Project Summary

Ben Picker

Part 1: Background word2vec

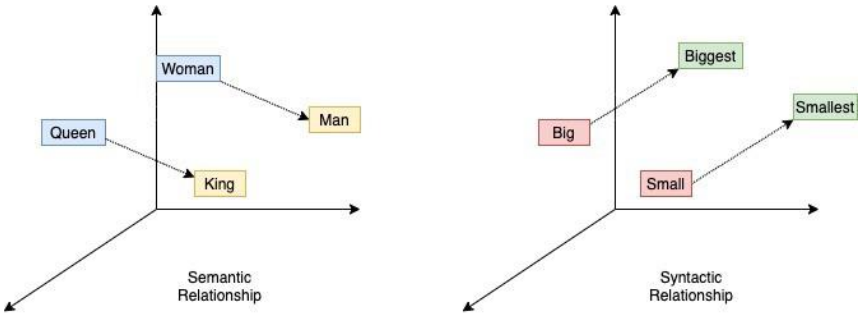
1.1. Introduction

In this project, I will attempt to build a basic word2vec model from scratch.

Suppose we wish to be able to compare words in terms of their meaning and we wish to have an intuitive notion of "similarity." For instance, the word *mankind* should be closer to the word *women* than *elephant* because, generally speaking, when people use the word *mankind*, it refers to all of humanity, including *women*. By contrast, an *elephant* has very little to do with *women* by comparison. Thus, we wish to capture this notion of distance in a similarity space and we plan to use vectors to do so.

We also would like to capture the notion that using "addition" can give us intuition about combining meanings of words. A classic example is we would like

$$\text{king} + \text{woman} - \text{man} = \text{queen}$$



Source:[1]

Intuitively, a *king* is the top royal *man*, so if we replace *man* with *woman*, we get top royal *woman*, which would be the *queen*. Thus, we would like to have our representation of words permit comparisons of this nature.

[1] https://towardsdatascience.com/word2vec-research-paper-explained-205cb7eecc30

Why we want vectors

Natural language is a complex system of meanings and we need a way to encode words so that some of that meaning is captured. Suppose our vocabulary is a set of words $\mathcal{V} = \{w_1, \dots, w_{|\mathcal{V}|}\}$ and the corresponding size is $|\mathcal{V}|$. These are literally just strings and we may have an index, $\mathcal{V}_{\text{idx}} = \{0,1, \dots, |\mathcal{V}| - 1\}$ such that we can have a look up table

Index	Word
0	"cat"
1	"ace"
⋮	⋮
$\lvert\mathcal{V}\rvert - 1$	"house"

How can we represent words as vectors?

One possibility is one-hot coding. A *one-hot coded vector* is a vector where each row (or column) represents a possible word in the vocabulary. If it has a 1, then it represents that particular word. If it is 0, then it doesn't represent that word. For example, the word *dog* would be encoded via

$$\begin{matrix} \text{labels} & \text{cat} & \text{ace} & \cdots & \text{dog} & \cdots & \text{house} \\ \mathbf{v}_{\text{dog}} = [& 0 & 0 & \cdots & 1 & \cdots & 0] \end{matrix}$$
$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{|\mathcal{V}|}$$

The problem with such a representation is that, because the vectors are sparse (specifically one-hot), we cannot use similarity operations like cosine similarity since the dot product between two distinct one-hot-coded vectors is 0.

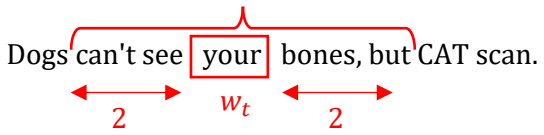
$$\text{sim}\left(\mathbf{v}_{w_i}, \mathbf{v}_{w_j}\right) = \cos^{-1}\left(\frac{\mathbf{v}_{w_i}^T \mathbf{v}_{w_j}}{\left\|\mathbf{v}_{w_i}\right\| \cdot \left\|\mathbf{v}_{w_j}\right\|}\right)$$

Consequently, we would like a vector representation of the vectors that permits such similarity operations.

