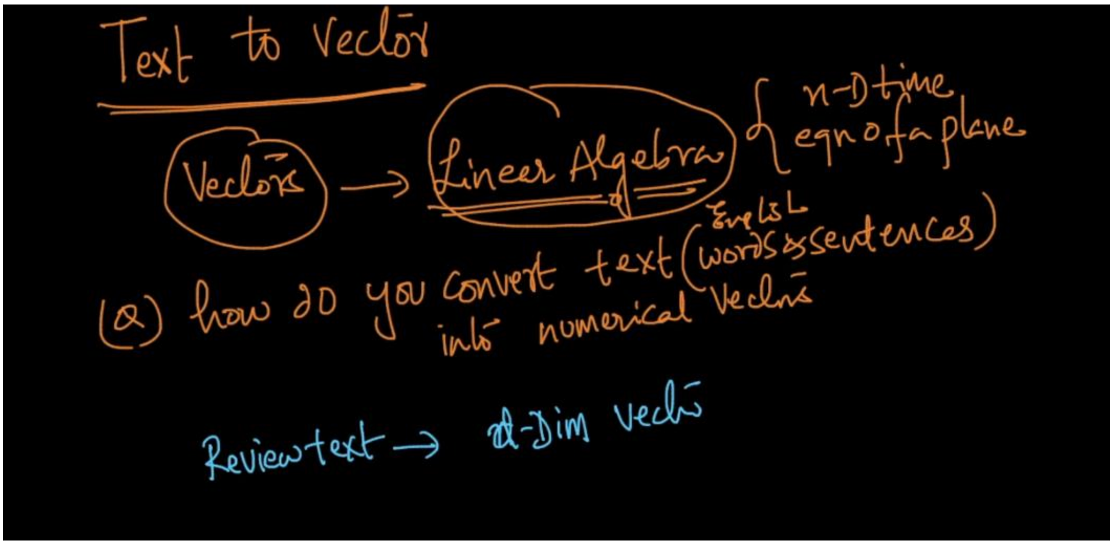
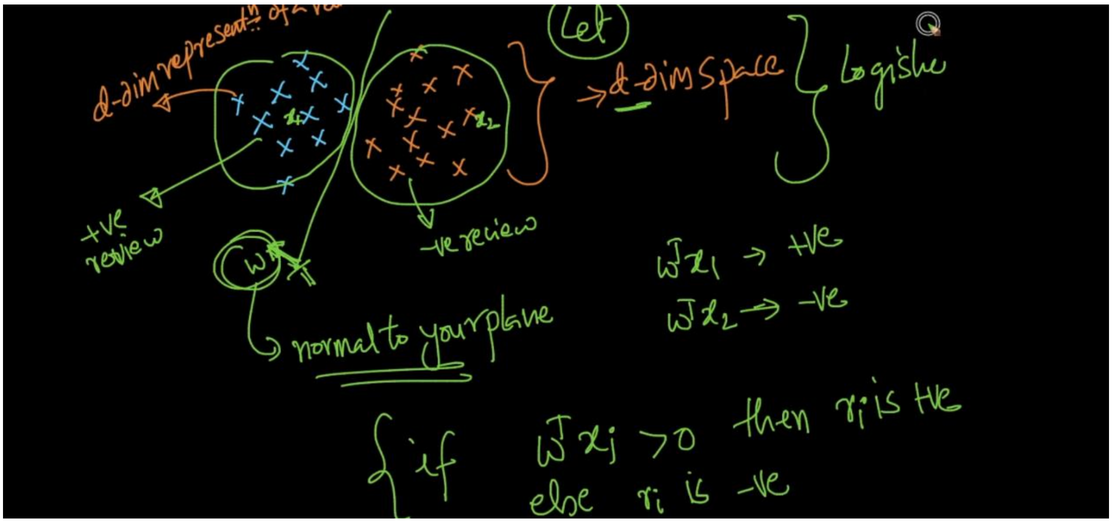
Basic Machine Learning

