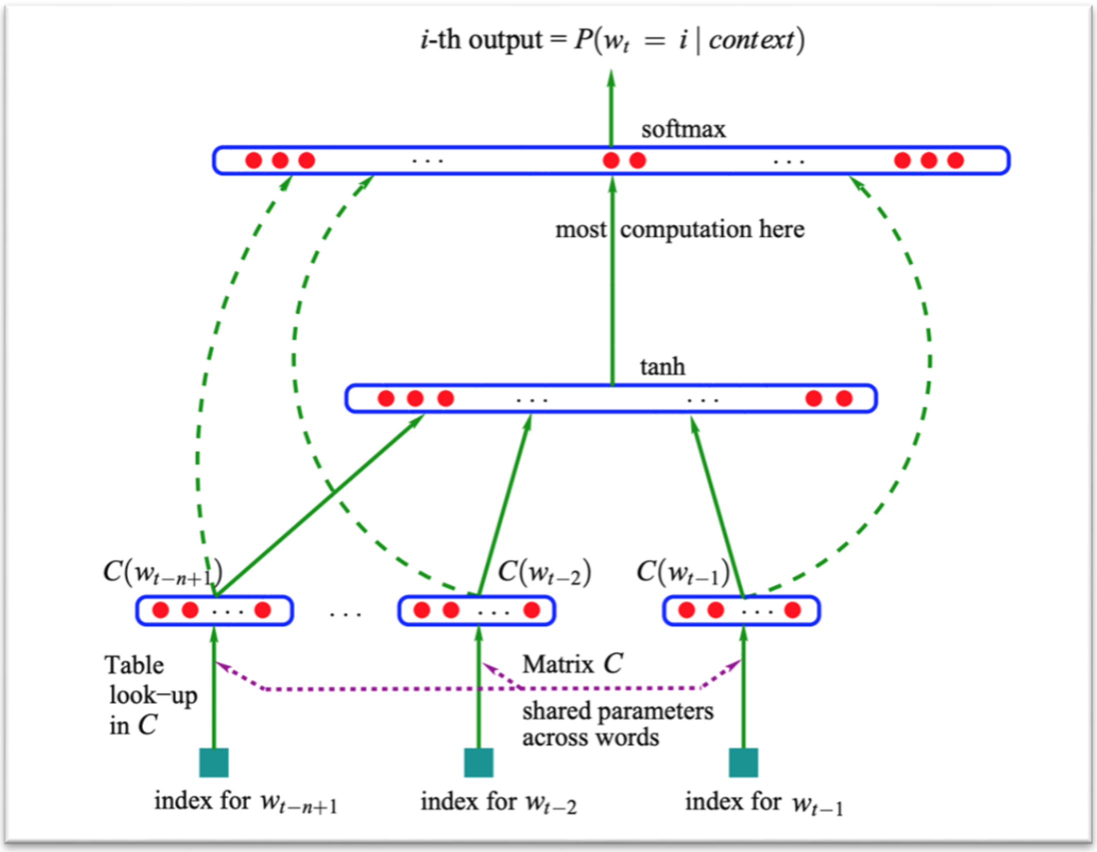
Vector(“King”)-Vector(“Man”)=Vector(“Queen”)-Vector(“Woman)

Models:

In this section, we'll look back at some of the methods before word2vec in the quest to represent words as vectors using neural networks and models.

1. Neural Network Language Model (NNLM)[2]:

An output layer follows a non-linear hidden layer in this two-stage language model. The first stage is a linear projection layer for word embeddings, and the second stage is an output layer. In the NNLM, the language model and the word vectors are simultaneously trained simultaneously.



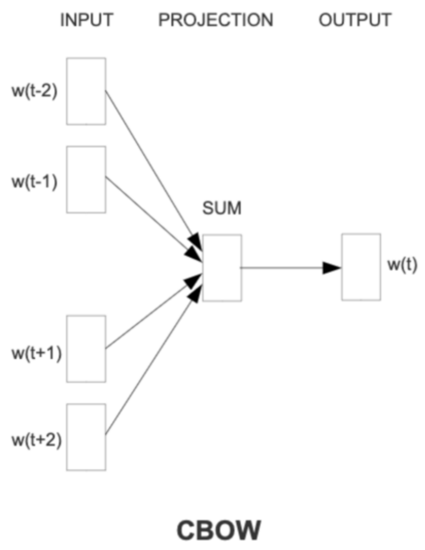
CBOW model architecture. Image from Word2Vec research paper. [Source]

The NNLM model has four layers: the input layer, the projection layer, the hidden layer, and the output layer. The value of the hyperparameter N determines the number of preceding words in the history of the current word that is considered when predicting the next term. Words are encoded using a one-hot encoding method with a vocabulary size of V in the input layer. The Matrix C is a shared projection matrix whose dimensions are N by D, with D standing for the embedding size. The projection layer and the hidden layer are densely connected, resulting in a weights matrix NxDxH from the projection layer to the hidden layer, where H is the size of the hidden layer. Following the generation of the hidden vector, the output layer contains weights denoted by the notation HV and then a softmax layer. Because of this, the total weights considered in NNLM are (NxD+NxDxH+HxV). By employing hierarchical softmax [3], in which the vocabulary is represented as a binary tree, it is possible to reduce the term (HxV) to Hxlog2(V). When ND is greater than V, the term (NxDxH) is responsible for most of the complexity. Because the word2vec model does not contain a hidden layer, this problem is circumvented and solved. During the training process for the NNLM model, the projection matrix serves as the lookup table for the trained word vectors.

