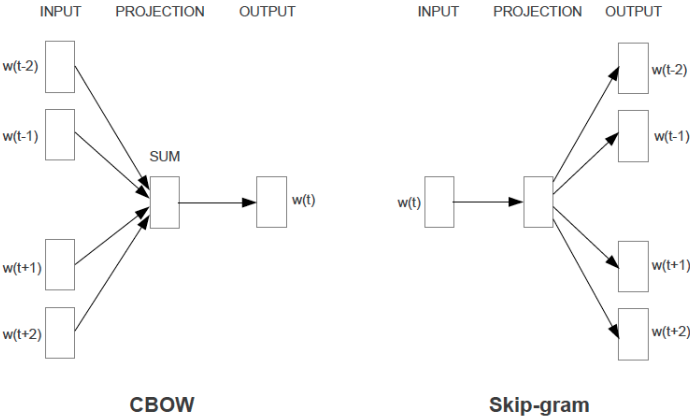
Learning Word Vectors: Overview

Basic Idea

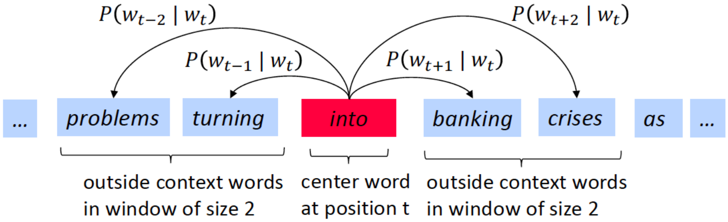
- We have a large corpus of text
- Every word in a fixed vocabulary is represented by a *vector*
- Go through each position t in the text, which has a center word c and context ("outside") words o
- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
- Keep adjusting the word vectors to maximize this probability

Two Variations: CBOW and Skip-grams

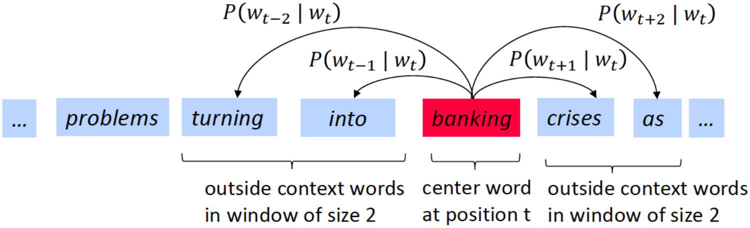


Word2Vec (Skip-gram) Overview

Example windows and process for computing $P(w_{t+j}|w_t)$



Example windows and process for computing $P(w_{t+j}|w_t)$



Word2Vec: objective function

For each position $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_j.

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j} \mid w_t; \theta)$$

θ is all variables to be optimized

sometimes called *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Minimizing objective function $\Leftrightarrow$ Maximizing predictive accuracy

Word2Vec: objective function

We want to minimize the objective function:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\ne 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

How to calculate $P(w_{t+j}|w_t;\theta)$?

We will use two vectors per word w:

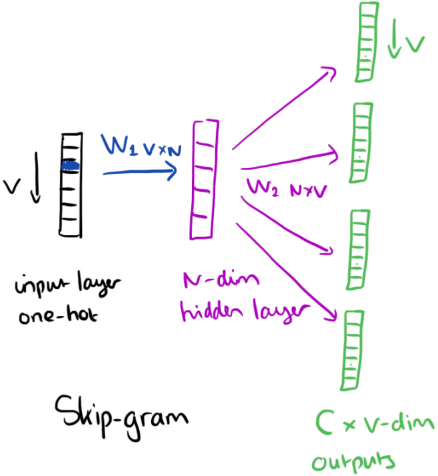
- v_w when w is a center word
- u_w when w is a context word

Then, for a center word c and a context word o

$$P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w\in V} exp(u_w^T v_c)}$$

Understanding $P(o|c)$ further

$$P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w \in V} exp(u_w^T v_c)}$$



Try this problem

Skip-gram

Suppose you are computing the word vectors using Skip-gram architecture. You have 5 words in your vocabulary, $\{passed, through, relu, activation, function\}$ in that order and suppose you have the window, '*through relu activation*' in your corpora. You use this window with 'relu' as the center word and one word before and after the center word as your context.

Compute the loss

Also, suppose that for each word, you have 2-dim in and out vectors, which have the same value at this point given by [1,-1],[1,1],[-2,1],[0,1],[1,0] for the 5 words, respectively. As per the Skip-gram architecture, the loss corresponding to the target word "activation" would be $-log(x)$. What is the value of x?