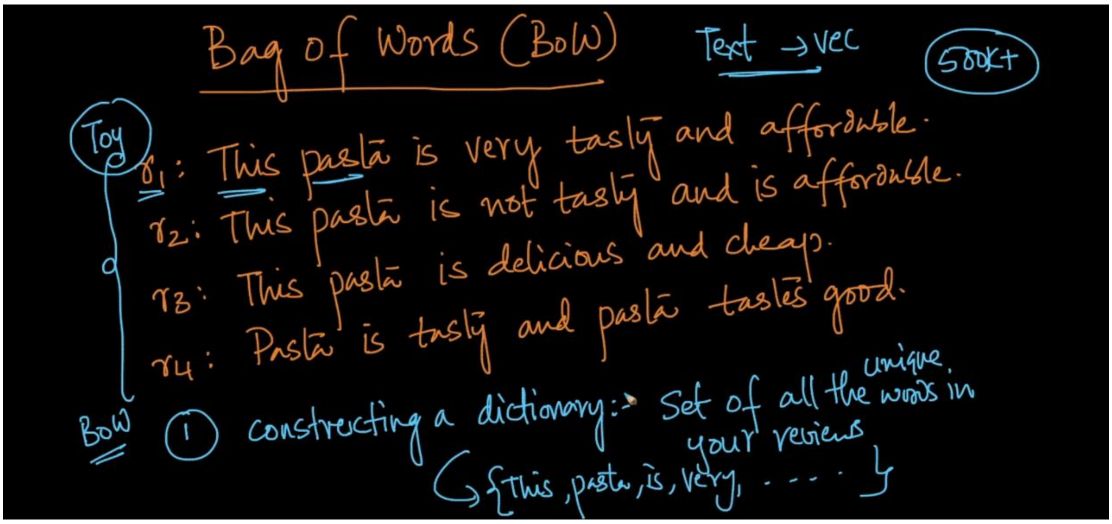


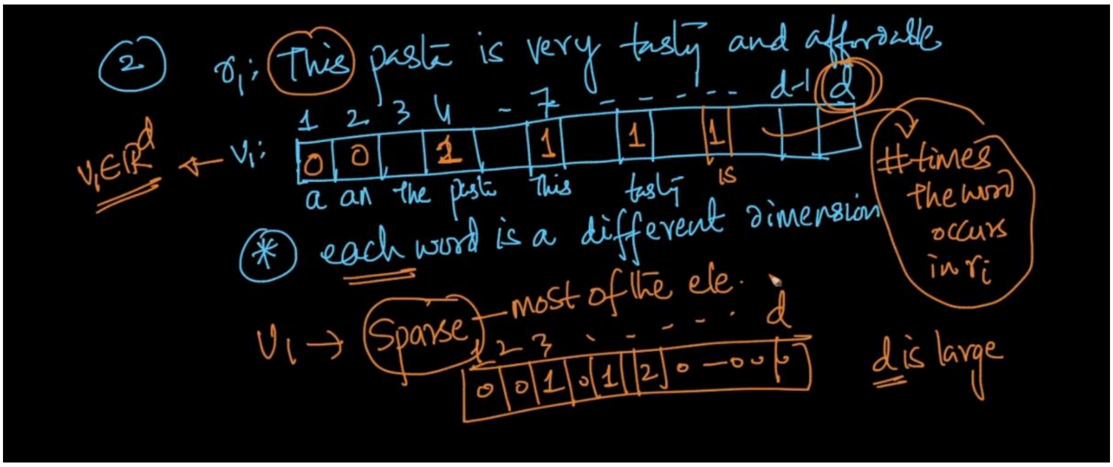
After data clean this is next imp step once we convert text to vector we can use all power of linear algebra .



See after convert text to vector we can make classification very easily in + and - review

By converting it to vector we can find min distance so we get similarity between vectors so we can easily group them and separate from each other .

