Word Embeddings via PMI-matrix Factorization

95865 recitation by Emaad Ahmed Manzoor, last updated 2018.01.26.

1 Vector representations

Machine learning algorithms take as input *vectors* corresponding to each sample in the data. There are various ways of transforming raw input into vector representations. Some examples:

1. *Images:* Given a grayscale $N \times N$ image containing pixel values $\in [0, 1]$, "unroll" the image into a long vector of size N^2.

2. *Documents:* Given a vocabulary V and a document, compute the frequency of each word in the document. Represent the document by a size-$|V|$ vector of its word frequencies.

2 Vector distance/similarity

Given two vectors $\boldsymbol{u} \in \mathbb{R}^N$ and $\boldsymbol{v} \in \mathbb{R}^N$, there are various ways to compute the distance and similarity between them. The most common way is to compute the Euclidean distance:

$$d_{\text{euclidean}} = \sqrt{\sum_{i=1}^{N} (\boldsymbol{u}[i] - \boldsymbol{v}[i])^2} \tag{1}$$

However, in the case of document vectors, the Euclidean distance is a poor measure of how different two documents are. Consider two documents as follows:

$$\begin{aligned} d_u &= \text{"the cat the dog"} \\ d_v &= \text{"the the the cat cat cat dog dog dog"} \end{aligned}$$

Here, the vocabulary is $V = \{\text{"the"}, \text{"cat"}, \text{"dog"}\}$ and $|V| = 3$. Let each word be associated with an index: "the" with 1, "cat" with 2, "dog" with 3; that specifies its position in the document vector. Hence, the document vectors lying in $\mathbb{R}^3$ are:

$$\boldsymbol{u} = [2, 1, 1] \tag{2}$$

$$\boldsymbol{v} = [3, 3, 3] \tag{3}$$

The Euclidean distance between $\boldsymbol{u}$ and $\boldsymbol{v}$ is 3. Observe that simply increasing the length of either document makes it more different than the other!

To address this issue, document distances are usually measured using their *cosine distance*:

$$d_{\text{cosine}} = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|} \tag{4}$$

This is nothing but the *angle* between the two document vectors; it is independent of how long (the magnitude) each vector is. Hence, you will commonly see cosine distances being used in the natural language processing domain.

3 Word embeddings

Word embeddings are vector representations of words. We would like word embeddings to capture linguistic regularities, such as the following:

1. Semantically similar words have similar embeddings, and dissimilar words have dissimilar embeddings. For example, we would like the embedding of `google` to be similar to the embedding of `apple` and `facebook` because all three of them tend to appear in similar *contexts*.

2. Composing word embeddings is semantically meaningful. For example, we would like the vector composed by `king - man + woman` to be very similar to the embedding of `queen`.

(Mikolov et al. 2013b) and (Mikolov et al. 2013a) introduced the *skip-gram* model to construct such embeddings in an unsupervised manner from any large text corpus using a neural network. In this recitation, we will do something similar, but by factorizing the PMI matrix. Both methods assume the *distributional hypothesis*: that words occurring nearby in text (in the same *context*) are semantically related.

3.1 Mathematical formulation

The input to the skip-gram model is constructed by building a collection of *(pivot word, context word)* pairs from the input document collection. This is done by considering each word as a *pivot*, and then looking at a *window* around that word to obtain its *context* words.

