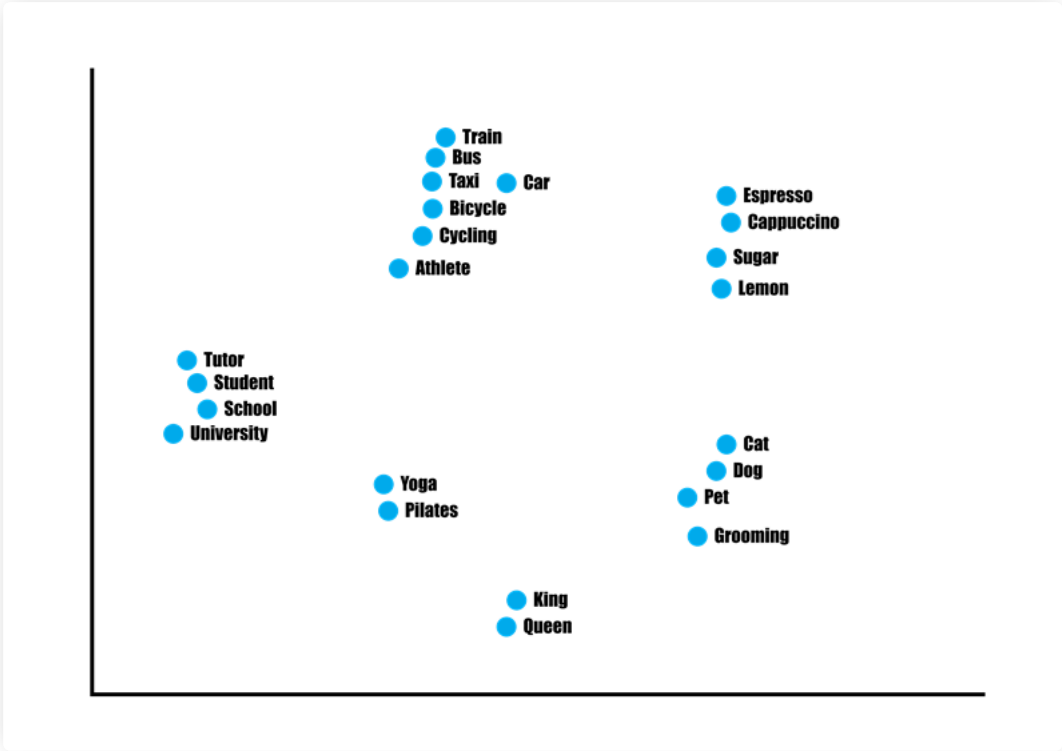
# A High-Level Introduction To Word Embeddings

George Pipis November 29, 2020 8 min read




Tags: word embeddings

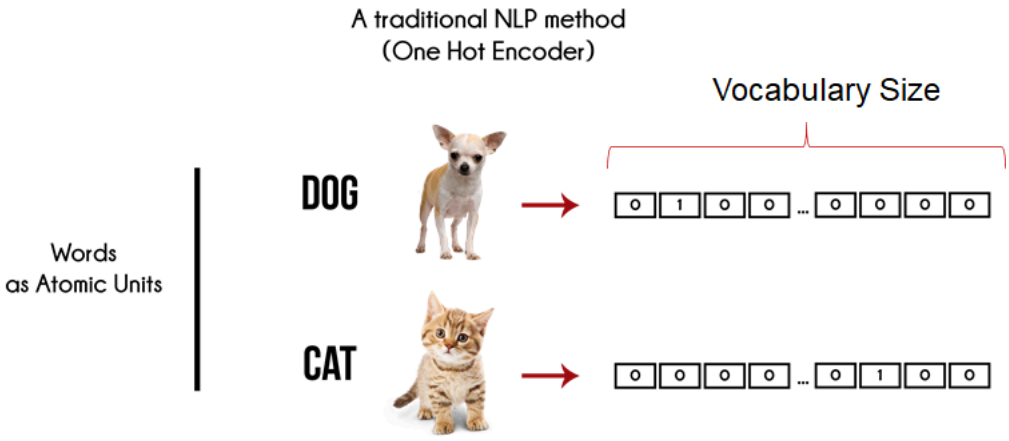
This post is a high-level introduction to Word Embeddings made by the **Predictive Hacks Team** (**Billy** & **George**).