2. Continuous bag of words model (CBOW):

The NNLM model is the basis for the CBOW model, with the primary difference being the absence of a linear hidden layer. The objective function of the CBOW model is to predict the middle word when given N/2 history words from the past and N/2 future words from the future.

When using N=8, the best possible outcome is achieved. Simply averaging the word vectors of N context words is done in the projection layer. Because the position of the word does not play a role in determining the word vector of the middle word, the structure is sometimes referred to as a "bag of words." The vector space denoted by the term Continuous is referred to as D.

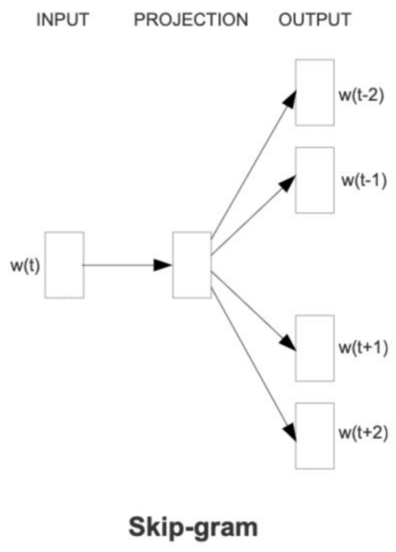


CBOW model architecture. The image is taken from a Word2Vec research paper. [Source]

To obtain a distribution over V, an averaged vector is sent to the output layer, where it is then followed by hierarchical softmax. The CBOW model is a straightforward log-linear one. The logarithm of the model's output can be represented as the linear combination of the weights associated with the model. NxD+Dxlog(2)V is the total number of weights involved in the training process for the CBOW model.

3. Continuous skip-gram model:

This model does the opposite of what the CBOW model intended to do. If you give it the current word, it will tell you what the nearby context words will be, both in the past and future. Therefore, the model is called a skip-gram because it predicts the N-gram words other than the current word, which serves as the input to the model. This is where the name comes from. The value of N is going to be 10, as decided upon. To generate output labels, fewer samples are taken from distant words than nearby words. This is because distant words are less relevant to the current word. When N equals 10, a random number R between 1 and 10 is produced using the sampling strategy described earlier, and R's history and future words are used as correct labels for the skip-gram model.



Skip-gram model architecture. Image from Word2Vec research paper. [Source]

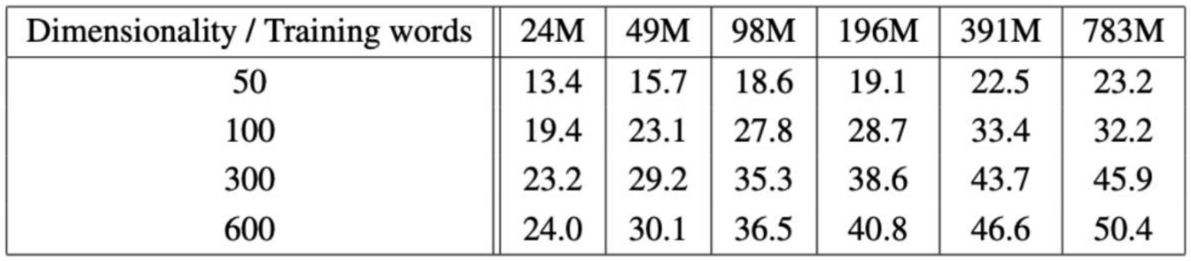
The overall complexity of the model can be expressed as NxD+NxDxlog2(V). It is important to note that N is multiplied by the Dxlog2(V) term, as this is not a single-class classification problem like CBOW but rather an N-class classification problem. As a result, the overall complexity of the skip-gram model is higher than that of the CBOW model.

Results:

This section discusses the results obtained by word2vec and NNLM models.

Test Set Preparation for Evaluation:

Word2vec model prepared a comprehensive test set with five semantic relations and nine syntactic relations between words, in contrast to earlier methods, which only displayed a small number of examples of tabular forms of words that are similar to one another. Currency: Country, City: State, Man for woman, and country for the capital city are some examples of how the words in the test set have a semantic relationship with one another. Syntactic relations can be found between antonyms, superlative degrees, present participles, past tenses of words, and the words themselves. The list of semantic and syntactic relations of the words included in the test set is used to compile the questions for the examination. A question along these lines might go something like this: "What is the word that is similar to small in the same sense that biggest is similar to big?" Big and biggest, as well as small and smallest, have a syntactic relationship similar to that of a superlative. The solution to the problem presented earlier can be derived using the vector algebraic expression X=Vector("biggest")-Vector("big")+Vector("small"). The answer that is predicted to be X is the word vector with a cosine distance similarity score closest to X. Only if the predicted output exactly matches the actual work does it qualify as a correct match. Regarding evaluation, synonyms of the production are still considered to be an incorrect match. On the test set, we make use of the accuracy metric.



Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

Table 1: Accuracy on a subset of the Semantic-Syntactic Word Relationship test set, using word vectors from the CBOW architecture with limited vocabulary.

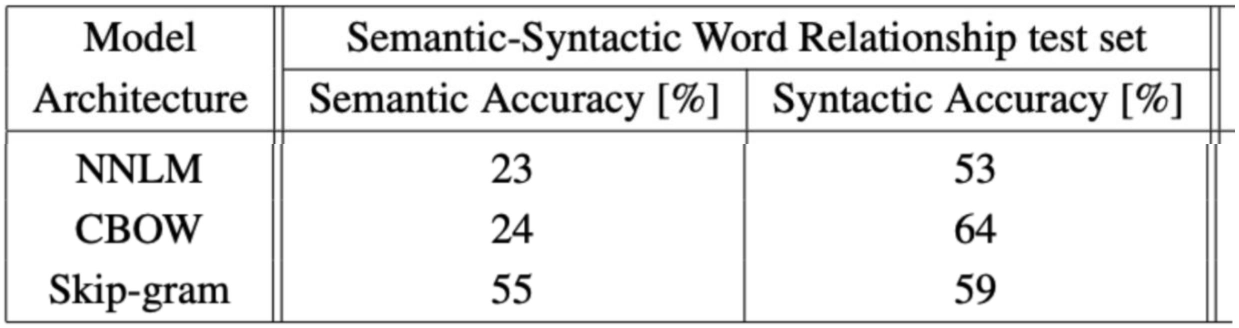
As can be seen in Table 1, the accuracy improves in both directions along with an increase in the dimensionality of the word vector and the size of the training corpus; however, there is a diminishing gain with an increase in the embedding size in comparison to the size of the training corpus.

Information from Google News Corpus containing 6B tokens in total. Because of the frequency, the vocabulary size is capped at 1 million. The number of epochs is **three**. The selected optimizer is **Stochastic gradient descent** with an initial learning rate of **0.025**, gradually decreasing linearly.

Model Accuracy vs. Architecture:

In this section, the training data and vector dimension are held at the same value, and the performance of various model architectures is compared.

Training criteria:
- Data - LDC corpora with 320M words, 82K vocabulary
- Vector dimension: 640



Model Architecture	Semantic-Syntactic Word Relationship test set	
	Semantic Accuracy [%]	Syntactic Accuracy [%]
NNLM	23	53
CBOW	24	64
Skip-gram	55	59

Table 2: Test set accuracy for different model architectures.

From table 2, CBOW is superior to the NNLM model because it produces better results on syntactic test sets while requiring significantly less time to train. Compared to CBOW, the Skip-gram model achieves better results on the semantic set with only a slight decrease in its performance on the syntactic side.

Large Scale Parallel Training:

Word2vec models have also used Jeffrey Dean's Distributed Belief System (DistBelief)[4] framework for large-scale parallel training of the models. As a result of the lower complexity of the word2vec model, models are trained on the enormous corpus by making use of DistBelief distributed training, which speeds up the training process. The NNLM, CBOW, and Skip-gram models were trained using the 6 billion tokens in the Google News corpus, with 100, 1000, and 1000-word vector dimensions, respectively. The length of time needed for training is proportional to the level of complexity of the model. A 100-dimensional word vector is used for the NNLM model rather than the 1000-dimensional word vectors used for the CBOW and Skip-gram models due to the high complexity of the NNLM model. Word vectors of CBOW and skip-gram, surprisingly, can be trained much more quickly than NNLM models. Within 2.5 days of training, Skip-gram reached its highest possible test accuracy of 65.6% by utilizing a mini-batch asynchronous gradient descent update and Adagrad optimizer while employing DistBelief.

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

Table 6: Comparison of models trained using the DistBelief distributed framework.

Summary:

It is possible to train Word2vec vectors using the CBOW or Skip-gram approach. They are trained on enormous corpora and with higher dimensional vectors to get a better continuous representation of the word vectors. This is possible because the models are less complex. When it comes to NLP tasks, word2vec models can be applied in various exciting ways. While developing neural network models, the vectors serve as features of the words. Word2vec can solve problems involving similarity between words and issues involving words that are not on the list.

Reference:

[1] Efficient Estimation of Word Representations in Vector Space, arXiv:1301.3781v3 [cs.CL] 7 Sep 2013 - https://arxiv.org/pdf/1301.3781.pdf
[2] A Neural Probabilistic Language Model [Bengio, Yoshua, et al.] - https://www.jmlr.org/papers/volume3/tmp/bengio03a.pdf
[3] Notes from Seminar by Benjamin - https://building-babylon.net/2017/08/01/hierarchical-softmax
[4] Large Scale Distributed Deep Networks [Jeffrey Dean] - https://storage.googleapis.com/pub-tools-public-publication-data/pdf/40565.pdf