The word2vec method was proposed to address the above issue.

1.2. Skip-Gram Usual Formulation[2]

The skip-gram method utilizes a concept called a *context*. Given a sequence of words $w_1, \dots, w_T$, the context of the tth word w_t are the words in a symmetric window around w_t. For instance, a context window of size 2 would be as follows:



[2] This section is based on here https://d2l.ai/chapter_natural-language-processing-pretraining/word2vec.html

Two types of words are distinguished: *context words* and *center words*. In this case, "your" is the center word, while "see" is a context word. We will denote center words w_c and context words w_o. The context-center pairs (w_o, w_c) associated with the word "your" above are as follows:

(can't, your), (see, your), (bones, your), (but, your)

Using this definition of context, both context words and center words belong to the same vocabulary, i.e. $w_o, w_c \in \mathcal{V}$.

The skip-gram word2vec method assumes that a center word can be used to generate the context words around it. Intuitively, when I say the word "father" you might think of words like "mother" and "family." We want to capture this intuition that, if we query a word like "father" from our model, we can find these most probable words around it.

Consider the following sequence:

The father loves his child.

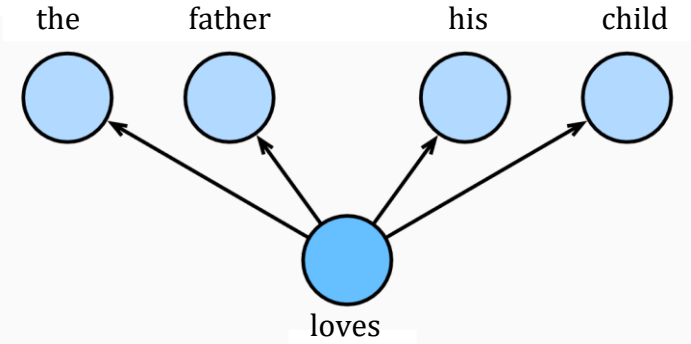
If we choose "loves" as our center word, we might want a distribution that maximizes the probability of the context words so we have a high chance of returning that context. The distribution would be

$$P(\text{“the”}, \text{“father”}, \text{“his”}, \text{“child”}|\text{“loves”})$$

We assume the context words w_o are independently generated from the center word w_c. Consequently, the conditional distribution can now be decomposed as a product of their probabilities:

$$\begin{aligned} &P(\text{“the”}, \text{“father”}, \text{“his”}, \text{“child”}|\text{“loves”}) \\ &= P(\text{“the”}|\text{“loves”}) \cdot P(\text{“father”}, |\text{“loves”}) \cdot P(\text{“his”}|\text{“loves”}) \cdot P(\text{“child”}|\text{“loves”}) \end{aligned}$$

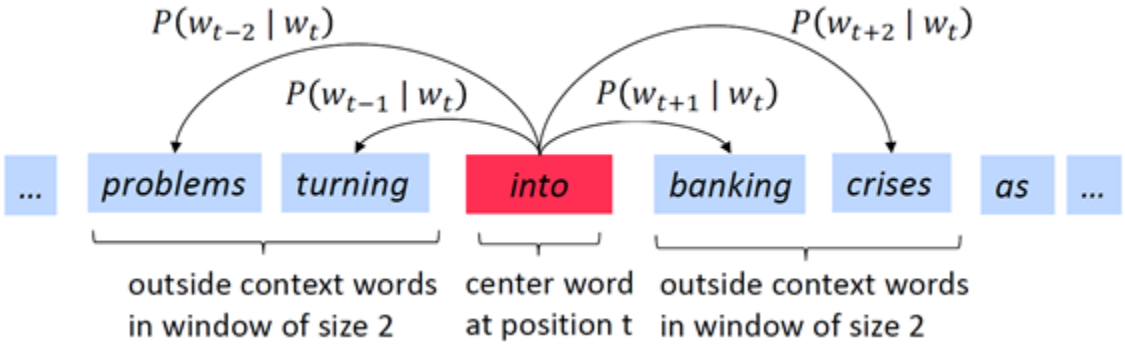
This can be represented with the following diagram



