Word embeddings as they are

Ivan Provilkov

MIPT DIHT

iv.provilkov@gmail.com

14 октября 2018 г.

Motivation

Сегодня нет способа хорошо интерпретировать то, как работают нейронные сети. Полезно уметь визуализировать результаты работы алгоритма, это помогает лучше понять, что он делает, и придумать улучшения.

Course organization

- Вначале лекции на повторение основных идей NLP.
- Затем разбор и реализация нескольких методов визуализации в репозитории на github.

Course Organization

Будет хорошо, если каждый подумает, что он хочет от курса.
Свои пожелания можно присылать мне на почту.

Word embeddings

- Task
- General idea
- Objective function
- Softmax
- Training method
- Architecture

What do we want?

We want to make vectors that represent word meaning, because we can use them in our algorithms. It's a critical component of everything from web search and content ranking to spam filtering, and when it works well, it's completely invisible to you.

Of course we can use one hot vectors, but there are some problems.

- Their is no natural similarity between them.
- These vectors do not contain meaning itself.

General idea

Core idea: A word's meaning is given by the words that frequently appear close-by.

One night, lightning struck the oak tree.

Pairs for lightning with window size 4:
(lightning, night), (lightning, one), (lightning, struck), (lightning, the)

Question Should we make pair with ',' ?

Word vectors (representations, embeddings)

So, we want to build dense vectors for each word, trying to encode meaning (all possible contexts) to it.

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Question What does dense mean?

Question How we can do such task using Statistics?

Word2Vec

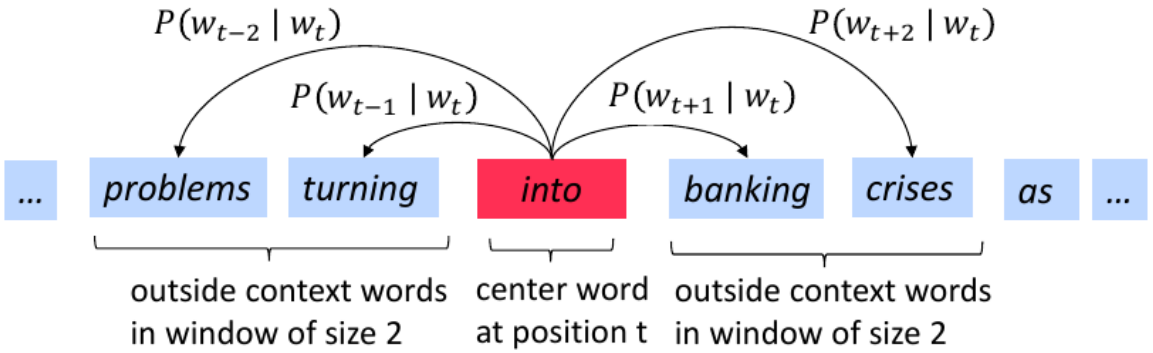
Word2vec (Mikolov et al. 2013) is a framework for learning word vectors.

What do we do?

- We take a large corpus of texts.
- Go through each word W in this text and its context C.
- Use the similarity function between W and C to calculate probability of W given C (or C given W).
- Keep adjusting word vectors to maximize this probability. For example using neural networks.

More closely

Example of computing $P(w_{t+j}|w_t)$.



What objective do we want to optimize?

We want to maximize probability for each word to appear in observed context. In other words we want to maximize probability of this sequence of words to meet together. In Statistics it is called likelihood.

$$L(\theta) = ?$$

Question What do you think we can paste here?

What objective do we want to optimize?

We want to maximize probability for each word to appear in observed context. In other words we want to maximize probability of this sequence of words to meet together. In Statistics it is called likelihood.

$$L(\theta) = \prod_{t=1}^{T} \prod_{-m \leq i \leq m, i \neq 0} P(w_{t+j} | w_t, \theta)$$

Question How can we simplify it for the computational reasons?

What objective we want to optimize?

Consequently, our loss:

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq i\leq m, i\neq 0}\log P(w_{t+j}|w_t, \theta)$$

Now we can minimize this function via optimization methods and at the same time maximize likelihood.
Question Do you understand it?
Question How can we approximate P?

Softmax

We can say that

$$P(c|w) = \text{softmax}(v_c \cdot v_w) = \frac{\exp(v_c \cdot v_w)}{\sum_{i=1}^{N} \exp(v_i \cdot v_w)}$$

Softmax:
"max"because amplifies probability of largest c_i,
"soft"because still assigns some probability to smaller c_j.