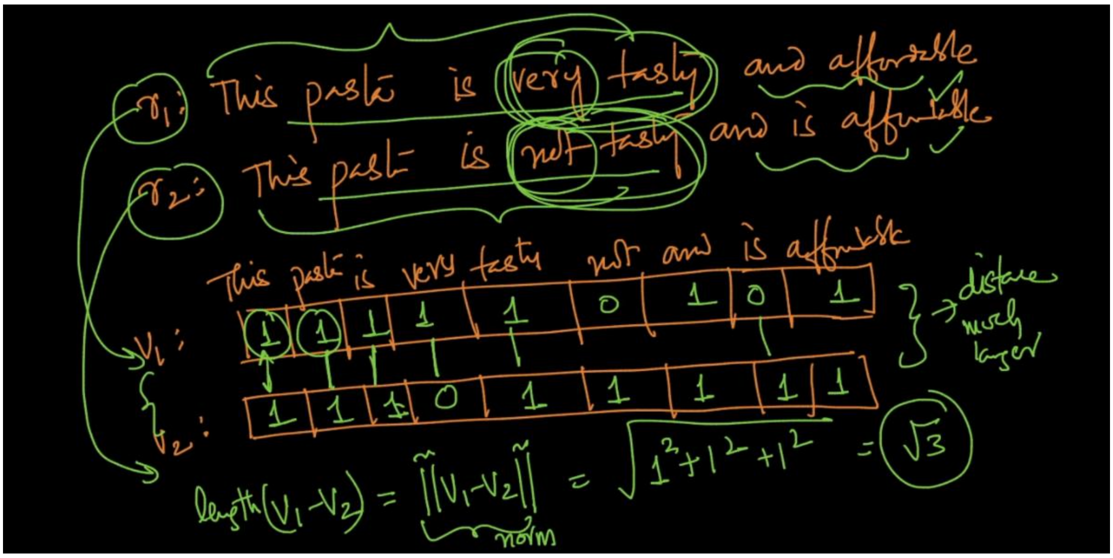




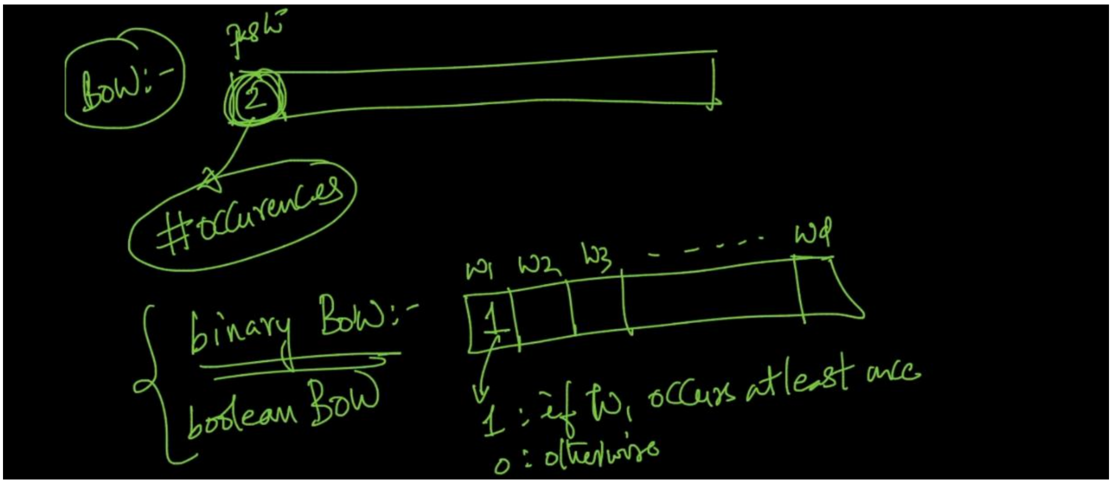
Problem of BOW is lets say I have 500k reviews now from that I get 10k unique words so I for each review I have vector of size 10k its called sparse vector means lots of values will zero .

Lets say new review is I like pasta . and now I have 10k unique words so for above example we create new vector of size 10k in that only 3 values for I like pasta will be 1 and others are 0 .

Now lets take real example

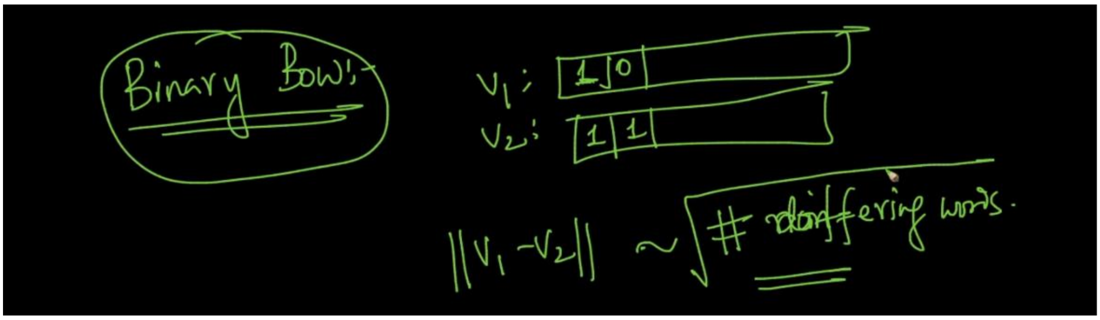


If you see above 2 example are totally opposite but still diff between then is very less and this is drawback of BOW .

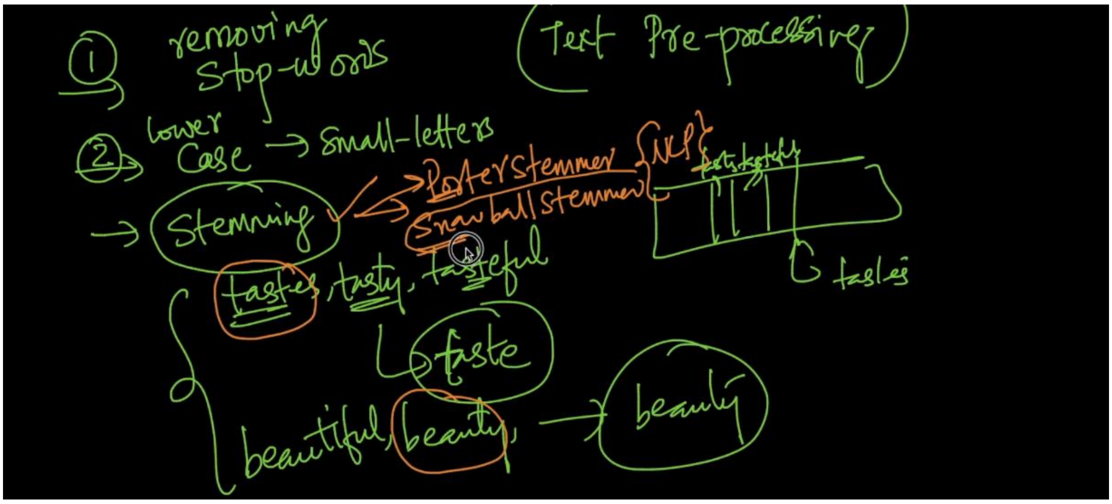


Above is another type its binary BOW used only 0 and 1 not word count .

Binary BOW tell no of different word in a sentence .