## A common representation of words

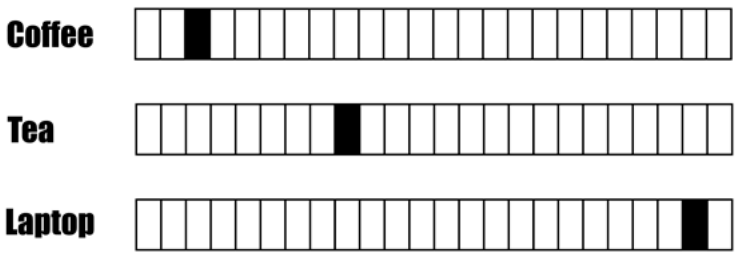
The most common representation of words in NLP tasks is the One Hot Encoding. Although this approach has been proven to be effective in many NLP models, it has some drawbacks:

- The encodings are arbitrary
- This approach leads to data sparsity with many zeros
- It doesn't provide any relation between words.

Below we can see an example of One Hot Encoding for the words "Cat" and "Dog". As we can see these two vectors are independent since their inner product is 0 and their Euclidean distance is $\sqrt{(2)}$. Notice that this applies to every pair in the




Notice that this applies to every pair in the vocabulary, meaning that every pair of words are independent and their distance is $\sqrt{(2)}$. For example, the words below are considered independent and the distance – similarity between any pair of words is the same.

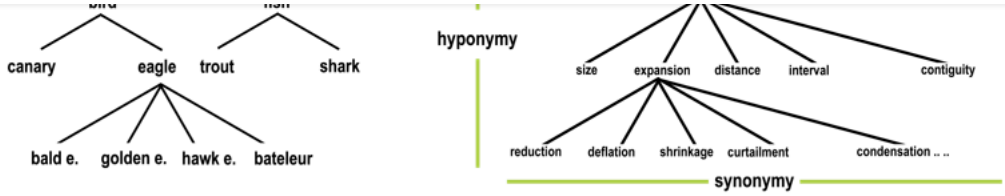



This is an issue for NLP tasks since we want to be able to capture the relation between words. Clearly, the **Coffee** is closer to the **Tea** than to the **Laptop**.

## Attempts to define the distance between words

Linguistics has worked in several word models trying to define word similarities. An example is the **WordNet** where:

- Groups English words into sets of synonyms called synsets.
- Provides short definitions and usage examples.
- Records a number of relations among these synonym sets or their members.
- ❌ Although it can represent a relative distance between words, it cannot represent the words mathematically.
- ❌ There is still a need for linguistics.




## Word Embeddings: Intuition

The idea was to build a model where words which are used in the same context are semantically similar to each other. We could use the phrase that "**A word is characterized by the company it keeps"**

Let's consider the following example of "Tea" and "Coffee" and think

A cup of **tea**
A cup of **coffee**
**Tea** or **coffee**?
**Coffee** and **tea** have caffeine
Let's go for a **coffee**
Let's get a **tea**
**Coffee** vs **Tea**: Which is Best?
I avoid adding sugar to my **tea**
I drink **coffee** with two spoons of sugar

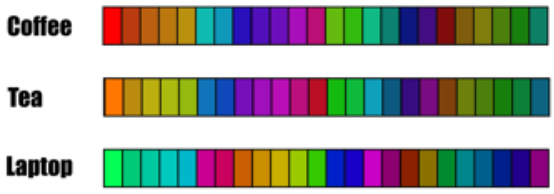

We want a model which will be able to state that **coffee** and **tea** are close and they are also close with words like **cup**, **caffeine, drink, sugar** etc.

## Word Embeddings: The Algorithms

All the available algorithms are based on the following principles:

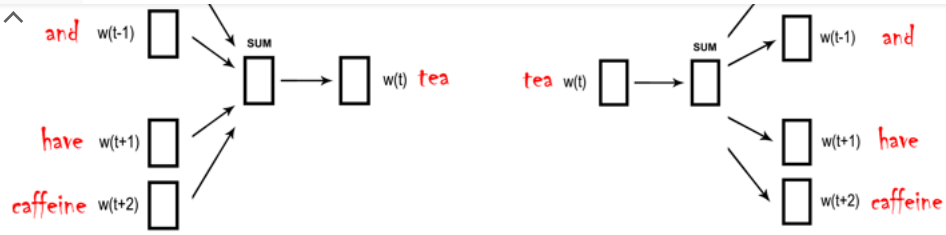
- Semantically similar words are mapped to nearby points.
- The basic idea is the Distributional Hypothesis: words that appear in the same contexts share semantic meaning like **tea** and **coffee**.