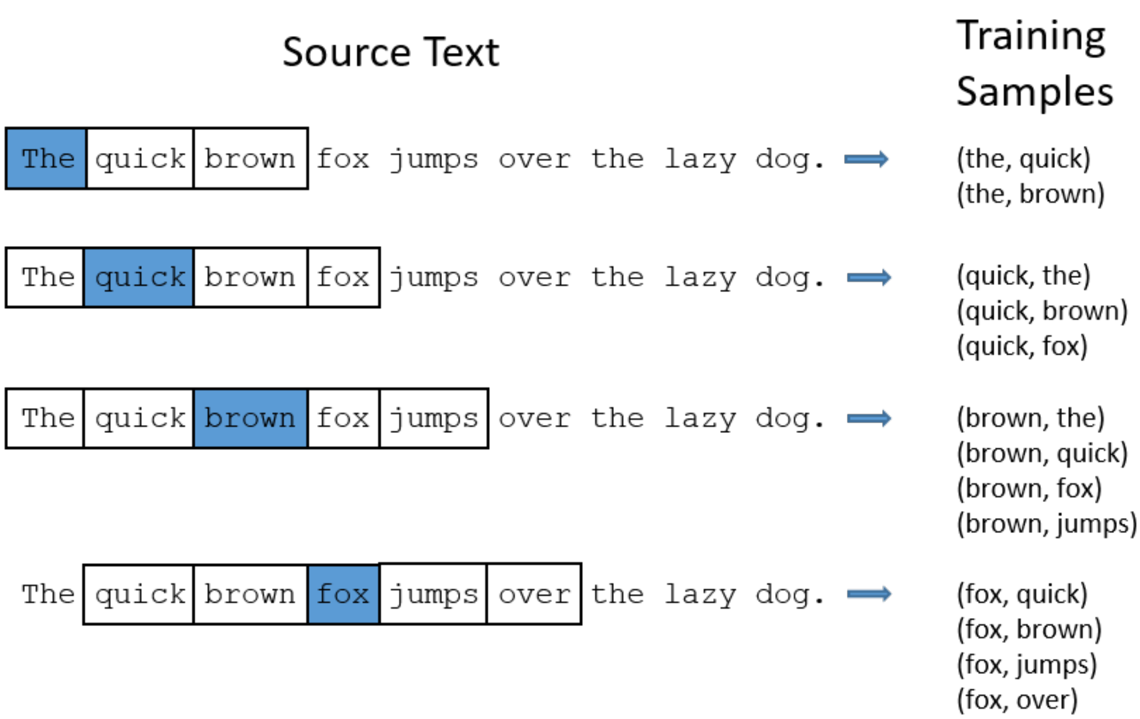

Figure 1: `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/`

The neural network is then trained to learn embeddings for each word such that:

1. $\boldsymbol{p} \cdot \boldsymbol{c}$ is maximized for every pivot word embedding $\boldsymbol{p}$ and the embedding of every word in its context $\boldsymbol{c}$.
2. $\boldsymbol{p} \cdot \boldsymbol{c'}$ is minimized for every pivot word embedding word $\boldsymbol{p}$ and the embedding of every word *not* in its context $\boldsymbol{c'}$.

3.2 PMI matrix factorization

Instead of using a neural network, we could obtain embeddings with the same properties by *factorizing* the *PMI matrix* of the document collection (Levy et al. 2014).

The *PMI matrix* $M \in |V| \times |V|$ is defined for every word pair (u, v) in the vocabulary V as follows (this should be familiar from the lecture):

$$M(u,v) = \log\left(\frac{P(u,v)}{P(u)P(v)}\right) \tag{5}$$

Matrix factorization via the singular-value decomposition (SVD) decomposes a matrix M into matrices U, Σ and V such that $M = U\Sigma V^T$. This decomposition has a number of nice mathematical properties that we will not dwell on. Each row of the U matrix is a word embedding, and the set of embeddings exhibits the desired properties mentioned earlier.

3.3 Implementation

- *Notebook:* `https://gist.github.com/emaadmanzoor/1d06e0751a3f7d39bc6814941b37531d`
- *Dataset:* `https://www.kaggle.com/hacker-news/hacker-news-posts/downloads/HN_posts_year_to_Sep_26_2016.csv`
- *Notes:* `http://www.eyeshalfclosed.com/teaching/95865-recitation-word2vec_as_PMI.pdf`

References

Levy, Omer et al. (2014). "Neural word embedding as implicit matrix factorization". In: *NIPS*, pp. 2177–2185.

Mikolov, Tomas et al. (2013a). "Distributed representations of words and phrases and their compositionality". In: *NIPS*, pp. 3111–3119.

Mikolov, Tomas et al. (2013b). "Efficient estimation of word representations in vector space". In: *ICLR*.