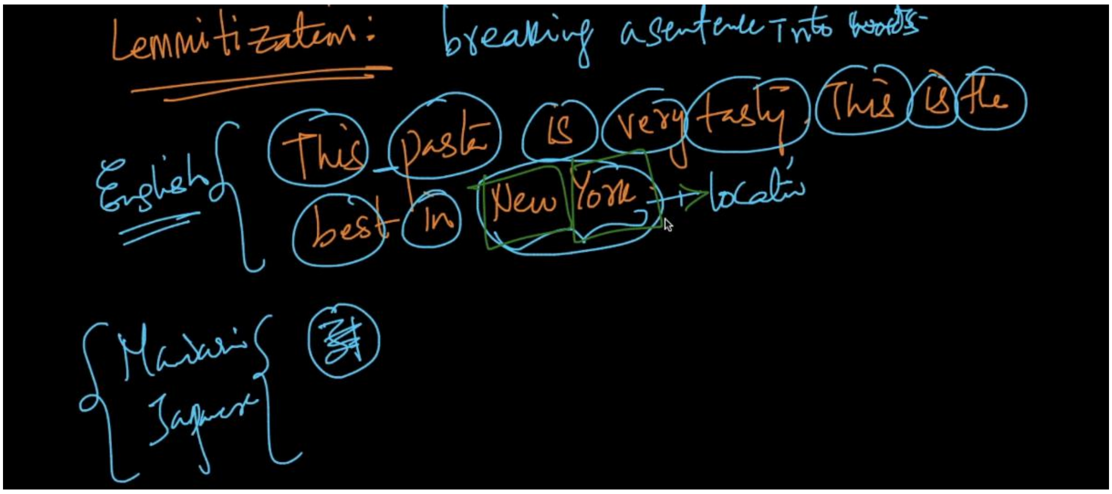


Text processing :

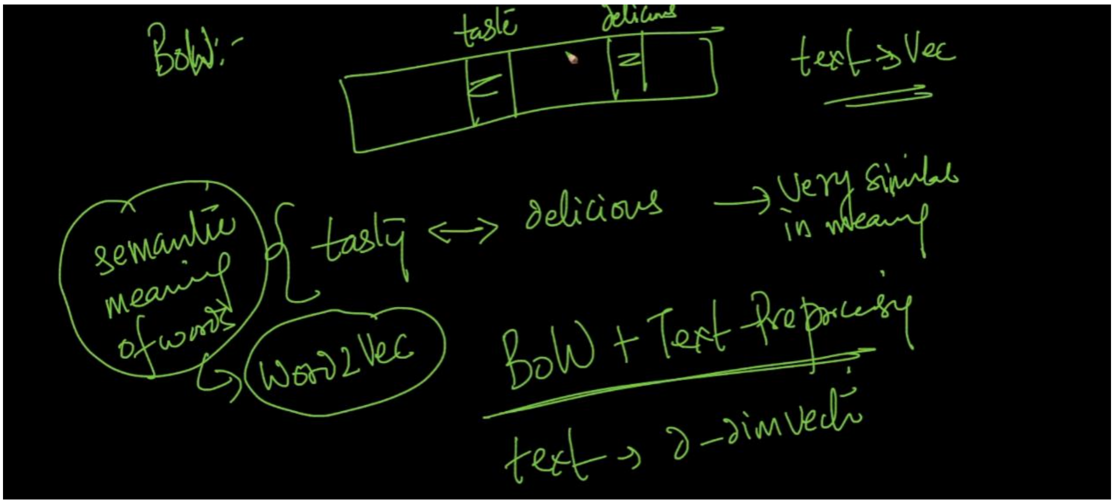


1st is remove stop words
2nd is lower case

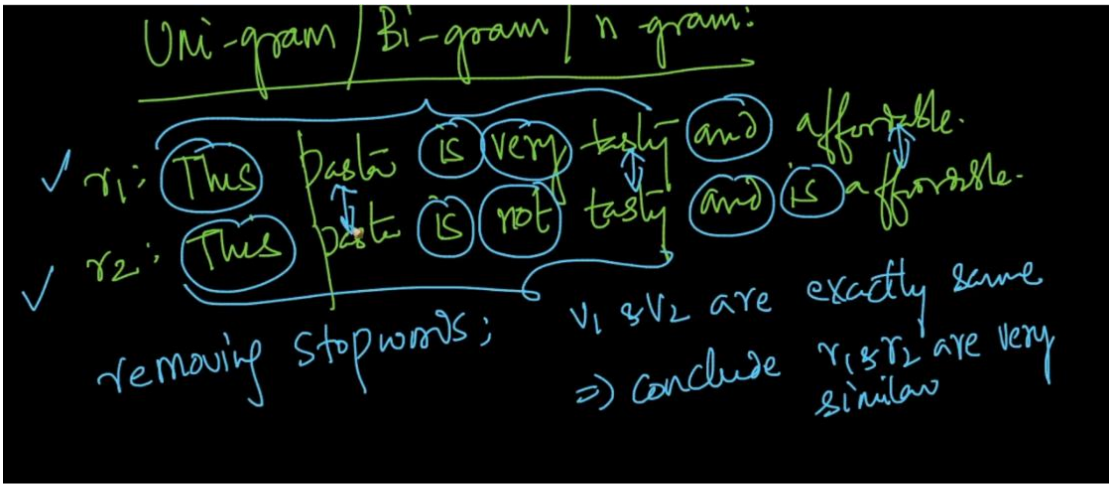
3rd is stemming means steam word to original form .
4th is Lemmitization which is used to break statement into words .