The most common algorithms are the **Word2Vec** (Mikolov et al. (2013) at Google) and **GloVe** (2014 Stanford) where they take as input a large corpus of text and produce a vector space typically of 100-300 dimensions. So the corresponding Word Embeddings of the words **coffee, tea** and **laptop** would look like:



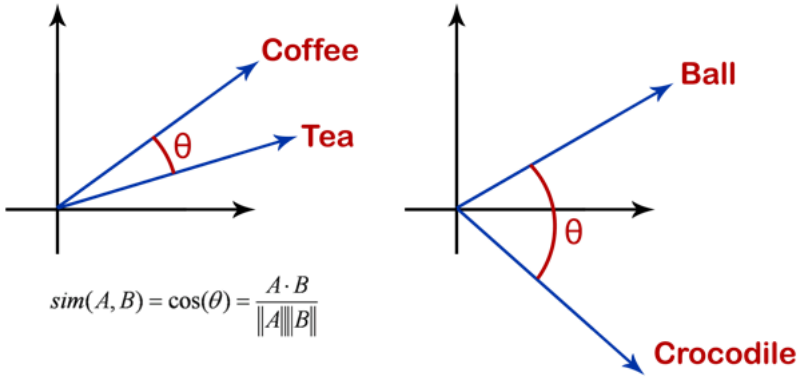

## Word2Vec Algorithm

Word2vec can be built using a continuous bag of words (**CBOW**) or **Skip-Gram.** In essence, it is a neural network with a single hidden layer where its weights are the embeddings. It is like trying to predict a middle word in a window of 3-5 words (CBOW) or the closest 2-4 neighbors of a specific word (Skip-Gram).




## Similarity between words

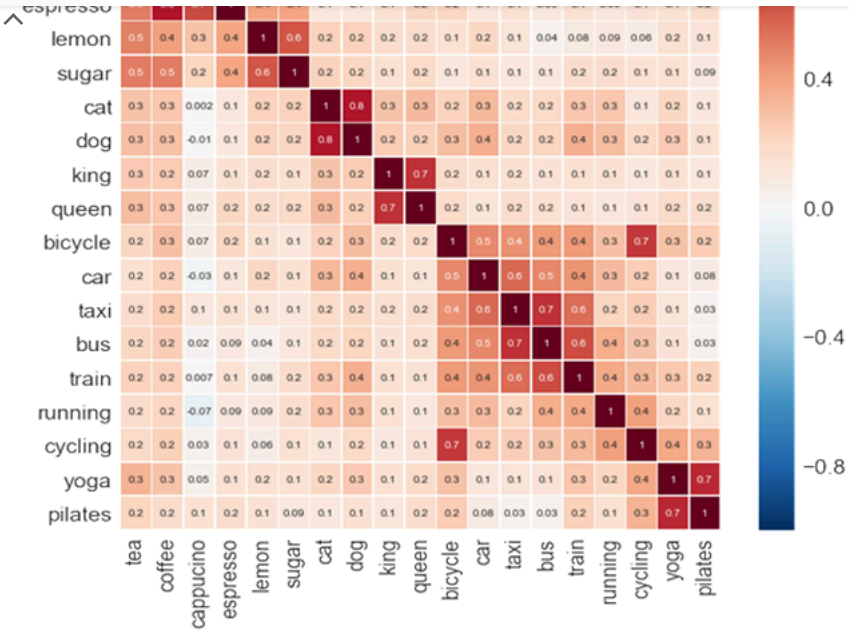
We can calculate the angle ***θ*** between two vectors of two or more dimensions and their similarity can be defined by the cosine of ***θ***. Example in two dimensions:




### Example: Cosine Similarity

**Pre-trained word and phrase vectors.**

We are using a pre-trained model from the Google News dataset (about **100 billion words**). The model contains 300-dimensional vectors for **3 million words and phrases**. In the heatmap below, we represent the Cosine Similarity of every pair of words. As we can see the **tea** is close to the **coffee**, the **yoga** is close to **pilates**, the **king** is close to the **queen**



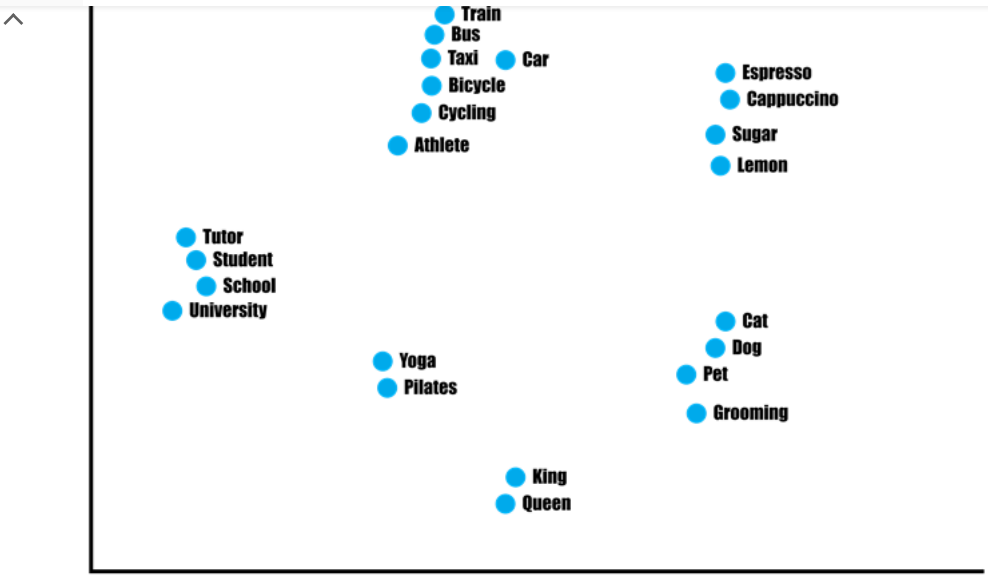

**Example: For a given word return the 5 most similar words**

Let's see which are the most similar words of **France**, **NBA**, **crossfit, piano** and **wine** based on google word2vec which was trained around 2013.

| Similar\|Word | France | NBA | crossfit | piano | wine |
|---|---|---|---|---|---|
| 1 | spain | shaq | boxercise | violin | chardonnay |
| 2 | french | celtics | aerobics_kickboxing | cello | sparkling_wine |
| 3 | germany | cavs | Jumping_rope | clarinet | sauvignon_blanc |
| 4 | europe | cobe | bodyweight_exercises | pianist | rosé |
| 5 | italy | nfl | tae_bo | trombone | merlot |

## Word Representation into 2 Dimensions

We cannot represent features of more than 3 dimensions. In our case we reduce the 300 dimensions into 2 using the T-SNE algorithm. The graph represents the word embeddings in 2D dimensions.

