Lecture Note - 08: NLP, Word Representation, Language Model, N-gram, MLE

Dihui Lai

March 29, 2020

Contents

1 Word Semantics and Vector Representations

- Homonymous: a word can have multiple definitions e.g. mouse could mean small rodents or it could mean computer devices.

- Synonyms/antonym (words' relations): couch/sofa, vomit/throw up, filbert/hazelnut; long/short, big/little
- Word sentiments
- Can we represent a word using vectors and quantify those measures?

1.1 Term-term matrix/Word-word matrix

Count the number of times (n) a word occurs in a context window (w) around the target word (t). Let's consider the following setence as an example:

Data scientists are big data wranglers, gathering and analyzing large sets of structured and unstructured data

In the setence, the words 'are', 'and', 'of' are stop words and serve as building blocks to form a setence. While constructing a word representation, let us ignore them for the moment and consider the words in their base format. Thus we end up with a sentence as of the following

data scientist big data wrangler gather analyze large set structure unstructure data

The unique words appeared in the sentence form a dictionary: {*data, scientist big wrangler, gather, analyze, large, set, structure, unstructure*}.

As a first step to construct a term-term matrix, we use the words from the dictionary as columns and the each word in the setence as rows. For simplicity, we consider the terms appear in a context - window of size 2, i.e. $w = \pm 1$. Check the first word in the setence *data*, the words appear within the context-window are *scientist* and *big*. We then fill the corresponding cells $M_{12} = 1$, $M_{13} = 1$ in the term-term matrix. Similarly, the second word *scientist*, has non-zero cell in the matrix $M_{21} = 1$ and $M_{23} = 1$. We repeat this practice and get the term-term matrix below

	data	scientist	big	wrangler	gather	analyze	large	set	structure	unstructure
data	0	1	1	0	0	0	0	0	0	0
scientist	1	0	1	0	0	0	0	0	0	0
big	1	1	0	0	0	0	0	0	0	0
data	0	0	1	1	0	0	0	0	0	0
wrangler	1	0	0	0	1	0	0	0	0	0
gather	0	0	0	1	0	1	0	0	0	0
analyze	0	0	0	0	1	0	1	0	0	0
large	0	0	0	0	0	1	0	1	0	0
set	0	0	0	0	0	0	1	0	1	0
structured	0	0	0	0	0	0	0	1	0	1
unstructured	1	0	0	0	0	0	0	0	1	0
data	0	0	0	0	0	0	0	0	0	1

To construct the term-term matrix, we aggregate the rows of in the matrix above by the row keys (see below). Each row in the term-term matrix is a representaiton of the word appeared in a document.

	data	scientist	big	wrangler	gather	analyze	large	set	structure	unstructure
data	0	1	2	1	0	0	0	0	0	1
scientist	1	0	1	0	0	0	0	0	0	0
big	1	1	0	0	0	0	0	0	0	0
wrangler	1	0	0	0	1	0	0	0	0	0
gather	0	0	0	1	0	1	0	0	0	0
analyze	0	0	0	0	1	0	1	0	0	0
large	0	0	0	0	0	1	0	1	0	0
set	0	0	0	0	0	0	1	0	1	0
structured	0	0	0	0	0	0	0	1	0	1
unstructured	1	0	0	0	0	0	0	0	1	0

For example, *data* and *scientist* have vector represetations $data = [0, 1, 2, 1, 0, 0, 0, 0, 0, 1]$, $scientist = [1, 0, 1, 0, 0, 0, 0, 0, 0, 0]$, respectively.

1.2 Neural Network Based Word Representation

Use neural network to learn word representation is a hot topics in recent years. One simple method is described as below

- The input variable is a one-hot encoding vector. If the vocabulary is of size V, an input vector is has V components $\vec{x} = [0, 0, 0...1, ...0]$
- The hidden layer has n neurons. The input weights matrix W is of size $V \times n$
- The output layer weights W' matrix is of size $n \times V$
- CBOW: take $2m$ words (i.e. w_{c-m}, $...w_{c-1}$, w_{c+1}, w_{c+m}) around the center word w_c as input w_c is the target.
- Skip-gram: take the center word w_c as the input and the $2m$ words (i.e. w_{c-m}, $...w_{c-1}$, w_{c+1}, w_{c+m}) around it as the target.

The word representation/embedding can be calculated as

$$w_i = x_i W$$

x_i is the i^{th} word in the dictionary, w_i is the i^{th} row in the input matrix W