It should be noted that the skip-gram model is a discriminative model because it has the form $P(Y|X)$. More generally, for window size m and center word w_t, we would denote this as follows:

$$P(w_{t-m}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+m}|w_t) = \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j}|w_t)$$

This intuition can be seen in the following diagram:



Source[3]

In the skip-gram model, each word i has two vectors that represent it, $\mathbf{v}_i, \mathbf{u}_i \in \mathbb{R}^d$, where $\mathbf{v}_i$ is its *center word representation* for the ith word and $\mathbf{u}_i$ is its *context word representation* for the ith word. The conditional probability of generating any context word w_o with any center word w_c is given by

$$P(w_o|w_c) = \frac{\exp(\mathbf{u}_o^{\mathrm{T}}\mathbf{v}_c)}{\sum_{i\in\mathcal{V}}\exp(\mathbf{u}_i^{\mathrm{T}}\mathbf{v}_c)}$$

Since we will be trying to optimize for finding the optimal set of context and center vectors, we will have a set of parameters θ, i.e.

$$\theta = \begin{bmatrix} \mathbf{v}_{w_1} \\ \vdots \\ \mathbf{v}_{w|\mathcal{V}|} \\ \mathbf{u}_{w_1} \\ \vdots \\ \mathbf{u}_{w_{|\mathcal{V}|}} \end{bmatrix} \in \mathbb{R}^{2|\mathcal{V}|\times d}$$

where we have assumed each $\mathbf{v}_i, \mathbf{u}_i \in \mathbb{R}^d, \forall i$.

Then, putting all these concepts together, we can now write the original skip-gram probability as

$$P(w_{t-m}, w_{t-m+1}, \dots, w_{t-1}, w_{t+1}, \dots, \dots w_{t+m-1}, w_{t+m}|w_t; \theta)$$

$$= \prod_{\substack{-m\le j\le m \\ j\neq 0}} P(w_{t+j}|w_t;\theta)$$

$$= \prod_{\substack{-m\le j\le m \\ j\neq 0}} \frac{\exp(\mathbf{u}_{t+j}^{\mathrm{T}}\mathbf{v}_t)}{\sum_{i\in\mathcal{V}}\exp(\mathbf{u}_i^{\mathrm{T}}\mathbf{v}_t)}$$

[3] https://medium.com/analytics-vidhya/maths-behind-word2vec-explained-38d74f32726b

Hence, we can see θ represented in this product by virtue of assuming the existence of the $\mathbf{v}_i, \mathbf{u}_i$. Then, the likelihood function would be represented by

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P\left(w_{t+j} | w_t; \theta\right)$$

1.3. Skip-Gram Training

The skip-gram model parameters θ are the center and context vectors. We will naturally use the log-loss instead, thus

$$\ell(\theta) = \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log\left(P\left(w_{t+j} | w_t; \theta\right)\right)$$

Typically, instead of maximizing ℓ, we reformulate the problem for a loss minimization instead, thus using the loss

$$\ell(\theta) = -\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log\left(P\left(w_{t+j} | w_t; \theta\right)\right)$$

Then we can see

$$\begin{aligned}
\ell(\theta) &= -\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log\left(P\left(w_{t+j} | w_t; \theta\right)\right) \\
&= -\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log\left(\frac{\exp\left(\mathbf{u}_{t+j}^{\mathrm{T}} \mathbf{v}_t\right)}{\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\mathrm{T}} \mathbf{v}_t\right)}\right) \\
&= -\left[\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \mathbf{u}_{t+j}^{\mathrm{T}} \mathbf{v}_t - \log\left(\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\mathrm{T}} \mathbf{v}_t\right)\right)\right]
\end{aligned}$$