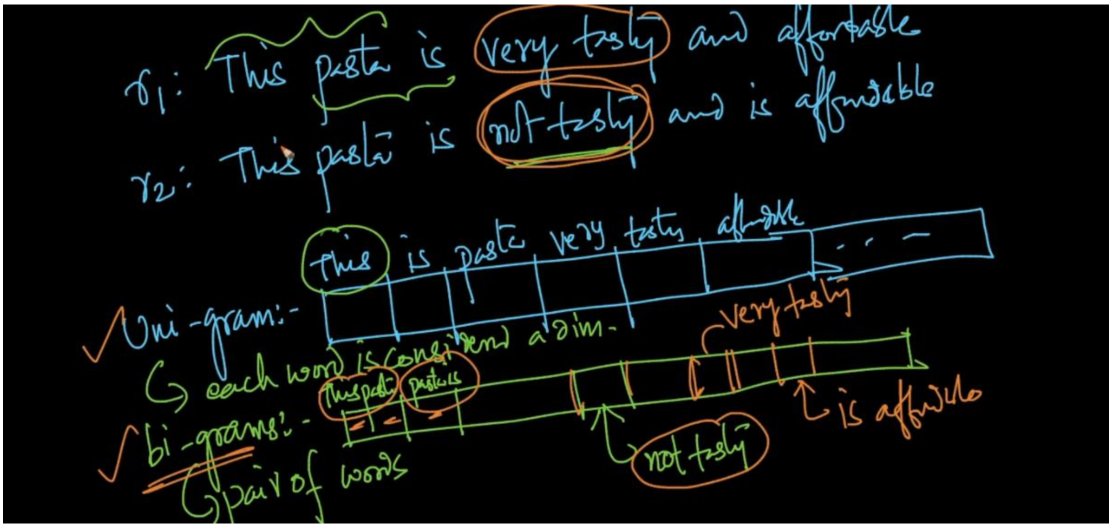


Here problem is we are not caring about similar words like tasty and delicious both are same but we taking as different in BOW ..

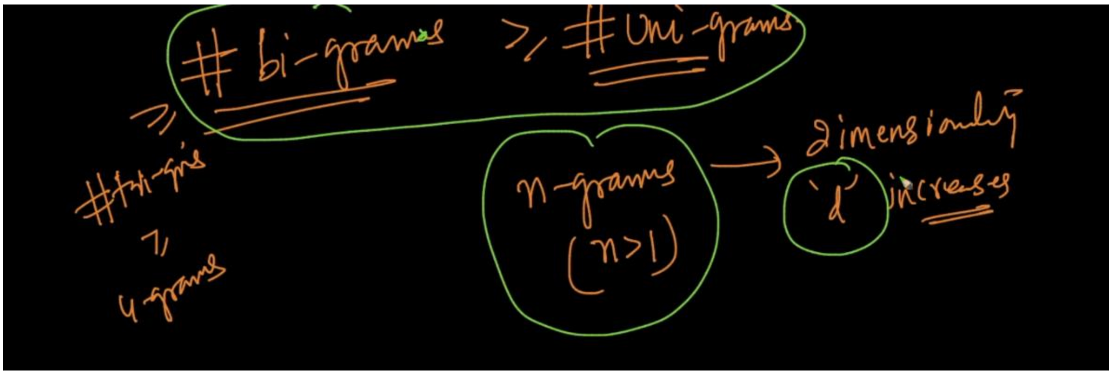


N - Grams and Bi-Grams :





See bi grams can retain some info like not test ,very testy with different dimension . that’s why they are important .



Bi grams > Tr -Grams so on so forth this will increase dimension of vector will see problem of that but still this will help to preserve meaning of data ..

i am Abhijeet

Bi gram :

