Technology Review

Author: Liping Xie

ABSTRACT

This technical review is focusing on comparing Word2Vec and FastText.

INTRODUCTION

According to Wikipedia, Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers. Word2Vec and FastText are the popular word embedding models.

Word2vec utilizes continuous bag-of-words (CBOW) or continuous skip-gram to produce a distributed representation of words. FastText is an extension of word2vec, it utilizes continuous bag-of-words (CBOW). The improvement of FastText is the inclusion of character n-grams which supports the computing word representations for “out-of-vocabulary” words (OOV).

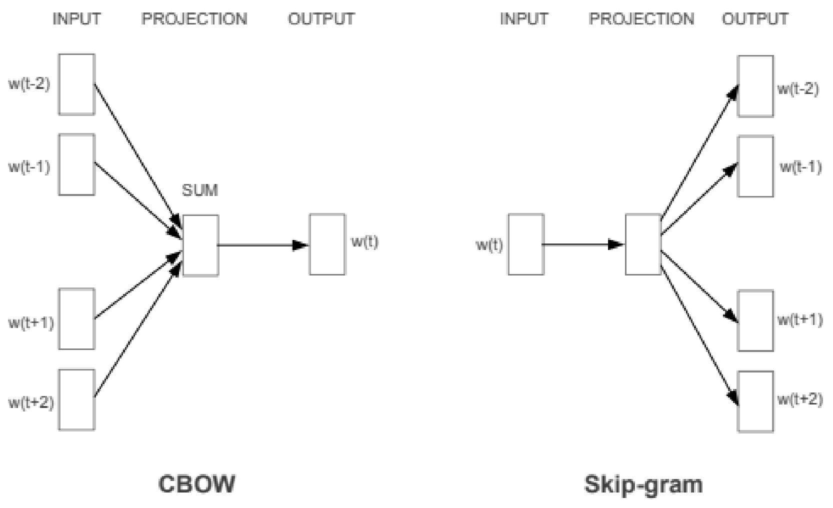
This review is mainly for comparing these two models based on model architecture, performance, applications, advantages and disadvantages.

REVIEW DETAILS

Model Architecture

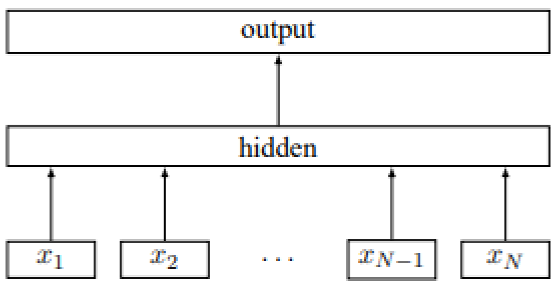
Word2Vec

Word2Vec utilizes Continuous Bag-of-Words Model (CBOW) or Continuous Skip-gram Model. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

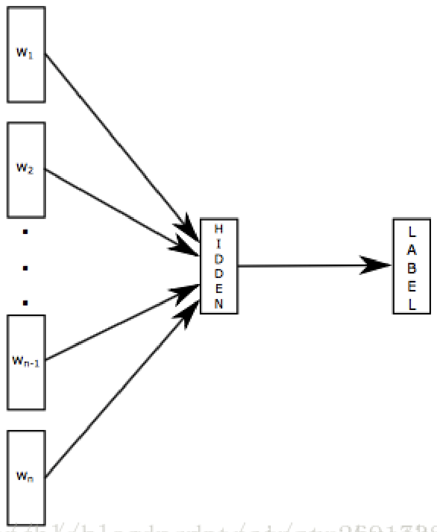


FastText

FastText’s architecture is similar to the CBOW model, where the middle word is replaced by a label.



Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.



FastText uses bag of n-grams as additional features to capture some partial information about the local word order. N-grams feature sums all n-grams in a word equal to the representation of that word. With this, unknown or rare words still be represented by the sum of the character representation when the related word representation is not available. For example, smile, smiling, and smilling may have similar meanings when using FastText, but in Word2Vec, the error in "smilling" cannot be removed automatically and will treat it as an unknow word.

Performance

Accuracy

Based on the test done by Frederic Godin using Twitter microposts (Ritter et al., 2011; Gimpel et al., 2011; Derczynski et al., 2013; Owoputi et al., 2013). FastText and Word2Vec have their own strength:

- FastText performs better than the Word2Vec as input vectors for PoS Tagging.

Algorithm	**Test set**	
	Ritter *et al.*	**Gimpel *et al.***
Ritter *et al.* (2011)	84.55	/
Derczynski *et al.* (2013)	88.69	/
Gimpel *et al.* (2011)	/	88.89
Owoputi *et al.* (2013)	**90.40**	**91.60**
CNN - no word embeddings	74.29	75.13
CNN - Word2vec embeddings	88.83	90.60
CNN - FastText embeddings	**90.53**	**91.74**

- FastText outperforms Word2Vec on Out-Of-Vocabulary (OOV) words, but Word2Vec performs slightly better in In-Vocabulary (InV) words.