We can use SGD on this loss to find the optimal θ. To do so, we compute

$$\begin{aligned}
\frac{\partial \ell}{\partial \mathbf{v}_t} &= -\left[\frac{\partial}{\partial \mathbf{v}_t} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \mathbf{u}_{t+j}^{\mathrm{T}} \mathbf{v}_t - \log\left(\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\mathrm{T}} \mathbf{v}_t\right)\right)\right] \\
&= -\sum_{\substack{-m \le j \le m \\ j \neq 0}} \left[\mathbf{u}_{t+j}^{\mathrm{T}} - \frac{\sum_{k \in \mathcal{V}} \exp\left(\mathbf{u}_k^{\mathrm{T}} \mathbf{v}_t\right) \mathbf{u}_k}{\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\mathrm{T}} \mathbf{v}_t\right)}\right]
\end{aligned}$$

$$= -\sum_{\substack{-m \le j \le m \\ j \neq 0}} \left[\mathbf{u}_{t+j}^{\mathrm{T}} - \sum_{k \in \mathcal{V}} \left(\frac{\exp(\mathbf{u}_k^{\mathrm{T}} \mathbf{v}_t)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^{\mathrm{T}} \mathbf{v}_t)} \right) \mathbf{u}_k \right]$$

$$= -\sum_{\substack{-m \le j \le m \\ j \neq 0}} \left[\mathbf{u}_{t+j}^{\mathrm{T}} - \sum_{k \in \mathcal{V}} P(w_k | w_c) \mathbf{u}_k \right]$$

See here for more.[4]

1.4. CBOW Usual Formulation

The continuous bag of words (CBOW) model starts by flipping around the basic idea of the skip-gram model. Let's return to our prior example

The father loves his child.

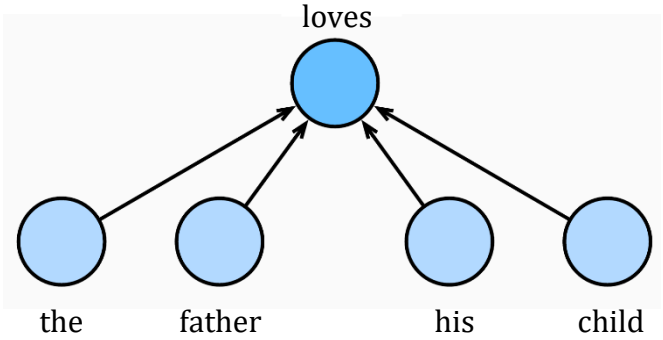
Whereas the skip gram model conditioned on the center word, in the CBOW model we condition on the context words. If we choose "loves" as our center word, we might want a distribution that maximizes the probability of

$$P(\text{“loves”} | \text{“the”}, \text{“father”}, \text{“his”}, \text{“child”})$$

However, unlike the skip-gram model, we cannot just decompose this into a product because the distribution already only captures a single word's probability. Instead, we will compute an "average context" and condition on this average vector as a way to turn the probability again into a sensible sigmoid probability.

Notably, in this model, the meaning of the $\mathbf{u}_i, \mathbf{v}_i$ vectors are flipped. $\mathbf{v}_i$ is the *context word representation* for word i and $\mathbf{u}_i$ is the *center word representation* for word i, again where $\mathbf{v}_i, \mathbf{u}_i \in \mathbb{R}^d$. The conditional probability will be structured so we are generating any center word w_c given its surrounding context words $w_{o_1}, \dots, w_{o_{2m}}$.Note, we have $2m$ words because of both below and above the center word there are m context words.

