Word2vec

Seminar 1 - Barbora Stepankova

1.) What does it do?

Word2Vec is an algorithm that accepts text corpus as an input and outputs a high-dimensional numeric vector representation for each word. [2]

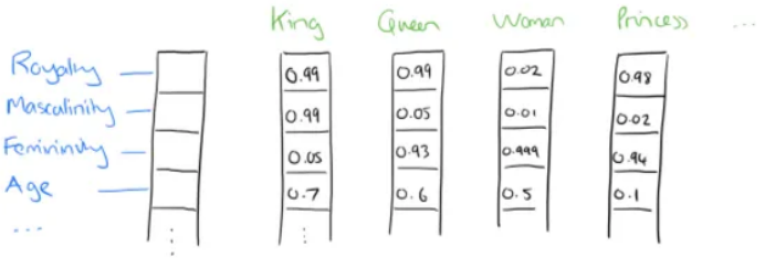
Word embedding, distributed representations of words in a corpus, captures both syntax and semantics.

Learns relations from the input text.

Distances between word vectors can be easily measured by cosine similarity.

The most famous example:
Vector(“King”) - Vector(“Man”) + Vector(“Woman) = Vector(“Queen”)

If I label the dimensions in a hypothetical word vector (there are no such pre-assigned labels in the algorithm of course), it might look a bit like this:



[2]

2.) Motivation behind it

The goal was to make training language models faster and more effective, by making a simple and fast word embedding model. Word2vec prepares word vectors for further work.

Previously both NNLM and RNNLM had most of their complexity from the non-linear hidden layer, which basically translated words into something the computer could understand.

3.) How does it work?

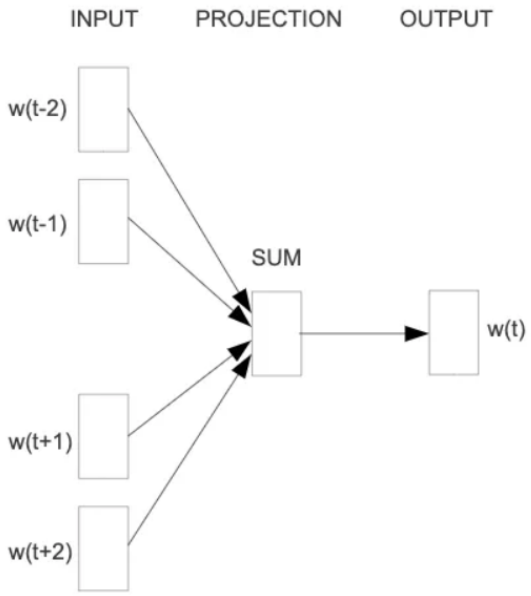
It is a shallow neural network, with only input, projection and output layers, without any hidden layer.

There are two algorithms to train word2vec model:

- CBOW
- Continuous skip-gram

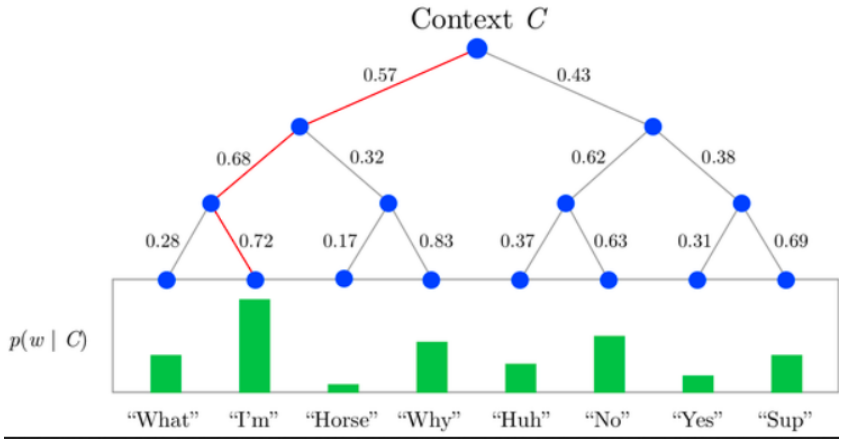
CBOW

- Continuous bag of words



[1]

