Distributed Representations for Natural Language Processing

Structure of this talk

- Why
- Word2vec
  - Architecture
  - Evaluation
  - Examples
- Discussion

Why?

Representation of text is very important for performance of many real-world applications: search, ads recommendation, ranking, spam filtering, …

- Local representations
  - N-grams
  - 1-of-N coding
  - Bag-of-words
- Continuous representations
  - **Distributed Representations**
  - Other

Distributed representations

- We hope to learn such representations so that Prague, Rome, Berlin, Paris etc. will be close to each other

- We do not want just to cluster words: we seek representations that can capture multiple degrees of similarity: Prague is similar to Berlin in some way, and to Czech Republic in another way

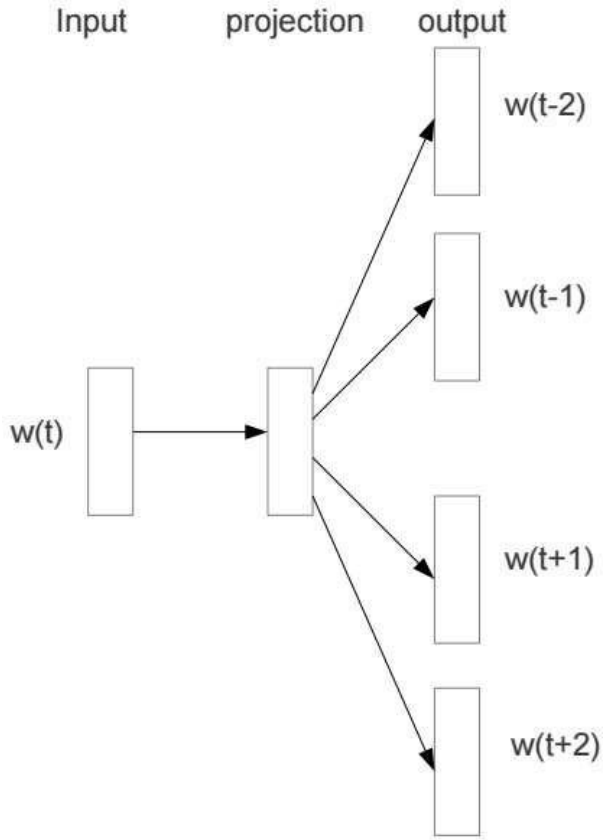
- Can this be even done without manually created databases like Wordnet / Knowledge graphs?

Word2vec

Two basic architectures:

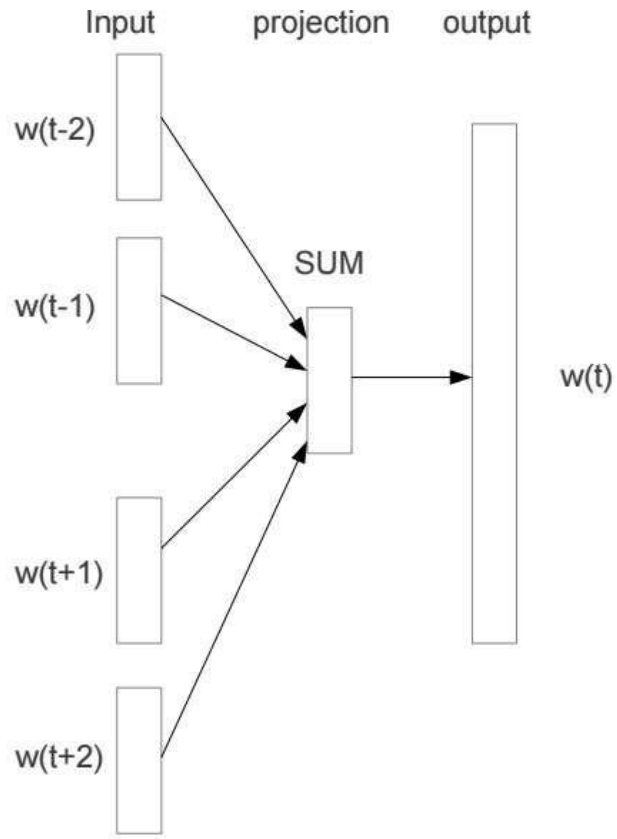
- Skip-gram
- CBOW

Skip-gram Architecture



- Predicts the surrounding words given the current word

Continuous Bag-of-words Architecture



- Predicts the current word by the given context

Word2vec

The user word2vec has the ability to switch and choose between algorithms.

The word order of the context does not affect the result in any of these algorithms.

The resulting coordinate representations of word vectors allow us to calculate the “semantic distance” between words.
And, precisely based on the contextual proximity of these words, the word2vec technology makes its predictions.

To achieve its most effective work, it is necessary to use large buildings for its training.
=> This will improves the quality of predictions.

Summary

- Word2vec: much faster and way more accurate than previous neural net based solutions - speed up of training compared to prior state of art is more than 10 000 times
- Features derived from word2vec are now used across all big IT companies in plenty of applications (search, ads, ..)
- Very popular also in research community: simple way how to boost performance in many NLP tasks
- Main reasons of success: very fast, open-source, easy to use the resulting features to boost many applications (even non-NLP)

Final notes

- Word2vec is successful because it is simple
- For modeling sequences of words
- Do not sum word vectors to obtain representations of sentences