[4] https://stats.stackexchange.com/questions/253244/gradients-for-skipgram-word2vec



Thus, consider the vector for each of the context words, i.e. $\mathbf{v}_{o_1}, \dots, \mathbf{v}_{o_{2m}}$. Then the average is

$$\bar{\mathbf{v}}_o = \frac{1}{2m}\sum_{i=1}^{2m} \mathbf{v}_{o_i}$$

We will now plug the average vector into the sigmoid function to get the probability. Thus,

$$P\left(w_c | w_{o_1}, \dots, w_{o_{2m}}\right) = \frac{\exp\left(\mathbf{u}_c^{\mathrm{T}} \bar{\mathbf{v}}_o\right)}{\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\mathrm{T}} \bar{\mathbf{v}}_o\right)}$$

Then, the likelihood function would be represented by

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P\left(w_t | w_{t+j}; \theta\right)$$

1.5. Training CBOW

The CBOW model parameters θ are also the center and context vectors. We will naturally use the log-loss instead, thus

$$\ell(\theta) = \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log\left(P\left(w_t | w_{t+j}; \theta\right)\right)$$

Again, instead of maximizing ℓ, we reformulate the problem for a loss minimization, thus

$$\ell(\theta) = -\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log\left(P\left(w_t | w_{t+j}; \theta\right)\right)$$

hen we can see

$$\ell(\theta) = -\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log\left(P\left(w_t | w_{t+j}; \theta\right)\right)$$

$$= -\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log\left(\frac{\exp\left(\mathbf{u}_t^{\mathrm{T}} \bar{\mathbf{v}}_{o_{t+j}}\right)}{\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\mathrm{T}} \bar{\mathbf{v}}_{o_{t+j}}\right)}\right)$$

$$= -\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \left[\mathbf{u}_t^{\mathrm{T}} \bar{\mathbf{v}}_{o_{t+j}} - \log\left(\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\mathrm{T}} \bar{\mathbf{v}}_{o_{t+j}}\right)\right)\right]$$

And so the derivative is

$$\frac{\partial \ell}{\partial \mathbf{v}_{o_i}} = -\frac{\partial}{\partial \mathbf{v}_{o_i}} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \mathbf{u}_t^{\mathrm{T}} \bar{\mathbf{v}}_{o_{t+j}} - \log\left(\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\mathrm{T}} \bar{\mathbf{v}}_{o_{t+j}}\right)\right)$$

$$= \sum_{\substack{-m \le j \le m \\ j \neq 0}} \left[\mathbf{u}_{t+j}^{\mathrm{T}} - \frac{\sum_{k \in \mathcal{V}} \exp\left(\mathbf{u}_k^{\mathrm{T}} \mathbf{v}_t\right) \mathbf{u}_k}{\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\mathrm{T}} \mathbf{v}_t\right)}\right]$$

$$= \sum_{\substack{-m \le j \le m \\ j \neq 0}} \left[\mathbf{u}_{t+j}^{\mathrm{T}} - \sum_{k \in \mathcal{V}} \left(\frac{\exp\left(\mathbf{u}_k^{\mathrm{T}} \mathbf{v}_t\right)}{\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\mathrm{T}} \mathbf{v}_t\right)}\right) \mathbf{u}_k\right]$$

$$= \sum_{\substack{-m \le j \le m \\ j \neq 0}} \left[\mathbf{u}_{t+j}^{\mathrm{T}} - \sum_{k \in \mathcal{V}} P(w_k | w_c) \mathbf{u}_k\right]$$

Part 2: My Project

The purpose of this introduction was to prepare the reader to appreciate why the formulation for this project is valuable. Essentially, rather than having to do SGD, we will make some assumptions about word2vec that allow us to instead use simple linear algebra via SVD to compute the word embeddings.

2.1. Skip-Bow via SVD Matrix Factorization: Introduction

The formulation used in this project will be that discussed in *Neural Word Embeddings as Implicit Matrix Factorization* by Levy and Goldberg. The paper itself has a few shortcomings summarized here.[5]

We are going to have two types of samples, *positive samples* and *negative samples*. Let S_p denote the set of observed context-center pairs in a given corpus, i.e. the positive samples, (hence the p subscript). We can count the frequency of w_o, w_c, and (w_o, w_c).

