**Project Synopsis**

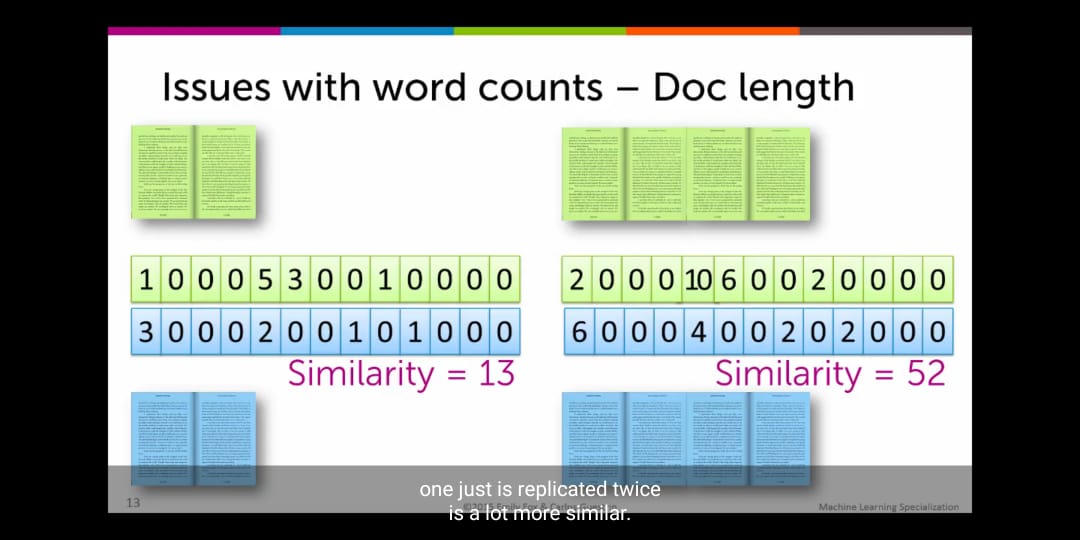
**Text Similarity**

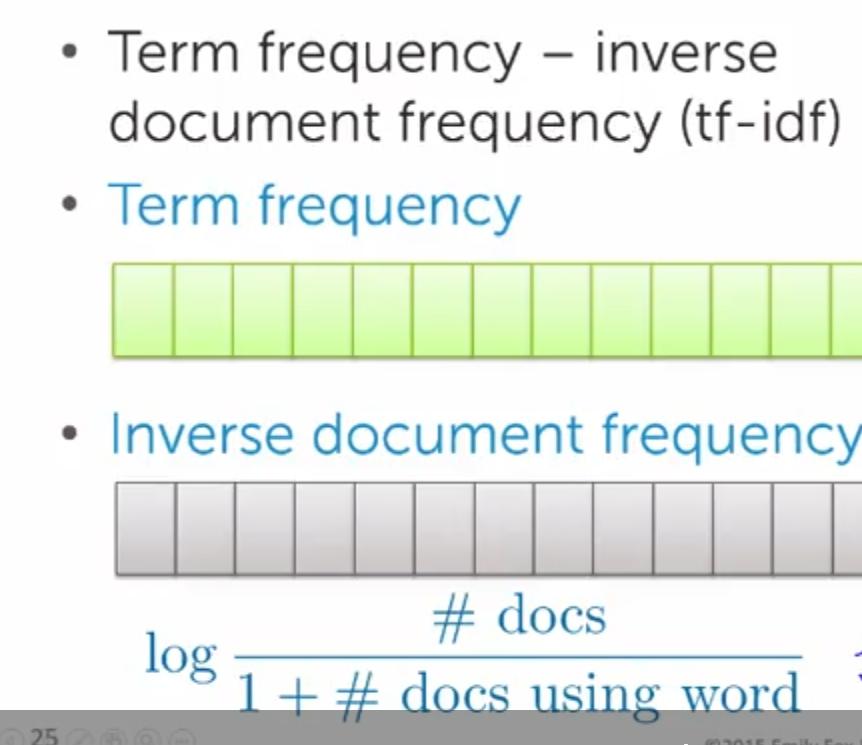
**Text Similarity:** Compare any two given texts and provide a similarity score (Using Cosine Similarity and Word Embeddings).

