Placing Words into Vector Spaces – A Summary

Computers interpret language differently as humans: Word vectors were introduced to capture the semantics of the symbols used for human language which is a key challenge of natural language processing (NLP). This summary will touch upon some basic and most commonly known principles and methods of placing words into vector spaces.

In NLP, we ideally want to have a model that can explain multiple aspects of words, some of which may include lexical semantics, words similarities, word senses, etc.[1] Lexical semantics is the linguistic study of word meaning and it covers subtopics like lemmas, senses, synonymy, word similarity, word relatedness and connotations. The standard way to represent word meaning in NLP is vector semantics which assists the modeling of the many aspects of word meaning as previously mentioned.[2] The idea is to use a three-dimensional space where the meaning of a word is identified by its **distribution** in text. In these three-dimensional spaces, words are represented as vectors, which are also called **embeddings**. It is suggested that words with similar distributions also have similar meanings as visualized in figure 1 below[2]. One drawback of embeddings is that words with multiple meanings are reduced to a single vector in the semantic space. To be more specific: homonyms are managed better than polysemies. A practical example of producing word embeddings is the **word2vec model** which creates short (dimension of 50-500) and dense (non-zero values) vectors that work better in NLP tasks than sparse vectors.[3]

Figure 1: A two-dimensional projection of embeddings for some words and phrases.[2]

Vector and distributional models of meaning are commonly based on **co-occurrence matrices** that help analyze text in context. The goal of a co-occurrence matrix is to show the number of times different row entities appear in the same context as the column entities.[2] We will focus on the term-document and the term-term co-occurrence matrices:
In the rows of a **term-document matrix**, words of the vocabulary are listed whereas each column displays a document from a set of documents.
Let's look the mathematics behind vector spaces: A vector is a list of elements, in this context integer numbers, and a vector space is "a collection of vectors, characterized by their dimension".[2] The ordering of the numbers in a vector space indicates various dimensions in which documents vary.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 2: The term-document matrix for four words in 4 Shakespeare plays.[2]

1 https://medium.com/@ram.analytics1/an-introductory-notes-on-vector-semantics-tf-idf-model-and-a-toy-implementation-9046198bf7d

2 https://web.stanford.edu/~jurafsky/slp3/6.pdf

3 https://angelina-yang.medium.com/what-are-the-main-problems-with-word-embeddings-like-glove-or-word2vec-250df55fc171

In a term-document matrix, the vectors representing each document have a dimensionality of |V|, which is the vocabulary size. Figure 2 shows that the column vectors, marked in red boxes, have a column vector length of four, thus the vocabulary size is four. Because multi-dimensional spaces are difficult to visualize, in most cases the dimensions are reduced to only two for which the needed dimensions are chosen.

If we take a look at the fourth dimension, the values for the word *wit* are similar in Julius Caesar and Henry V (the values *2* and *3*). The term-document matrix was initially introduced for document information retrieval which is a task that looks for the document *d* from the collection of documents *D*. The document *d* should best match a query *q* given by the user. "The term-document matrix shows the meaning of a word by the documents it tends to occur in"[2] because similar vectors mean that the words are similar as they more likely occur in similar documents.

The **term-term matrix** (also called word-word matrix or term-context matrix) is the other popular co-occurrence matrix in which the columns are labeled by individual words instead of documents. The dimensionality here is |V| x |V|. Each cell shows the number of times the row (target) word and the column (context) word co-occur.

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Figure 3: Co-occurrence vectors for four words, shows just 6 of the dimensions.[2]

The cells in figure 3 indicate that the words *cherry* and *strawberry* are more similar to each other than the words *digital* and *information*.

When the vectors are finally put in a vector space, the next step is to measure the similarity between two vectors with the most common of similarity metric, **the cosine** – a normalized dot product. Through calculating the cosine with the formula, two vectors are more similar if the cosine is closer to the number 1:

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

Deriving from the coordination system below in figure 4, the angle between *digital* and *information* is smaller than the angle between *cherry* and *digital*; the smaller the angle between the two vectors the more similar they are.

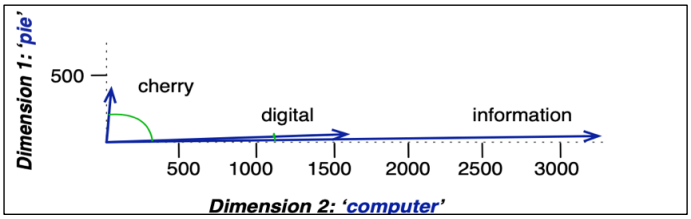

Figure 4: A graphical demonstration of cosine similarity.[2]

Till this point we have discussed the raw frequency of words which is not very discriminative in terms of word meanings. For good discrimination uninformative words like *the* or *it* need to be ignored, even if they occur in high frequencies. PPMI algorithm and tf-idf weighting are the two most common solutions to tackle this problem: **PPMI** stands for pointwise positive mutual information and is popular for word-context matrices whereas **tf-idf weighting** weights each cell by it inverse document frequency and term frequency and is mostly used to determine if two documents are similar.[2] The tf-idf weighting of the value for word *t* in document *d*, $w_{t,d}$ thus combines term frequency with *idf*: $w_{t,d} = tf_{t,d} \; x \; idf_t$