- $N_p(w_o, w_c)$, the number of times context-center pair (w_o, w_c) occurs in S_p.
- $N_p(w_o) = \sum_{w_c' \in \mathcal{V}} N_p(w_o, w_c')$, the number of times w_o appears as a context word in S_p.
- $N_p(w_c) = \sum_{w_o' \in \mathcal{V}} N_p(w_o', w_c)$, the number of times w_c appears as a center word in S_p.
- $N_p(w_o|w_c) = \frac{N_p(w_o, w_c)}{N_p(w_o)}$, the fraction of times that w_o is a context word for center word w_c in S_p.

Consider a context-center pair (w_o, w_c) and suppose we have two vectors $\mathbf{v}_{w_o}$ and $\mathbf{v}_{w_c}$ denoting words w_o and w_c respectively. Let O denote a binary random variable such that

$$1 = \text{the pair } (w_o, w_c) \text{ was observed}$$
$$0 = \text{the pair } (w_o, w_c) \text{ was not observed}$$

We assume the following distribution for O

$$\mathbb{P}(O = 1|w_o, w_c) = \sigma\left(\mathbf{v}_{w_o}^T \mathbf{v}_{w_c}\right) = \frac{1}{1 + e^{-\mathbf{v}_{w_o}^T \mathbf{v}_{w_c}}}$$

where σ denotes the logistic function, $\mathbf{v}_{w_o}, \mathbf{v}_{w_c} \in \mathbb{R}^d$ are d-dimensional word embeddings for the context word w_o and the center word w_c, respectively.

We also will generate a separate set of negative examples. For each $w \in \mathcal{V}$, we will sample

$$w_c \sim \text{Multinomial}(k, \mathbf{q})$$

[5] http://building-babylon.net/2016/05/12/skipgram-isnt-matrix-factorisation/

where k is the number of samples, $\dim(\mathbf{q}) = |\mathcal{V}|$, and $q_{w_c} = \frac{N_p(w_c)}{|S_p|}$. This generates a new data set S_n and we have the same corresponding quantities, $N_n(w_o, w_c), N_n(w_o), N_n(w_c), N_n(w_o|w_c)$ with the same meanings except on negative samples.

Our goal will be to find the optimal embedding θ^*. The skip-gram method seeks to learn embeddings for all center words and context words such that $\mathbb{P}(O = 1|w_o, w_c)$ for pairs in the corpus while also maximizing $\mathbb{P}(O = 0|w_o, w_c)$ for randomly sampled negative examples. Let $S_p^{(w_c)}$ and $S_n^{(w_c)}$ be the set of context-center pairs that contain center word w_c for positive and negative samples respectively.

Recall the loss function

$$\ell(\theta) = \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log\left(\sigma\left(\mathbf{u}_{t+j}^{\mathrm{T}} \mathbf{v}_t\right)\right) - \sum_{k=1}^{K} \log\left(\sigma\left(-\mathbf{u}_{t+j}^{\mathrm{T}} \mathbf{v}_t\right)\right)$$

This is equivalent to the Levy and Goldberg formulation

$$\ell(\theta) = \sum_{w_c \in \mathcal{V}} \sum_{w_o \in \mathcal{V}} \left[N_p(w_o, w_c)\ \log\left(\log\left(\sigma\left(\mathbf{u}_{w_o}^{\mathrm{T}} \mathbf{v}_{w_c}\right)\right)\right) + N_p(w_c) \frac{N_p(w_o)}{|S_p|} \log\left(\sigma\left(-\mathbf{u}_{w_o}^{\mathrm{T}} \mathbf{v}_{w_c}\right)\right) \right]$$

The loss function can alternatively be written as

$$\ell(\theta) = \sum_{w_c \in \mathcal{V}} N_p(w_c) \left[\sum_{w_o \in S_p^{(w_c)}} N_p(w_o|w_c) \log\left(\sigma\left(\mathbf{v}_{w_o}^T \mathbf{v}_{w_c}\right)\right) + \sum_{w_o \in S_n^{(w_c)}} N_n(w_o|w_c) \log\left(\sigma\left(-\mathbf{v}_{w_o}^T \mathbf{v}_{w_c}\right)\right) \right]$$