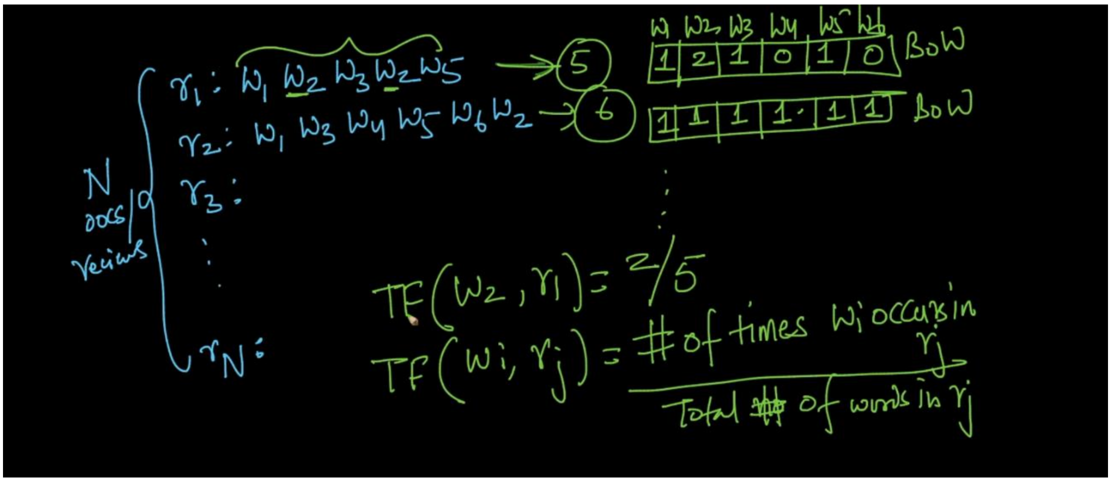
i am

am abhijeet

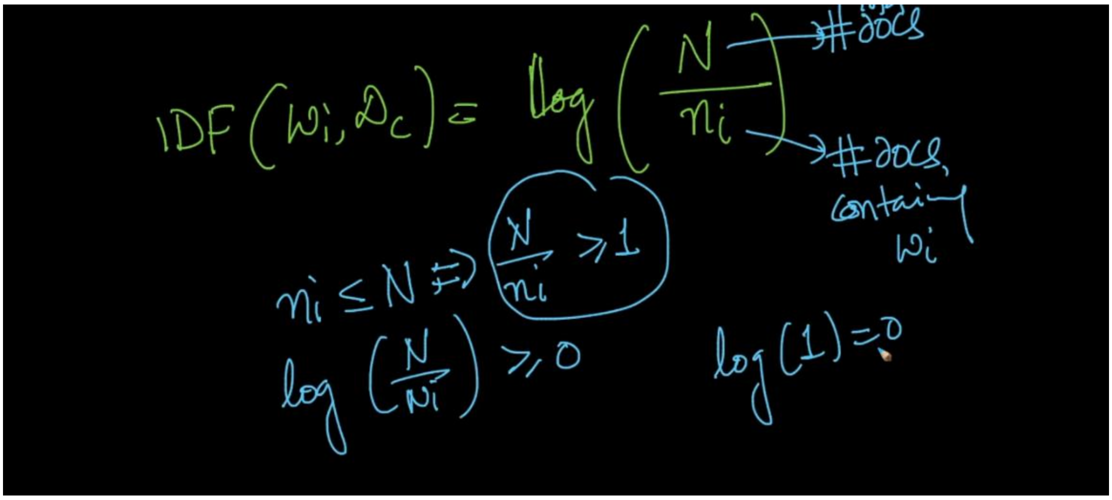
Tr Gram

i am abhijeet

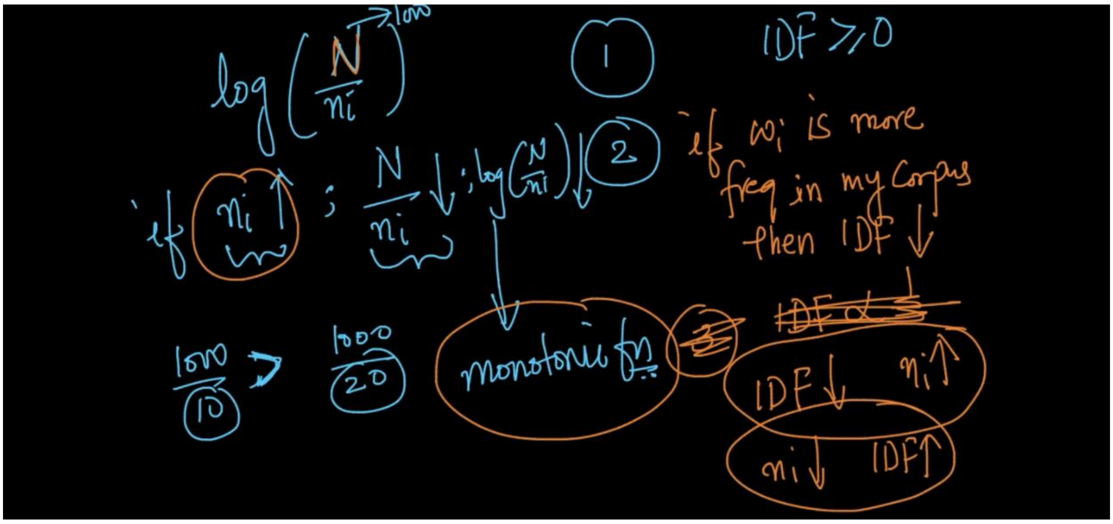
TF IDF :



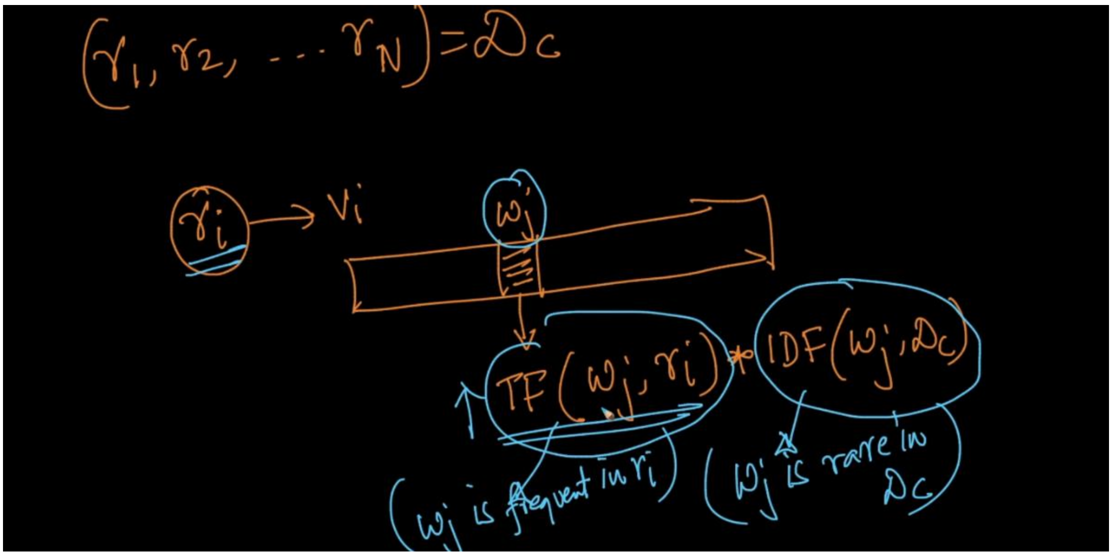
TF Will tell probability of word . we will check how much time a word occurs In as sentence if word occurs lots of time then get max TF value .