Dataset	**Word type**	**Word embedding**	
		Word2vec	**FastText**
Ritter *et al.*	OOV	59.06	78.74
	InV	91.30	91.11
	All	89.47	90.41
Gimpel *et al.*	OOV	39.47	64.74
	InV	92.57	92.46
	All	90.48	91.37

- Word2Vec performs better than FastText as input vectors for a CNN+CRF architecture for NER.

Model	F1	
	Validation	Test
Baseline	34.29	31.97
No word embeddings	17.48	17.56
Word2Vec embeddings	**54.11**	**48.78**
FastText embeddings	50.2	46.75

Based on the Sentiment Analysis done by Jatin Mandav using Twitter Data for Word2Vec and FastText, both of them achieved approximately 69% accuracy.

According to a lot of users' practices with small datasets of different languages, FastText provides better result in Text Classification tasks.

FastText's n-grams feature may cause overfitting. That's because N-grams feature sums all n-grams in a word equal to the representation of that word, the word vector representation is based on the related text contents. When we use these word vectors for the new data, it may not fit and will cause overfitting. This problem can be improved with proper parameter during the training.

Training Time

Since both FastText and Word2Vec are using Hierarchical softmax for computation, this linear computation model that outperforms complex deep learning models can train a billion words in a few minutes on a standard multi-core CPU. But when includes n-gram in training FastText model, the input vectors are increased for the same set of data, the training time is longer than Word2Vec.

Applications

NLP Tasks

Both FastText and Word2Vec can be used for Text Classification and Sentiment Analysis, they have similar performance on the large dataset. Since they use linear model for computation, both of them can handle simple Text Classification task very well, however, it does not consider the information behind the word orders, in the tasks which are sensitive to the word orders (eg. Sentiment Analysis), their performance is not as good as non-linear models. For example, for the following sentences:

"The movie is not very good, but I still like it. "
"The movie is very good, but I still do not like it."
"I do not like it, but the movie is still very good."

These sentences are following similar patten and have similar words, but their meanings are quite different. If we use n-grams to calculate the vector values, they are likely be labeled as the same type. To solve this problem, we may need to use more complex non-linear models.

With the n-grams feature, FastText performs better in finding the similar words. And in the systems that with large number for classes and dataset, engineers are tend to prefer FastText for fast training and fast prediction purpose.

Language Support

FastText provides pre-trained models for 157 different languages which are trained on Common Crawl and Wikipedia using fastText. User just need to download them directly with command line or from python.

For Word2Vec, Google provides pre-trained model for English. It includes word vectors for a vocabulary of 3 million words and phrases that they trained on roughly 100 billion words from a Google News dataset. The vector length is 300 features. The model can be loaded with gensim and pass in the path to the model file. Pre-trained model for other languages can be obtained from different companies and individual users.

Advantages and Disadvantages

Word2Vec

Advantages:

- It is computationally efficient.
- The library is very fast and requires little memory.
- It can handle simple text classification task very well in a large corpus.
- It is easy to understand and perform the implementation.
- The size of the embedding vector is very small.

Disadvantages:

- The word vector representations are based on the surrounding words, it doesn't benefit from the information in the whole document.
- The sub-word information is not captured.
- It cannot handle Out Of Vocabulary words (OOV), OOV words will be assigned random values.
- It does not produce multiple vectors for each word depending on the context which impacting the accuracy for Sentiment Analysis.

FastText

Advantages:

- Comparing to other methods for achieving the same accuracy, the library is very fast.
- Sentence Vectors (Supervised) can be computed easily.
- For the small datasets, it provides better accuracy and Sentiment Analysis result than Word2Vec.
- It is easy and fast to be trained

- With n-grams feature, Out Of Vocabulary words vector can be built with the average vector representation of its n-grams.
- It provides pre-trained models for 157 different languages.

Disadvantages:

- FastText is not a standalone library and requires another library for the pre-processing.
- It provides a python implementation which is not officially supported.
- This model is based on prediction but not on statistics over the corpus.
- It does not produce multiple vectors for each word depending on the context which impacting the accuracy for Sentiment Analysis.
- It may cause overfitting due to the n-grams features

CONCLUSIONS

The computation model for FastText and Word2Vec are very similar, they have similar accuracy in Text Classification tasks, similar training time and similar limitations on Sentiment Analysis. FastText doesn't outperforms Word2Vec in all areas, but its inclusion with n-grams feature allows it to support OOV much better than Word2Vec which is very important when user would like to train their own model with their dataset. Since FastText provides pre-trained models for different languages, it is more convenient to use. FastText provides slightly better user experience on computation speed. When work on the large corpus, Word2Vec and FastText provide similar performance on Text Classification Tasks. But when work on small corpus, FastText works better.

CITED:

Efficient Estimation of Word Representations in Vector Space: https://arxiv.org/pdf/1301.3781.pdf

Bag of Tricks for Efficient Text Classification: https://arxiv.org/pdf/1607.01759.pdf

Enriching Word Vectors with Subword Information: https://arxiv.org/pdf/1607.04606.pdf

https://fredericgodin.com/research/twitter-word-embeddings/

https://www.linkedin.com/pulse/sentiment-analysis-using-word2vec-fasttext-universal-sentence-mandav