- There is a sliding window, and the middle word is being predicted from the surrounding words, without dependency on the order of the surrounding words (therefore continuous BOW)
- An averaged vector of the surrounding words is passed to the output layer followed by hierarchical softmax to get distribution over vocabulary
    - *Hierarchical Softmax is an alternative to softmax that is faster to evaluate: it takes O(log(n)) time to evaluate compared to O(n) for softmax. It utilises a multi-layer binary tree (Huffman tree), where the probability of a word is calculated through the product of probabilities on each edge on the path to that node.* [5]

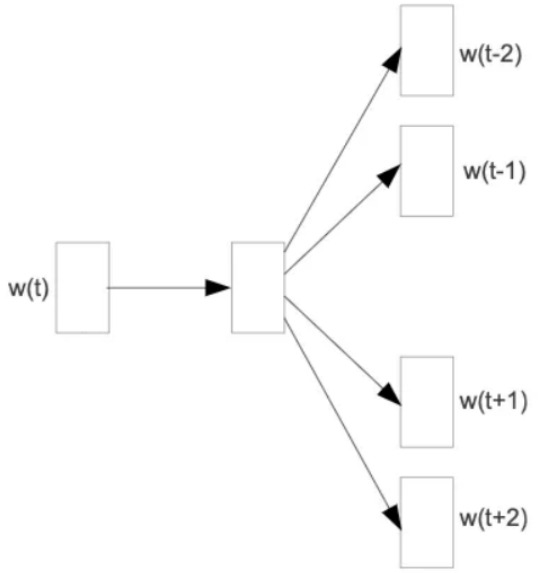


- [5]

Skip-gram

- Given the middle word, it predicts the surroundings from history and the future.
- The surrounding words are weighted by their closeness to the middle word. The closer to the middle word, the greater the importance.

- The bigger the window, the better results, but at a big computational cost.



[1]

4.) Impact

Word2vec made many tasks much easier, for example:
- Word prediction for added variability
- Clustering, synonyms
- Sentiment classification - positive/negative
- POS tagging

It also brings to light strong biases in texts regarding stereotypes. If the training data supplied for training contains these biases (which it almost surely does), we can really easily see how different stereotypes are clearly backed up by the math. According to word embeddings trained on large amounts of data, *Man is to Computer Programmer as Woman is to*
Homemaker [6]

5.) Difficulties

Dealing with out-of-vocabulary words
Vectors are independent between words
- No sub-words

Is outdated, outperformed by transformers, but still very useful when speed and simplicity are needed.

BERT distinguishes between the words as different parts of a sentence.
Meaning that BERT is context-sensitive, and word2vec is static.

"*Word2vec is **trained** using CONTEXTUAL neighbours but **used NON-CONTEXTUALLY** for a downstream NLP task. As only a single-vector-per-word is saved as the representation*" [3]

'river *bank* 'and '*bank* deposit' - word2vec has just one representation for 'bank' not dependent on the context, BERT can have several.

6.) Tomas Mikolov and his research team as part of Google Brain

Tomas Mikolov wanted to persuade people that researching language models is the way to go and that it is the future.

Google Brain was a deep learning artificial intelligence research team under the umbrella of Google AI, a research division at Google dedicated to artificial intelligence.

7.) Used data

For training our own word2vec, we can use any data we have. The data don't even have to be textual, it can be used to vectorize chemical compounds or gene sequences!! [4]

Originally it was trained on the Google News Corpus with 6B tokens. Vocabulary size was limited to 1M based on the frequency.

8.) Implementation in Python:

For example gensim:
https://radimrehurek.com/gensim/models/word2vec.html?source=post_page-----205cb7eecc30----------------------------

9.) References

[1] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

[2] https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1

[3] https://medium.com/@ankiit/word2vec-vs-bert-d04ab3ade4c9

[4] https://serokell.io/blog/word2vec

[5] https://paperswithcode.com/method/hierarchical-softmax

[6] Bolukbasi, Tolga, et al. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." *Advances in neural information processing systems* 29 (2016).