Calculating and Visualizing Word Similarity – A Summary

So far, we have covered how words are placed into vector spaces so that the computer can capture and interpret the semantics of the symbols used for human language, a key challenge of natural language processing (NLP). We have looked at commonly known methods and principles for placing words into different vector spaces. This summary will address the next step, the most popular ways to measure and visualize distances and similarities between word embeddings placed in a vector space.

To set the mathematical foundation for the topics discussed in this summary we will firstly look at the **dot product**, also called the scalar product or the inner product, which is closely related to both the Euclidean distance and as well the cosine similarity. Because the dot product tends to be high when two word-embeddings have larger values in the same dimension, the dot product thus serves as a similarity metric:

$$\text{dot product}(\mathbf{v},\mathbf{w}) = \mathbf{v}\cdot\mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

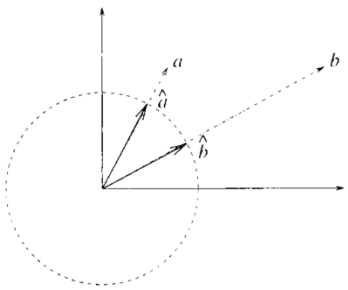
To calculate the similarity between two word vectors we use the most common similarity metric, the cosine of the angle between the vectors. Due to the cosine being based on the dot product there is an issue that needs to be solved; the raw dot product prefers long vectors. Since frequent words tend to co-occur with more words and have larger co-occurrence values with each of them, more frequent words come to have longer vectors. This means that with the raw dot product will be higher for frequent words - the same frequently occurring words end up too similar to most other words. However, this is troublesome because the goal of the similarity metric is to show how similar two words are regardless of their frequency. To counter this problem, we **normalize the dot product**, by dividing the dot product by the lengths of each of the two vectors. The dot product of two **normalized vectors** *v* and *w* corresponds exactly with their cosine similarity:

Vector normalization is a way to disregard unfair advantages and disadvantages. It evens out the similarity scores by removing any extra preference or penalty given to the word vectors of frequently

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

occurring words. One way to think of a normalized or *unit* vector is as a projection of the vector onto a circle of radius 1. Since all unit vectors have the same length, the only distinction these vectors have is their directions. The vector normalization of a vector *v* would be computed and illustrated as follows:

$$\hat{v} = \frac{v}{||v||}$$



The dot product is also related to the Euclidean distance which is defined as the length of a line segment between two points in a space and can be computed as follows:

$$d(a,b) = \sqrt{\sum (b_i - a_i)^2}$$

To make the **distinction between the cosine similarity and the Euclidean distance** clearer, the following can be derived by comparing both metrics:
The values of the Euclidean distance range from 0 to ∞ whereas the values of a cosine similarity stretch between 0 to 1. One drawback of the Euclidean distance is that large vectors are *too far* from smaller vectors whereas a disadvantage of the cosine similarity is that frequently occurring words are given a higher similarity and thus making words *too similar*.

The Euclidean distance lies the foundation to furthermore calculate the Euclidean distance matrix. Let's do a **comparison between the co-occurrence matrix and the distance matrix** to make their differences clearer to understand:
The **co-occurrence matrix** is a table that displays the number of times row entities (e.g. words) appear in the same context as column entities (e.g. documents). This matrix's advantage is that it preserves the semantic relationship between words, e.g. *cat* and *dog* tend to be closer together than *cat* and *book*. On the other side, the co-occurrence matrix reaches its limits when it comes to capturing nuances of language use like irony or sarcasm.
The **distance matrix** is defined as a table that shows the pairwise distances between points where a point's distance with itself is 0. Advantages of calculating a distance matrix are that it can be used to group patterns in larger dimensions and is useful to resolve ambiguities. One of the weakness of the distance matrix is that complex relationships between words may not be identified.

As a result of the principles and methods discussed above, the visualization of embeddings can help with the understanding, improving and applying models of word meaning. One of the many visualization methods is called **clustering** that organizes objects into meaningful groups, called clusters, in the embedding space. Similar objects are grouped into the same cluster and dissimilar objects in other clusters. In our case words are clustered based on the context they occur in. Clustering allows us to get generalizations over the data and is used in multiple NLP tasks, such as semantics, summarizations, text mining or word sense disambiguation.

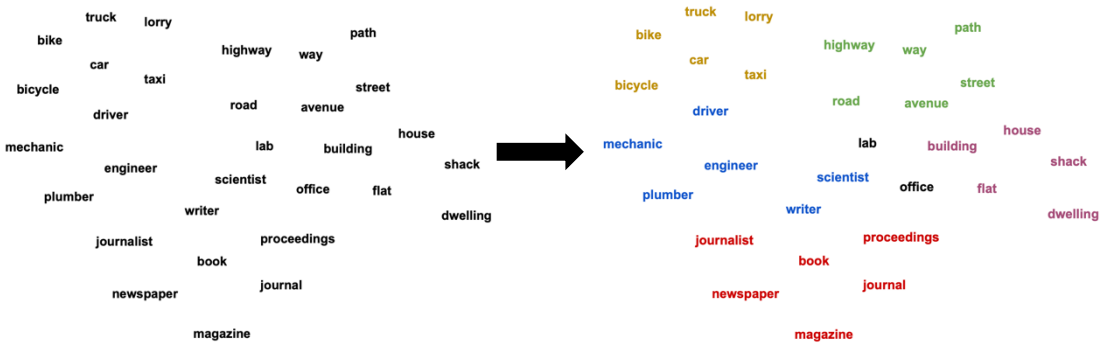


Figure 1: Visualization of how a set of nouns is organized into meaningful clusters