Training

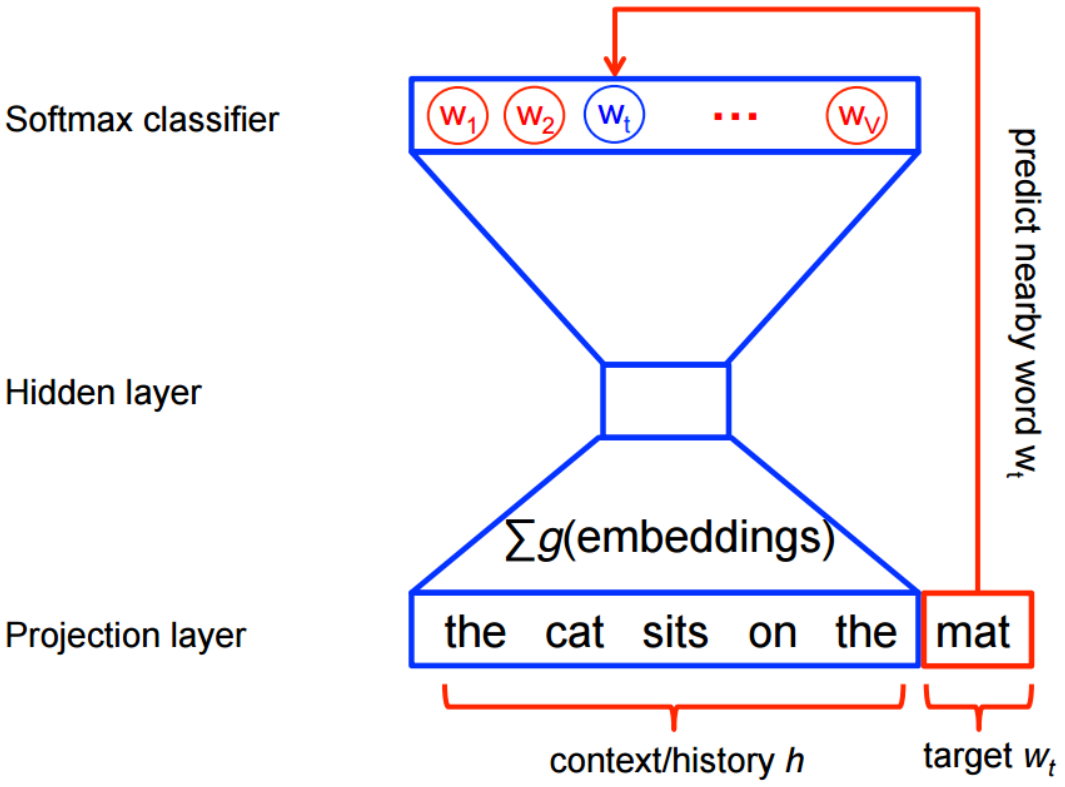
We want the model to be successful at discriminating real words from noise words.
Our binary classification objective(logistic regression):

$$J_{neg} = \log Q_\theta(D = 1|\text{context word}, \text{middle word}) + \log Q_\theta(D = 0|\text{noise word}, \text{middle word}). \quad (1)$$

Where Q_θ is the binary logistic regression probability.

This objective is maximized when the model assigns high probabilities to real words, and low probabilities to noise words.

Neural network



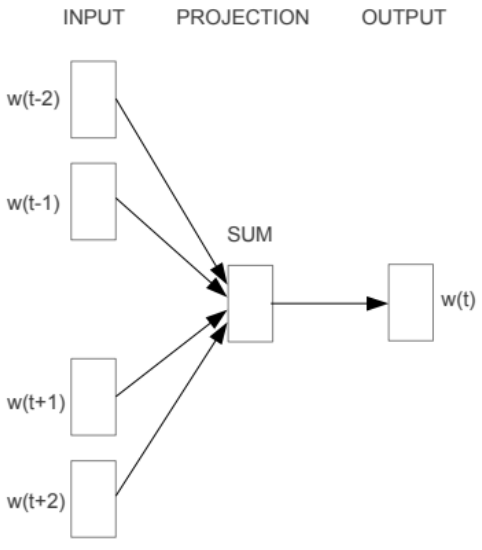
Neural network

What do we need?

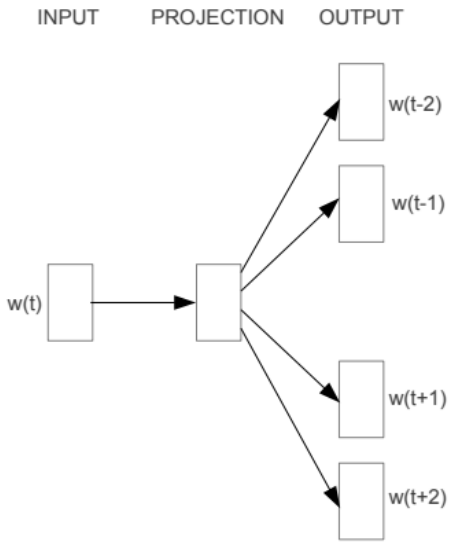
- Embeddings([vocabulary size, embedding size])
- NN weights W([vocabulary size, embedding size]), bias([vocabulary size])

Question What should we do?

Different approaches



CBOW



Skip-gram

More complex approaches

- **Glove** - mixture of statistical modeling and Word2Vec.
- **Fasttext** assumes a word to be formed by a n-grams of character. Benefits: It can give the vector representations for the words not present in the dictionary or rare words. And character n-grams embeddings tend to perform superior to word2vec and glove on smaller datasets. Also it uses Hierarchical softmax and trains faster.

The End