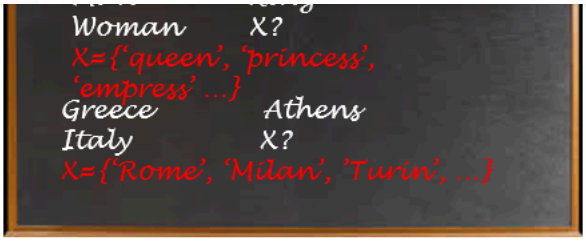


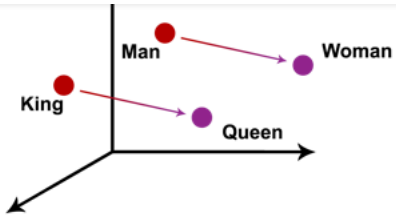

## Context Matters

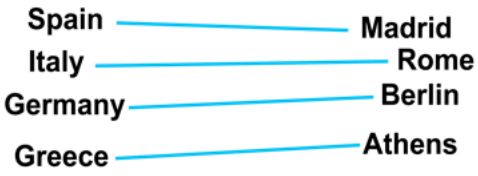
Consider the scenario that you ask a kid from primary school to tell you similar words of the word "**python**" and that you ask a lady to tell you similar words of the word "**ruby**". What you would expect to get as answers? It is most likely the kid to give answers like "snake", e "anaconda", "cobra", "viper", "lizard" etc and it is most likely the lady to give answers like "diamond", "gem", "stone" etc. On the other hand, if we try to find similar words to the words "ruby" and "python" in StackOverflow Community then we would get answers like "java", "R", "c", "programming", "language" etc. Clearly, a significant factor is the corpus that we have used to train the Word Embeddings and depending on the analysis that we want to perform we need to choose the proper one.

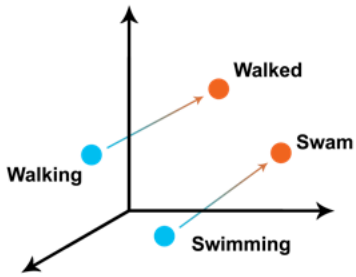
## Word Analogies

With Word Embeddings we can return "Word Analogies" which are very important for NLU tasks



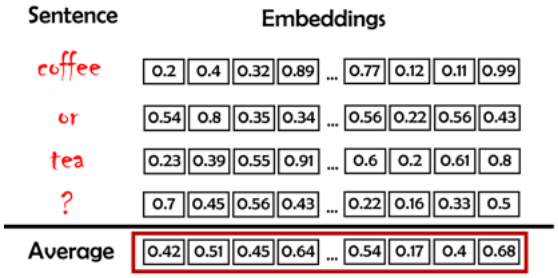



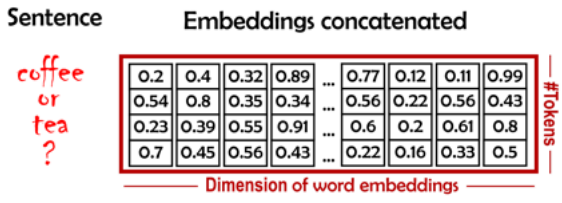

As we can see, in the Analogy "man-king woman?" it returns words such as "queen", "princess" and "empress". For the analogy "greece-athens Italy?" it returns words such as "rome", "milan" and "turin". As we see there is no unique answer and there is not only one right answer. For example for the analogy with the cities, apart from Rome which is the capital of Italy, it returned the Milan which is the technological hub of Italy.

## Document embeddings

Earlier we showed the word embedding. The question is how we can represent a whole document like a tweet or a review as a vector. There are many different approaches for this case and we will mention three of them.

- One approach is to take the average of every single word vector. The document will have the same dimensions as the word embeddings.
- We can concatenate all the words-vectors, so the final dimension will be **(number of words) x (dimensions of word embeddings)**
- We can apply a Doc2Vec machine learning algorithm.




## Applications

The word embeddings are widely used in the following tasks:

- Machine Translation
- Sentiment Analysis
- Document Classification/ Clustering
- Topic Modelling
- Automatic Speech Recognition
- Document Similarities
- Natural Language Generation
- Natural Language Understanding

Tags: word embeddings

## Share This Post

## Leave a reply

Default Comments (0) Facebook Comments

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment

## Subscribe To Our Newsletter

Get Updates And Learn From The Best

PREVIOUS
Simpson's Paradox and Misleading Statistical Inferen...

NEXT
How to Create a Powerful TF-IDF Keyword Research ...

## More To Explore


### Image Captioning With HuggingFace

Image captioning with AI is a fascinating application of artificial intelligence (AI) that involves generating textual descriptions for images automatically.

George Pipis • March 21, 2024

**Intro To Chatbots With HuggingFace**

In this tutorial, we will show you how to use the Transformers library from HuggingFace to build chatbot pipelines. Let's

George Pipis • March 15, 2024

#Tag Cloud

api aws chatgpt consecutive crypto cryptocurrency data science deploy elbow method example flask huggingface interview question k-means kraken langchain linux logistic regression lstm machine learning monte carlo nlg nlp object detection OpenAI opencv pandas pillow probability pytesseract python R recommender systems scraping SQL streak streamlit tensorflow text generation text summarization time series transformers tutorial unix web app