2.2. Deriving Optimum

Let $x \equiv \mathbf{v}_{w_o}^T \mathbf{v}_{w_c}$. Then

$$\ell(\theta) = \sum_{w_c \in \mathcal{V}} \sum_{w_o \in \mathcal{V}} \left[N_p(w_o, w_c)\ \log\big(\sigma(x)\big) + N_p(w_c) \frac{N_p(w_o)}{|S_p|} \log\big(\sigma(-x)\big) \right]$$

We can consider the derivative of ℓ with respect to x.

$$\begin{aligned} f(u) &= \log(u) \\ u &= \sigma(k) \\ k &= -x \end{aligned}$$

$$\begin{aligned} \frac{\partial f}{\partial u}\frac{\partial u}{\partial k}\frac{\partial k}{\partial x} &= \frac{1}{u}\sigma'(k) \cdot (-1) \\ &= -\frac{1}{\sigma(k)}\Big(\sigma(k)\big(1 - \sigma(k)\big)\Big) \\ &= \sigma(-x) - 1 \end{aligned}$$

$$\begin{aligned}\frac{\partial f}{\partial u}\frac{\partial u}{\partial k} &= \frac{1}{u}\sigma'(k)\\ &= \frac{1}{\sigma(k)}\Big(\sigma(k)\big(1-\sigma(k)\big)\Big)\\ &= 1-\sigma(k)\\ &= 1-\sigma(x)\end{aligned}$$

So then,

$$\begin{aligned}\frac{\partial \ell}{\partial x} &= N_p(w_o,w_c)\big(1-\sigma(x)\big)+N_p(w_c)\frac{N_p(w_o)}{|S_p|}(\sigma(-x)-1)\equiv 0\\ &= N_p(w_o,w_c)\frac{1}{1+e^x}-N_p(w_c)\frac{N_p(w_o)}{|S_p|}\frac{e^x}{1+e^x}=0\end{aligned}$$

Let $a = N_p(w_o,w_c), b = N_p(w_c)\frac{N_p(w_o)}{|S_p|}$

$$\begin{aligned}a\frac{1}{1+e^x}-b\frac{e^x}{1+e^x}&=0\\ \left(-\frac{(1+e^x)^2}{b}\right)a\frac{1}{1+e^x}-\left(-\frac{(1+e^x)^2}{b}\right)b\frac{e^x}{1+e^x}&=\left(-\frac{(1+e^x)^2}{b}\right)0\\ e^x(1+e^x)-\frac{a}{b}(1+e^x)&=0\\ e^{2x}+e^x-\frac{a}{b}e^x-\frac{a}{b}&=0\\ e^{2x}-\frac{a}{b}e^x+e^x-\frac{a}{b}&=0\\ e^{2x}-\left(\frac{a}{b}-1\right)e^x-\frac{a}{b}&=0\end{aligned}$$

So, plugging a, b back in we have

$$e^{2x}-\left(\frac{N_p(w_o,w_c)}{N_p(w_c)\frac{N_p(w_o)}{|S_p|}}-1\right)e^x-\frac{N_p(w_o,w_c)}{N_p(w_c)\frac{N_p(w_o)}{|S_p|}}=0$$

Part C

$$\begin{aligned}y &\equiv e^x\\ k &= \frac{N_p(w_o,w_c)}{N_p(w_c)\frac{N_p(w_o)}{|S_p|}}\end{aligned}$$

So then substituting in,

$$y^2-(k-1)e^x-k=0$$

Let $A=1, B=(1-k), C=-k$. Substituting in, we have

$$Ay^2+By+C=0$$

Then by the quadratic formula,

$$\begin{aligned} y &= \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \\ &= \frac{k - 1 \pm \sqrt{(1-k)^2 - 4(-k)}}{2} \\ &= \frac{k - 1 \pm \sqrt{k^2 - 2k + 1 + 4k}}{2} \\ &= \frac{k - 1 \pm \sqrt{(k+1)^2}}{2} \\ &= \frac{k - 1 \pm k + 1}{2} \end{aligned}$$

SOLUTION 1	SOLUTION 2
$\frac{k-1+k+1}{2} = \frac{2k}{2} = k$	$\frac{k-1-(k+1)}{2} = -\frac{2}{2} = -1$ This isn't the right one.