* **What is text similarity?**

Textual **Similarity** is a process where two **texts** are compared to find the **Similarity** between them. This thesis goes into depth with subjects as **Similarity** and its definition, Language and Words, then algorithms and other tools used to find Textual **Similarity**

* **How do we go about it?**

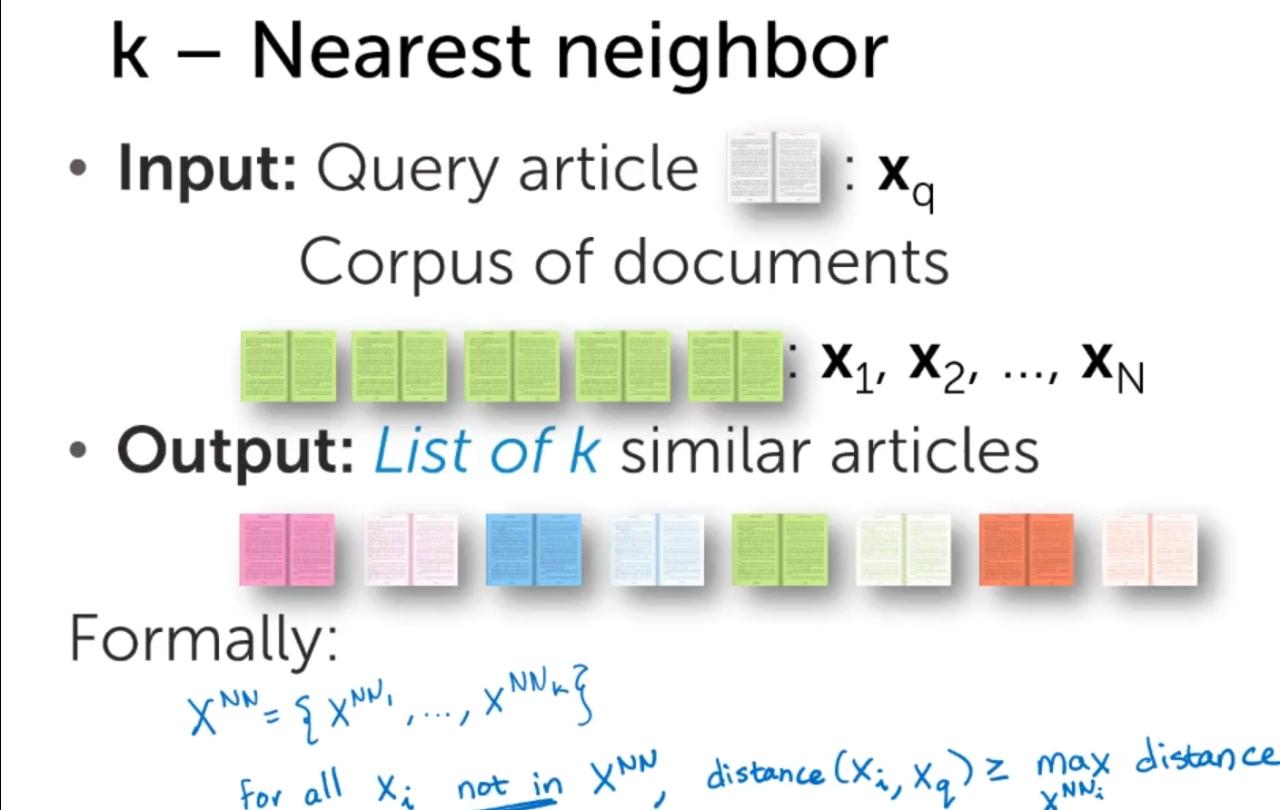
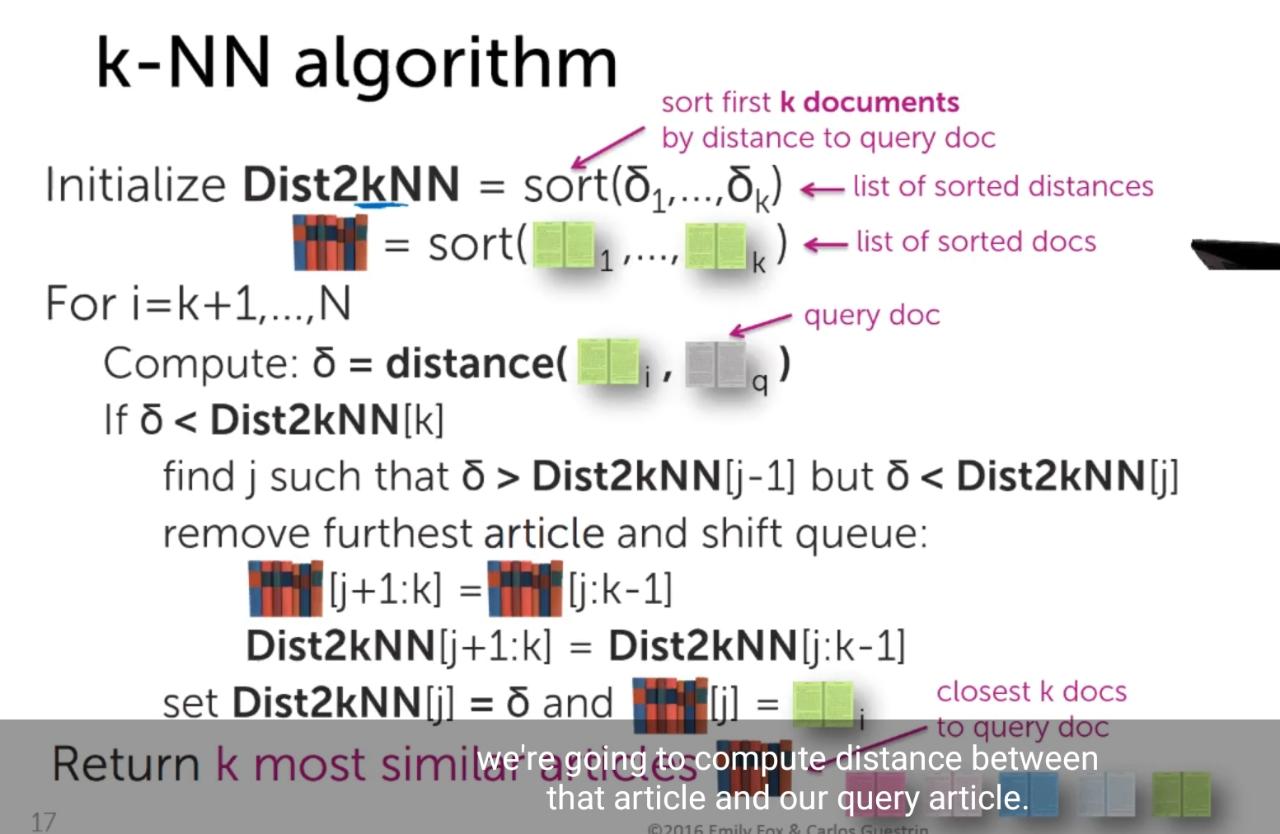
The most basic method can be to keep a count of all the words in a particular Document and then match the frequency of those same words in some other document and calculate similarity as follows:



However the problem with this method is that with the page length, the frequency of the most common words increases and hence the rare words can’t be highlighted or are ignored. To deal with this problem, **tf-idf** approach is used wherein the idf is calculated using the formula given in image 2 which results in highlighting the rare words more accurately!

But if we are asked to find similarity from a large set of data or the **corpus,** then we may have to process each document individually, this can be achieved in the following ways:

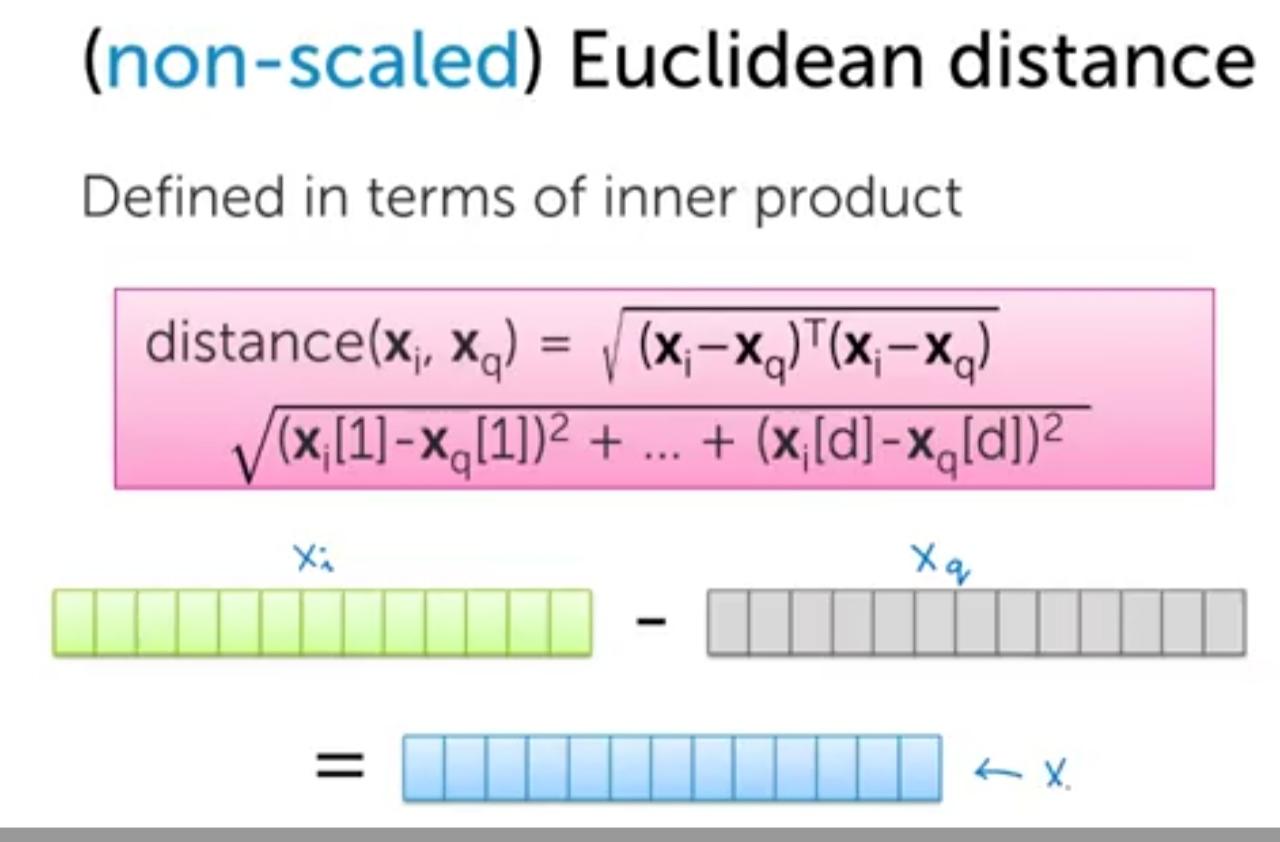
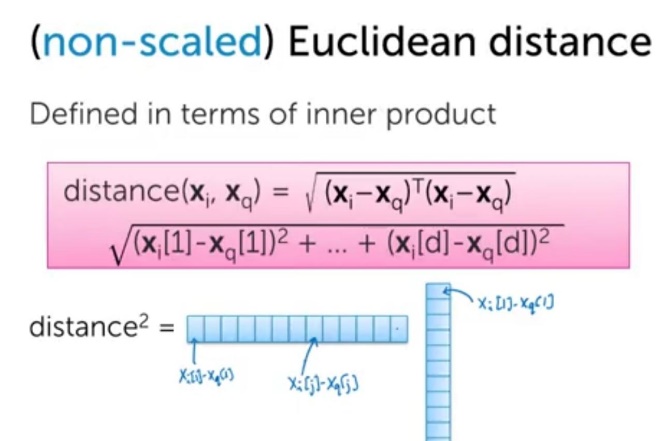
1. We can arrange each document in space based on the similarity and then find the nearest document. This method is called 1-NN (single nearest neighbour)
2. In this method we find the list of ‘K’ Nearest Neighbours by the following algorithm:



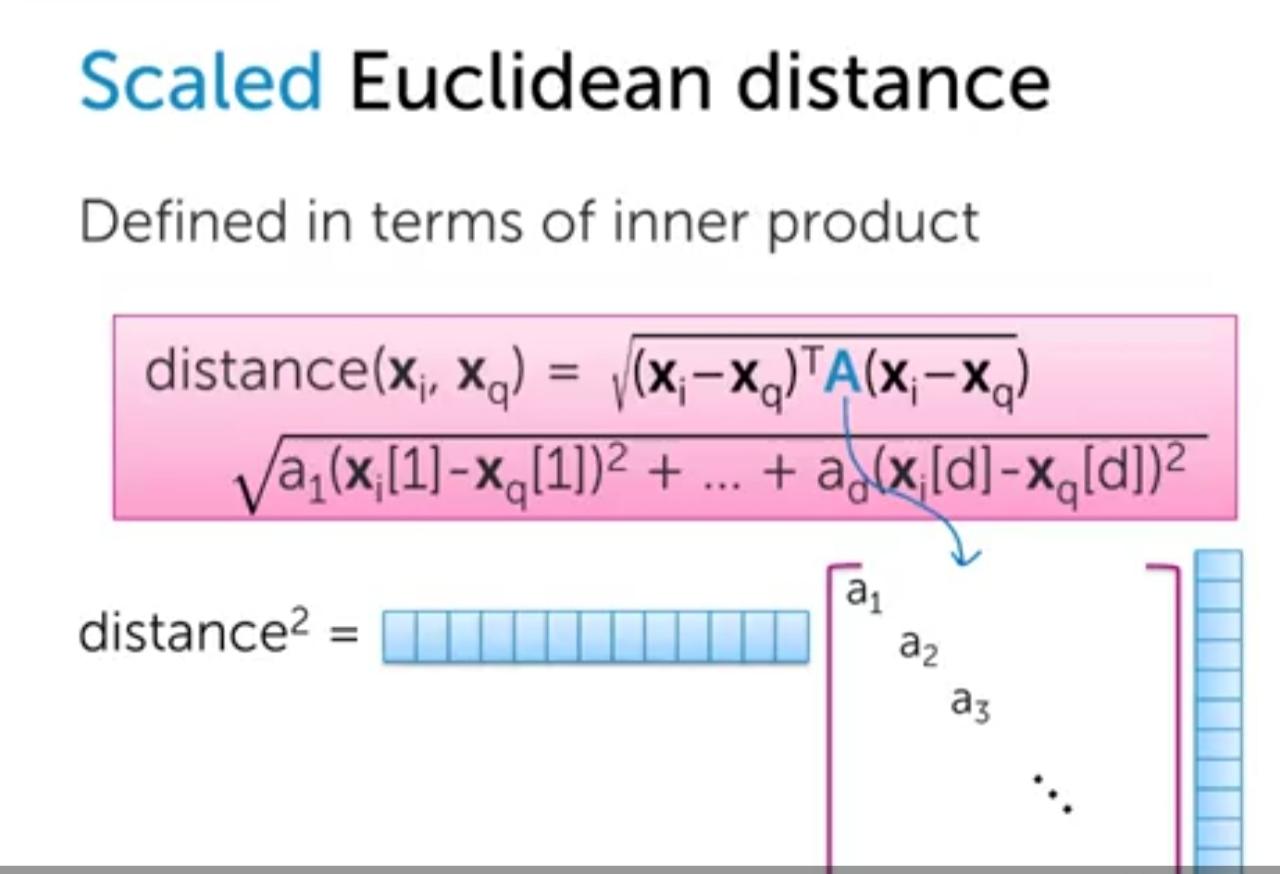
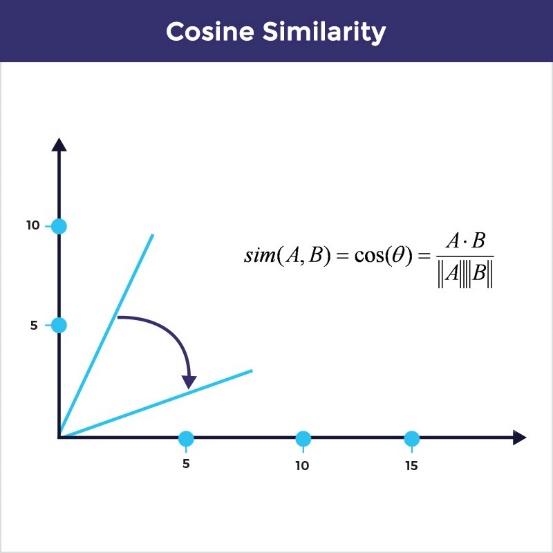
Cosine similarity and Euclidian Distance come into the picture as distance metrics when there are weighted features as:

* Some features are more relevant than others
* Some features vary more than others

To handle this scenario, the Euclidian distance can be calculated as:

The cosine similarity can be calculated as:



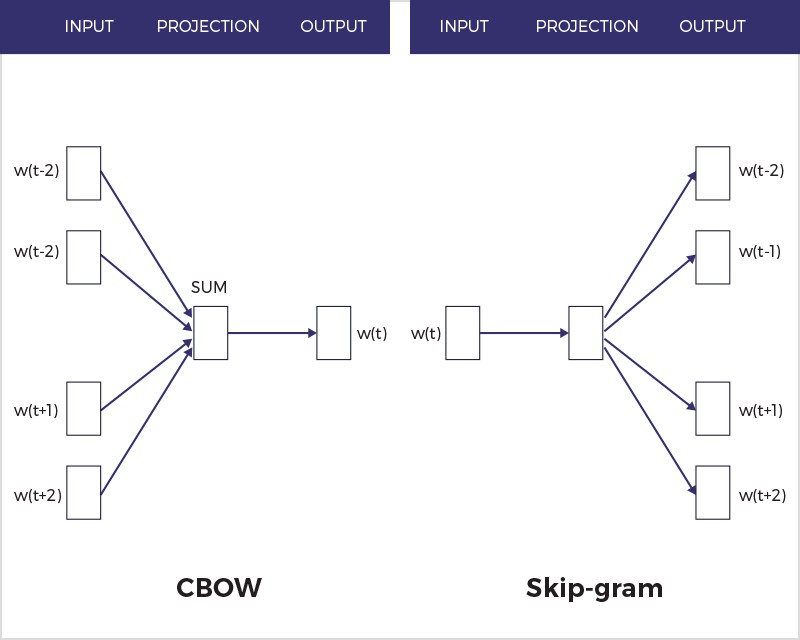
When to use Cosine similarity?

When the length of the document is more, cosine Similarity is preferred as we **NORMALIZE the vector.**

But if we are considering the number of reads of the document then we must go for Euclidean Distance.

*Word2vec*

Using Word2Vec embeddings, word will be represented as a multidimensional array. The two unsupervised algorithms, Skip-gram, and CBoW are used to generate word embeddings.



The size of the embedding can vary based on the target size selected at the time of training. For this case study, we are using a pre-trained model trained on Wikipedia dataset. This model has approximately 6 lakhs words in the vocabulary. Each word is represented with 300 vector length. In the vector space, similar words will be nearer to each other.

Gensim library is one of the popular for word embedding operations. This allows you to load pre-trained model, extract word-vectors, train model from scratch, fine-tune the pre-trained model.

The basic steps for any of these methods are:

1. Text Pre-Processing
2. Feature Extraction
3. Vector Similarity
4. Decision Function
5. Data Preparation
6. Data Pre-processing
7. Uniform case
8. Remove stop words
9. Remove punctuation
10. Remove non-ascii characters
11. Lemmantization