N = no of documents means no of review .
Ni = means number of review that word is present ..



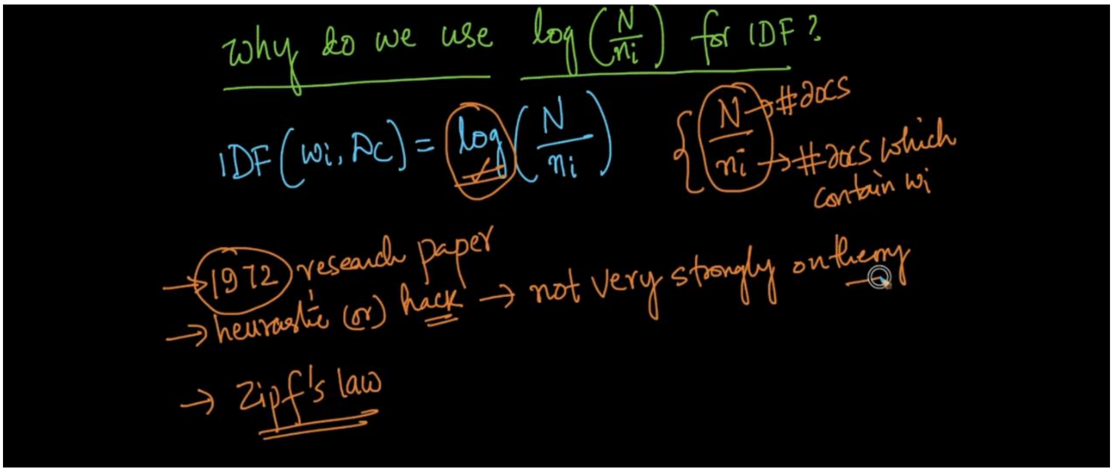
IDF will be low when given word occurs very less in whole data set . That means IDF will give more weight to a rare word .

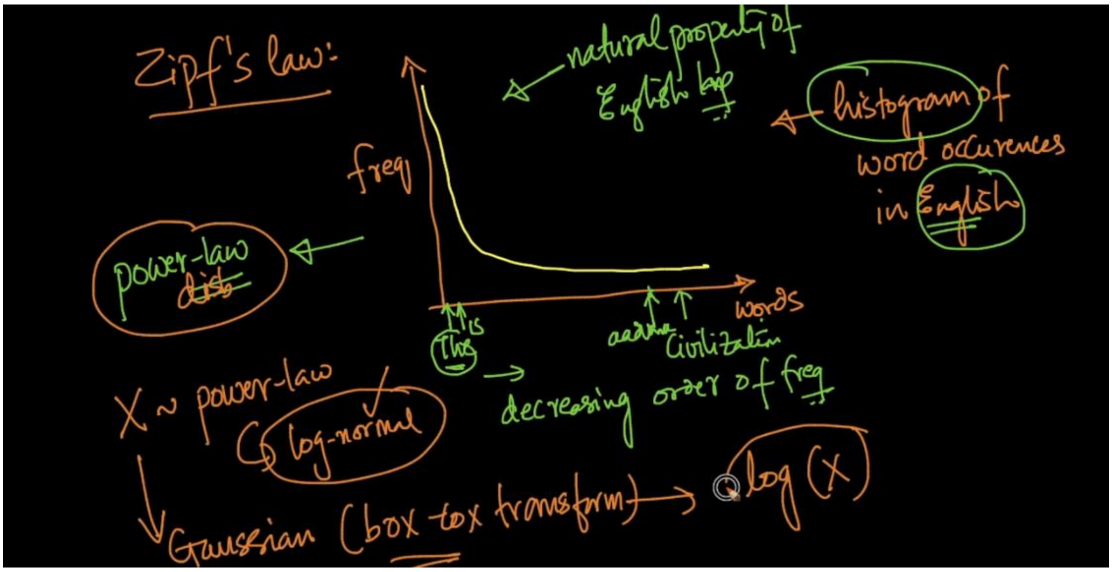


TF will high if word occurs frequently in a review . IDF will be high if word rare in whole review data set .

But this will also not solve problem of semantic words like BOW .