So then,

$$e^x = y = k = \frac{N_p(w_o, w_c)}{N_p(w_c)\frac{N_p(w_o)}{|S_p|}}$$

So,

$$e^x = \frac{N_p(w_o, w_c)}{N_p(w_c)\frac{N_p(w_o)}{|S_p|}}$$

$$x = \log\left(\frac{N_p(w_o, w_c)}{N_p(w_c)\frac{N_p(w_o)}{|S_p|}}\right)$$

And plugging back in for x,

$$\mathbf{v}_{w_o}^T \mathbf{v}_{w_c} = \log\left(\frac{N_p(w_o, w_c)}{N_p(w_c)\frac{N_p(w_o)}{|S_p|}}\right) = \log\left(\frac{N_p(w_o, w_c)|S_p|}{N_p(w_o)N_p(w_c)}\right)$$

Thus, the optimum is given by

$$\mathbf{v}_{w_o}^T \mathbf{v}_{w_c} = \log\left(\frac{N_p(w_o, w_c)|S_p|}{N_p(w_o)N_p(w_c)}\right)$$

2.3. Using SVD to find word embeddings

We have now found the optimal dot product between the original embedding vectors we started with. Thus, suppose we create a matrix of these entries. Let $M \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ be a matrix such that

$$M_{ij} = \mathbf{v}_{w_o}^T \mathbf{v}_{w_c} = \log\left(\frac{N_p(w_i, w_j) \cdot N(S_p)}{N_p(w_i) \cdot N_p(w_j)}\right)$$

M_{ij} is known as the pair-wise mutual information (PMI) of (w_i, w_j).

If we factorize M using SVD as follows, we can obtain the word embeddings. Thus,

$$M = U\Sigma V^T = \underbrace{U\Sigma^{\alpha}}_{W_o}\underbrace{\Sigma^{1-\alpha}V^T}_{W_c}$$

where $U \in \mathbb{R}^{|\mathcal{V}|\times k}, \Sigma \in \mathbb{R}^{k\times k}, V^T \in \mathbb{R}^{k\times|\mathcal{V}|}$ and k is the number of

$$W_o = U\Sigma^{\alpha}$$
$$W_c = \Sigma^{1-\alpha}V^T$$

W_o is a our matrix of the center words and W_c is the corresponding context words. Levy and Goldberg state α is a tunable parameter and different values may work better on different tasks. We follow their lead and use $\alpha = \frac{1}{2}$.

2.4. Results

I used the Wikipedia data set from Hugging Face. Due to space constraints and time constraints, I didn't try it on the larger data set since only my local machine was available. See here for more details.[6]

I tested two separate outputs. Firstly, I wanted to be able to query for words that were close to a given target word.

Closest results

INPUT	RESULT
physics	['mechanics', 'quantum', 'thermodynamics', 'dynamics', 'sciences'],
republican	['democrat', 'election', 'democrats', 'presidential', 'whig']
einstein	['lorentz', 'spiral', 'planck', 'physicists', 'mechanics']
fish	['eggs', 'chr', 'salmon', 'gear', 'wild']
algebra	['enough', 'algebras', 'sounding', 'press', 'male'],

I then also wanted to be able to enter in "addition" and "subtraction" operations.

[6] https://huggingface.co/datasets/wikipedia

Algebra results

INPUT	RESULT
professor + employee − student	['ceo', 'employee', 'mayer', 'businessman', 'entrepreneur']
tokyo + china − japan	['beijing', 'shanghai', 'hong', 'nanjing', 'taipei']

Overall, the results were as expected, although the larger data set was required. The code is set to use the smaller data set as the default.