CS300 Assignment 5 - Lecture Scribe

Word Representation
CS224D-Stanford University Lecture-2
Deep Learning for NLP
Rushab Munot, 14405

November 2016

1 Introduction

1.1 What is a meaning?

An idea that is represented by a word/phrase/art/etc - Webster Dictionary.
This is what we, as humans understand. But how to make a computer understand the meaning of a word? A common answer given is to use a taxonomy (like Wordnet) which gives relationships between objects.

1.2 Taxonomy based Representation

For example, a panda is a mammal, a mammal is an animal, an organism, all of them are living beings, which are physical entities which are entities. This basically forms a tree-structure, which must be a DAG (Can't have loops which will go infinitely long which computation)

1.3 Problems with this type of representation:

- Consider the following words - expert, good, adept, practiced. Can they be used interchangeably? Probably no. But they do have similar meanings. This can't be implemented through this simple structure.
- New words coming up need to be added regularly - Example tweets on twitter.
- Requires human work to create the structure and update it.
- Numerical analysis difficult - proficient and expert are more similar than expert and good.
- Doesn't exist for a huge number of languages

2 One Hot Vectors - A simple numerical approach

Words are many a times are regarded as atomic entities. This mathematically is represented as large and sparse vectors.

- In such vectors each word is represented by a vector with size equal to the vocabulary size.
- The words in the vocabulary are arranged in a fixed order (alphabetically).
- All entries in this matrix are zero except one.
- If the i'th entry is one then the concerned word is the corresponding I'th word in the vocabulary.

2.1 Example

Suppose we have a 5 word vocabulary - and, a, the, motel, hotel. hotel will be represented as [0,0,0,0,1], 'the' as [0,0,1,0,0]

2.2 Problems:

- Very large vectors. Google has a billion word vocabulary.
- No notion of similarity. Can't say which words are similar and which are not.

To overcome this problem, a very obvious observation is used. The meaning of a word depends on its neighbors. We look at, say 3 words that come before and 3 words that come after the given word i.e. we look at words in a window around the concerned word. This idea captures both semantic and syntactic information. A window size of 2-5 is sufficient for most tasks.

3 Window-Based Co-Occurrence Matrix

Consider the following corpus:
I like deep learning.
I like NLP.
I enjoy flying.

Counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

The final co-occurrence matrix C is:

$$\begin{matrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{matrix}$$

Note that it a symmetric matrix.

3.1 Problems

- Increase in size with vocabulary
- Very high dimensional - requires lots of storage
- Has Sparsity issues
- Less robust models

How do we overcome these problems?

We transform the matrix to a lesser dimensional matrix with lesser amount of sparsity. Usually 25-1000 dimensions per word.

3.2 Reducing dimensionality - Singular Value Decomposition

Idea - The matrix C is written as C = U*S*V' (' denotes transpose), where S is a diagonal matrix of singular values (in descending order). U is the matrix of left singular values, V is matrix of right singular values.

(Singular values are the eigen values of X'*X. Refer to the wikipedia pages of Singular Values and SVD for more details.

3.3 Problems?

- Very frequent words have a lot of impact. For example the words 'the','he','a' are very very frequent and have very little impact. Some fixes are to fix an upper limit to the entries in the matrix.
- Another thing that can be done is to ignore all of them.
- Use weights in windows. give closer words more importance
- Pearson Correlations

3.4 An insight into the word vectors learnt

After learning the word vectors after applying svd on the co-occurence matrix, very interesting semantic patterns can be seen:
For example, when these words are plotted, tiger, lions, cats will be closer to each other. All Countries will be close to each other, words like 'show', 'shown', 'showed', 'showing' will be very close to each other so will words like 'long', 'longer', 'longest'.

3.5 More Problems with SVD

A major problem is computational overhead (of the order of n^3 for an $n \times n$ square matrix). We have millions of words to train on.
Another problem is that it is very hard to incorporate new words.

4 Overcoming these problems - Getting into Deep-Learning

A way out of this is to learn these vectors directly from the corpus. We look at Word2Vec a recent, simple and fast model (Mikolov et al. 2013)

4.1 Word2Vec - Mikolov et al. 2013

- The idea is to predict words in a window (of length say m) of a given word
- So given a word, we try to guess what words can occur around that word.
- For training these word vectors, we use the following neural network:

$$\text{Input Vector } (V \times 1) \rightarrow \text{HiddenLayer } (H \times 1) \rightarrow \text{Output Vector } (2mV \times 1)$$

- The input vector is a one-hot representation of the input word.
- The hidden layer size H is equal to the word vector size.
- The output vector is 2m one-hot vectors arranged together in order. These are the 2m words in the window that we have predicted.
- The neural network is trained over a large training set.
- If W is the weight matrix for input-¿hidden layer, x is the input vector, then h=W'x
- For the hidden$\rightarrow$output matrix, apart from linear transformation (Weight matrix W') we further add 2m softmax layers (one for each output word), as the target output is a sequence of 2m one hot vectors.
- In this method we get 2 sets of word vectors - one from W and the other from W'
- The required word vectors are the rows of W (or W')
- Note that W is of size V$\times$h

4.2 Results

- word2vec produces wonderful results - It develops a linear relationship between words.
- Thus, we get relations of the type king is to queen what boy is to girl or man is to woman.
- Another example - ’long’ is to ’longer’ what ’tall’ is to ’taller’ or what ’near’ is to ’nearer’
- Further these are linear relations: ’king’-’queen’ = ’boy’-’woman’
- Along with these similar words remain close to each other if the vectors are plotted.
- Thus the results of the co-occurence matrix methos hold along with these.

4.3 Problems

A single word may have multiple meanings but for all these meanings we have the same word vector.

5 GloVe

- Getting the best of both- Count based as well as Direct approach methods
- Stands for global vectors
- Scalable to huge corpora
- Good performance even on small corpus and small vectors