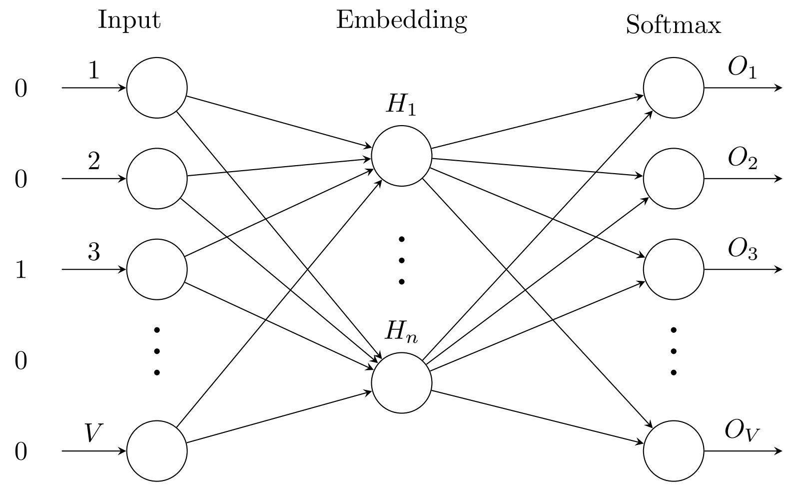

Figure 1: Neural network architecture for learnign COBW and skip-gram embeddings. A vocabulary is fed into the neural network using one-hot encoding methods. For a vocabulary of size V, the input vector is of size 1xV

1.3 Word Representation using Neural Network

2 Cosine Similarity

By looking at the Term-term matrix in Table. 1, we can see that data and scientist seems to have a common context word *big* and could be close in their meanings. How can we quantify this? One possibility is using the dot-product of their vector representation.

$$\vec{v} \cdot \vec{w} = \sum_{i=1}^{N} v_i w_i$$

However, the dot-product favors vectors of higher frequency. Words that appears often are likely to have higher dot-product value than word of low occurence. To normalize the frequency, we can use cosine similarity meature, which is defined as below

$$cosine(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_1^N v_i^2}\sqrt{\sum_1^N w_i^2}}$$

3 Language Model

3.1 N-gram Language Models

- Models that assign probabilities to sequences of words are called language models or LM.

- An n-gram is a sequence of N words e.g. 2-gram (or bigram) "Good Morning", 3-gram "Turn it on"
- N-gram lanuage models estimate the probability of the last word of an n-gram given the previous words

LM: What is the probability of having a sentence that consists a sequence of words: w_1, w_2, w_3 ... w_N, i.e. $P(w_1, w_2, w_3...w_N)$.

Recall the chain rule:

$$P(w_1, w_2, w_3...w_N)$$
$$= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)P(w_4|w_1, w_2, w_3)...P(w_N|w_1, w_2, ...w_{N-1})$$

In the case of bigram, we assume $P(w_N|w_1, ..., w_{N-1}) = P(w_N|w_{N-1})$, since the word is only dependent on the previous word, it is also called Markov assumption. In general case of an n-gram, we assume $P(w_N|w_1, w_2, ...w_{N-1}) = P(w_N|w_{N-1}, w_{N-2}, ...w_{N-n+1})$

3.2 MLE Estimation for bigram

In the case of bigram, the MLE estimation can be formulated as

$$P(w_N|w_{N-1}) = \frac{C(w_{N-1}w_N)}{\sum_w C(w_{N-1}w)} = \frac{C(w_{N-1}w_N)}{C(w_{N-1})}$$

Here, $C(w_{N-1})$ is the count of a word's occurence in a document. $C(w_{N-1}w_N)$ is the number of co-occurence of the word pair w_{N-1} w_N, where w_N appears after w_{N-1} . For example, if we are interested in knowing the probabily that "*house*" occurs after "*white*", $P(house|white)$ we can do the followings: count the total occurence of the word "*white*" in a document and then count the co-occurence of the word pair "*white house*"

3.3 Example: MLE Estimation for bigram

Estimate the bigram for the following corpus, here $\langle s\rangle$ and $\langle /s\rangle$ are introduced as the symbols that represents the begining and end of a setence.

$\langle s\rangle$ *I am Sam* $\langle /s\rangle$
$\langle s\rangle$ *Sam I am* $\langle /s\rangle$
$\langle s\rangle$ *I do not like green eggs and ham* $\langle /s\rangle$

We begin buy counting the words occurence and have $C(I) = 3$, $C(Sam) = 2$, $C(\langle /s\rangle) = 3$, $C(\langle s\rangle) = 3$, $C(\langle s\rangle I) = 2$, $C(\langle s\rangle Sam) = 1$

So we have $P(I|\langle s\rangle) = \frac{2}{3}$, $P(Sam|\langle s\rangle) = \frac{1}{3}$, $P(do|I) = \frac{1}{3}$, $P(am|I) = \frac{2}{3}$, $P(Sam|am) = \frac{1}{2}$, $P(\langle /s\rangle|Sam) = \frac{1}{2}$

The in-sample probability of $P(\langle s\rangle I\ am\ Sam\langle /s\rangle) = P(I|\langle s\rangle)P(am|I)P(Sam|am)P(\langle /s\rangle|Sam) = \frac{2}{3} \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{9}$

3.4 Compare LMs

How do we compare two LM?

- A test data/hold out data set can be used to evaluate a LM. Apply the estiamated conditional probability to the test data set and compare the resulting probability.
- More often than not, perplexity is used as a preferred metric, instead of the raw probability. Perplexity is defined as

$$\begin{aligned} PP(W) &= P(w_1, w_2, ...w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1, w_2, ...w_N)}} \end{aligned}$$

- Maximize probability is equivalent to minimize perplexity