Gradient Descent for Parameter Updates

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

Training the model

Compute **all** vector gradients.

- θ represents all model parameters: in our case, V-many words, 2 $d-$ dimensional vectors for each word
- We optimize these parameters by walking down the gradient

Training the model

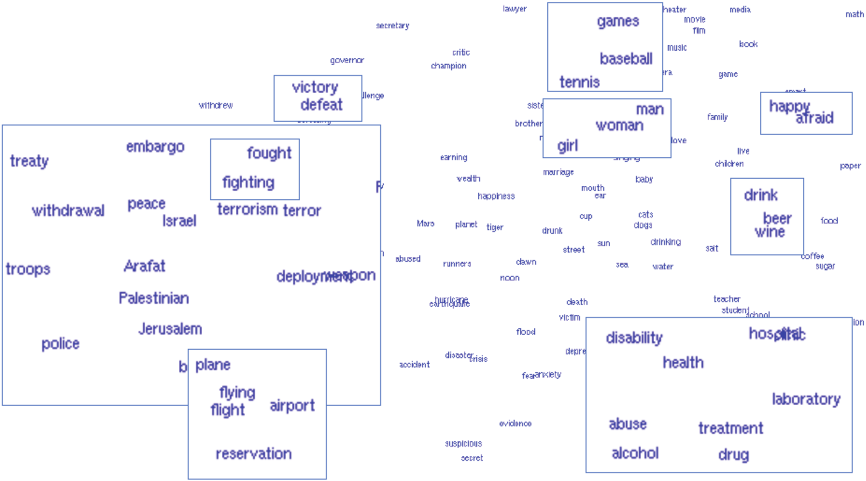
$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

For *one example window* and *one example outside word*:

$$log\ p(o|c) = log \frac{exp(u_o^T v_c)}{\sum_{w \in V} exp(u_w^T v_c)}$$

Try deriving the gradients for the center word v_c and the outside word u_o.

Visualization



Issues with word2Vec as we discussed

Computational Overhead

$$P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w \in V} exp(u_w^T v_c)}$$

Solutions

- Negative Sampling
- Hierarchical Softmax

Both the solutions optimize a different objective function!

Negative Sampling: Formulation

- Consider a pair (w, c) of word and context. *Did this pair come from the training data?*
- Let $P(D = 1|w, c)$ denote the probability that (w, c) came from the corpus data.
- $P(D = 0|w, c)$ will be the probability that it did not come.
- $P(D = 1|w, c)$ is denoted with the sigmoid function

$$P(D = 1|w, c, \theta) = \sigma(v_c^T v_w) = \frac{1}{1 + e^{(-v_c^T v_w)}}$$

Negative Sampling: Objective Function

- maximize the probability of a word and context being in the corpus data if it indeed is, and
- maximize the probability of a word and context not being in the corpus data if it indeed is not

$$\begin{aligned}
\theta &= \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} P(D=0|w,c,\theta) \\
&= \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} (1-P(D=1|w,c,\theta)) \\
&= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log P(D=1|w,c,\theta) + \sum_{(w,c)\in \tilde{D}} \log(1-P(D=1|w,c,\theta)) \\
&= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} + \sum_{(w,c)\in \tilde{D}} \log(1-\frac{1}{1+\exp(-u_w^T v_c)}) \\
&= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} + \sum_{(w,c)\in \tilde{D}} \log(\frac{1}{1+\exp(u_w^T v_c)})
\end{aligned}$$

Negative Sampling: Objective Function

Minimizing the negative log likelihood

$$J = -\sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} - \sum_{(w,c)\in \tilde{D}} \log(\frac{1}{1+\exp(u_w^T v_c)})$$

$\tilde{D}$ is a “false” or “nagative” corpus. We can generate $\tilde{D}$ on the fly by randomly sampling from the word bank

$$-\log \sigma(u_c^T \cdot \hat{v}) - \sum_{k=1}^{K} \log \sigma(-\tilde{u}_k^T \cdot \hat{v})$$

Here $\{\tilde{u}_k | k = 1, \ldots, K\}$ are sampled from $P_n(w)$, where $P_n(w)$ is unigram model raised to the power $3/4$. Some intuition:

$$\text{is: } 0.9^{3/4} = 0.92$$
$$\text{Constitution: } 0.09^{3/4} = 0.16$$
$$\text{bombastic: } 0.01^{3/4} = 0.032$$