LSA

All languages have their own intricacies and nuances which are quite difficult for a machine to capture (sometimes they’re even misunderstood by us humans!). This can include different words that mean the same thing, and also the words which have the same spelling but different meanings.

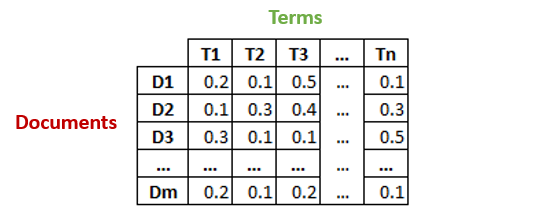
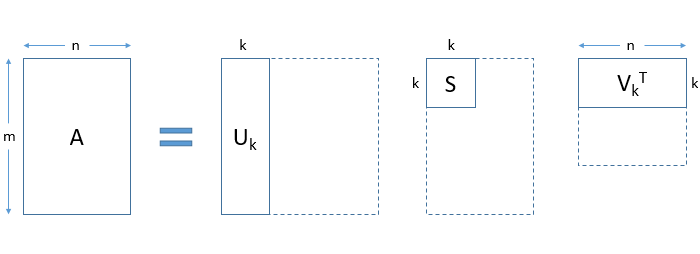
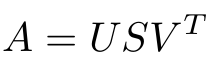
For example, consider the following two sentences:

1. I liked his last **novel** quite a lot.
2. We would like to go for a **novel** marketing campaign.

In the first sentence, the word ‘novel’ refers to a book, and in the second sentence it means new or fresh

### **Steps involved in the implementation of LSA**

Let’s say we have **m**number of text documents with **n** number of total unique terms (words). We wish to extract **k** topics from all the text data in the documents. The number of topics, k, has to be specified by the user.

* Generate a document-term matrix of shape **m x n**having TF-IDF scores**.**
* Then, we will reduce the dimensions of the above matrix to **k**(no. of desired topics) dimensions, using singular-value decomposition (SVD).
* SVD decomposes a matrix into three other matrices. Suppose we want to decompose a matrix A using SVD. It will be decomposed into matrix U, matrix S, and VT (transpose of matrix V).Each row of the matrix ****Uk (document-term matrix)**** is the vector representation of the corresponding document. The length of these vectors is k, which is the number of desired topics. Vector representation for the terms in our data can be found in the matrix ****Vk (term-topic matrix)****.
* So, SVD gives us vectors for every document and term in our data. The length of each vector would be **k**. We can then use these vectors to find similar words and similar documents using the cosine similarity method.

O/p-

### **Topics Visualization**

To find out how distinct our topics are, we should visualize them. Of course, we cannot visualize more than 3 dimensions, but there are techniques like PCA and t-SNE which can help us visualize high dimensional data into lower dimensions. Here we will use a relatively new technique called UMAP (Uniform Manifold Approximation and Projection).

import umap

X\_topics = svd\_model.fit\_transform(X)

embedding = umap.UMAP(n\_neighbors=150, min\_dist=0.5, random\_state=12).fit\_transform(X\_topics)

plt.figure(figsize=(7,5))

plt.scatter(embedding[:, 0], embedding[:, 1],

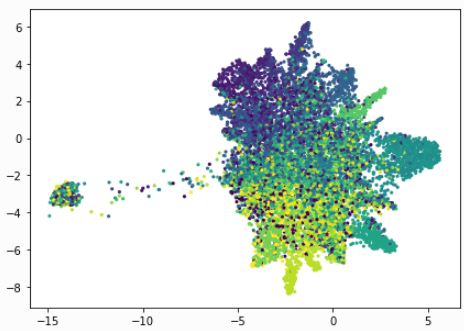
c = dataset.target,

s = 10, # size

edgecolor='none'

)

plt.show()



As you can see above, the result is quite beautiful. Each dot represents a document and the colours represent the 20 newsgroups. Our LSA model seems to have done a good job. Feel free to play around with the parameters of UMAP to see how the plot changes its shape.

## **Pros and Cons of LSA**

Latent Semantic Analysis can be very useful as we saw above, but it does have its limitations. It’s important to understand both the sides of LSA so you have an idea of when to leverage it and when to try something else.

****Pros:****

* LSA is fast and easy to implement.
* It gives decent results, much better than a plain vector space model.

****Cons:****

* Since it is a linear model, it might not do well on datasets with non-linear dependencies.
* LSA assumes a Gaussian distribution of the terms in the documents, which may not be true for all problems.
* LSA involves SVD, which is computationally intensive and hard to update as new data comes up.