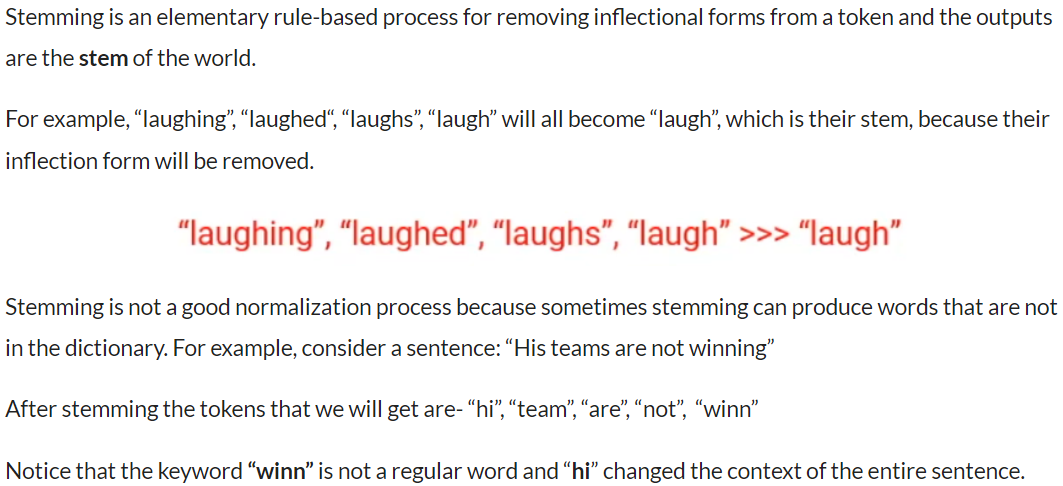
Tokenization:

-White-space Tokenization

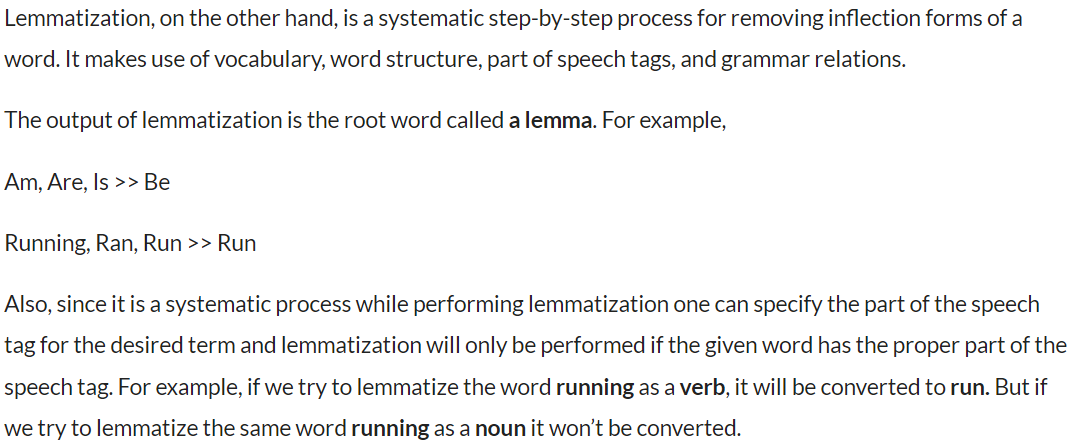
-Regular Expression Tokenizations

Normalization:

-Stemming

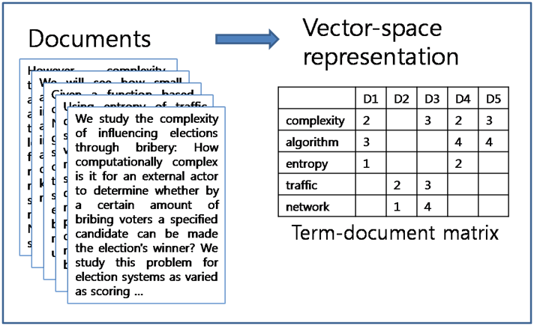


-Lemmatization

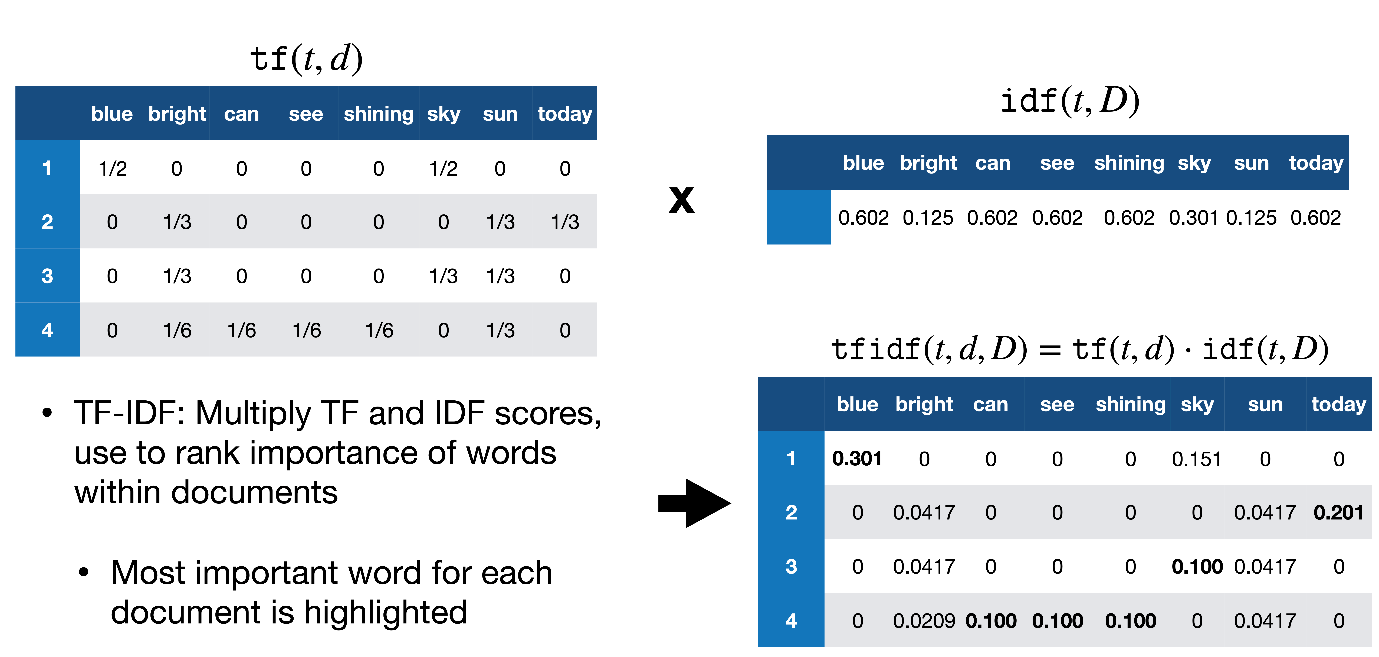


The Process:

We can get repeated words within our document. A better representation is a vector form, that can tell us how many times each word can occur in a document. The following is called a document term matrix and is shown below:



We then divide the term frequency by the document frequency of that word. This helps us with the frequency of occurrence of terms in that document and inverse to the number of documents it appears in. Thus we have the TF-IDF. The idea is to assign particular weights to words that tell us about how important they are in the document.



**Term Frequency (TF):**

Suppose we have a set of English text documents and wish to rank which document is most relevant to the query , “Data Science is awesome !” A simple way to start out is by eliminating documents that do not contain all three words “Data”,”is”, “Science”, and “awesome”, but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its *term frequency*.

T***he weight of a term that occurs in a document is simply proportional to the term frequency.***

**Formula :**

*tf(t,d) = count of t in d / number of words in d*

**Document Frequency :**

Thismeasures the importance of document in whole set of corpus, this is very similar to TF. The only difference is that TF is frequency counter for a term t in document d, where as DF is the count of **occurrences** of term t in the document set N. In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term consists in the document at least once, we do not need to know the number of times the term is present.

*df(t) = occurrence of t in documents*

**Inverse Document Frequency(IDF):**

While computing TF, all terms are considered equally important. However it is known that certain terms, such as “is”, “of”, and “that”, may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing IDF, an *inverse document frequency* factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

IDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because stop words such as “is” is present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage.

*idf(t) = N/df*

Now there are few other problems with the IDF , in case of a large corpus,say 100,000,000 , the IDF value explodes , to avoid the effect we take the log of idf .

During the query time, when a word which is not in vocab occurs, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator.

that’s the final formula:

**Formula :**

*idf(t) = log(N/(df + 1))*

tf-idf now is a the right measure to evaluate how important a word is to a document in a collection or corpus.here are many different variations of TF-IDF but for now let us concentrate on the this basic version.

**Formula :**

*tf-idf(t, d) = tf(t, d) \* log(N/(df + 1))*