Why use Log in TF IDF :

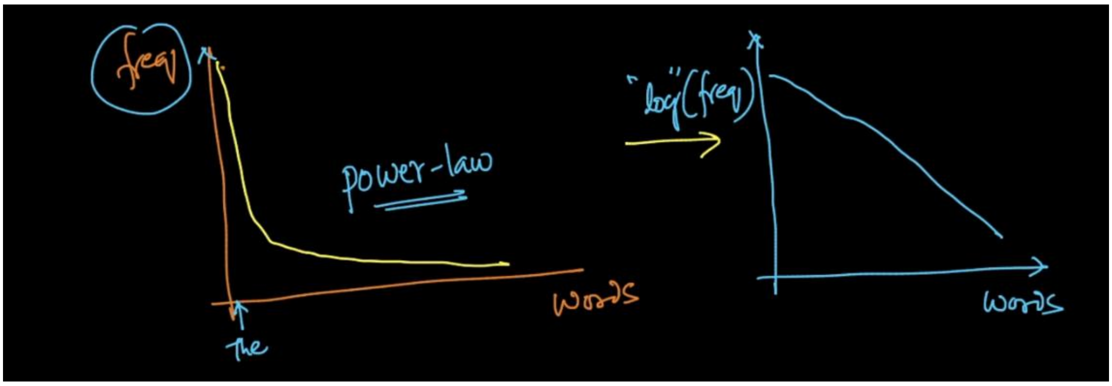




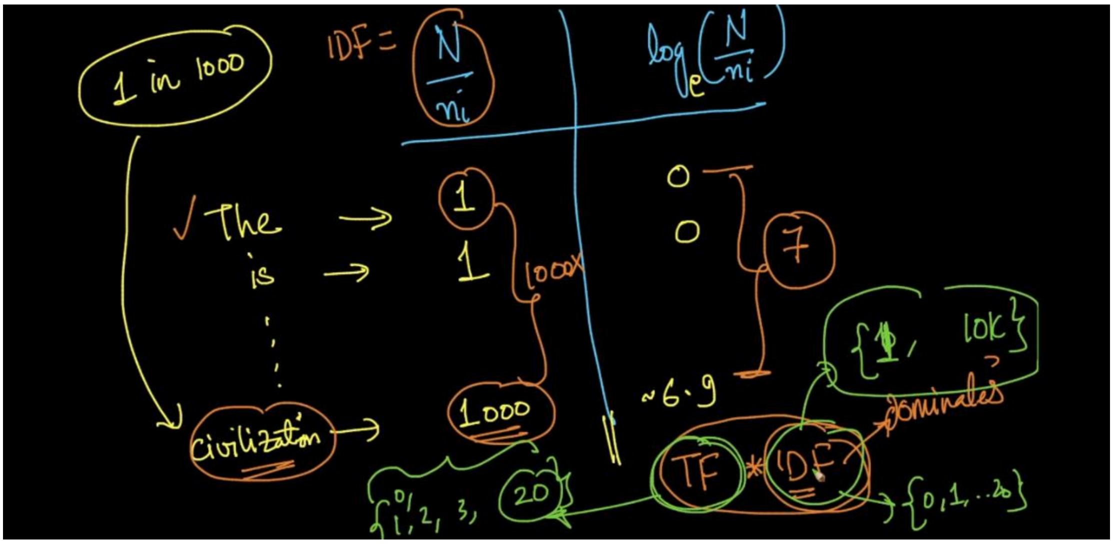
See that graph Is like power dist . words like the occurs more time than word like civilization .

So as we study in power dist we can convert it into normal dist by taking log of it .

Because of log we get straight line see below image . and that is much more manageable .



Another reason check below image .



See if we remove log what value we get for IDF for word like The is 1,is =1 …. civilization =1

.

Now check another side if we add log we get value 0 for the words like the,is etc .. and .

If we see civil word we get IDF = 1000 this is very much high and if multiply this value with TF we will get very unexpected result so we use log to keep value in well range .

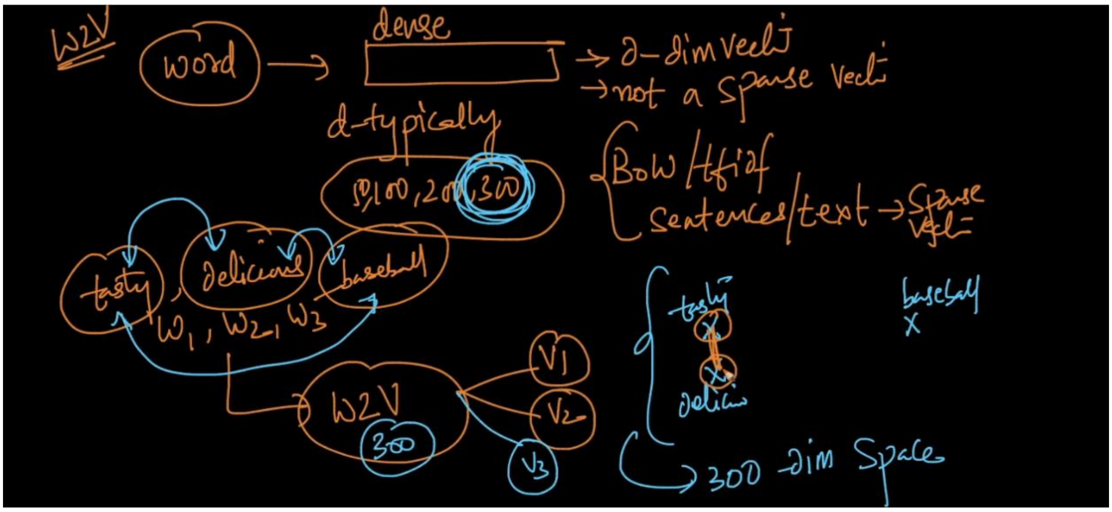
Word2Vec :

Now we study BOW ,TF IDF and both technique not consider semantic meaning of words .

Word2Vec will solve that problem .

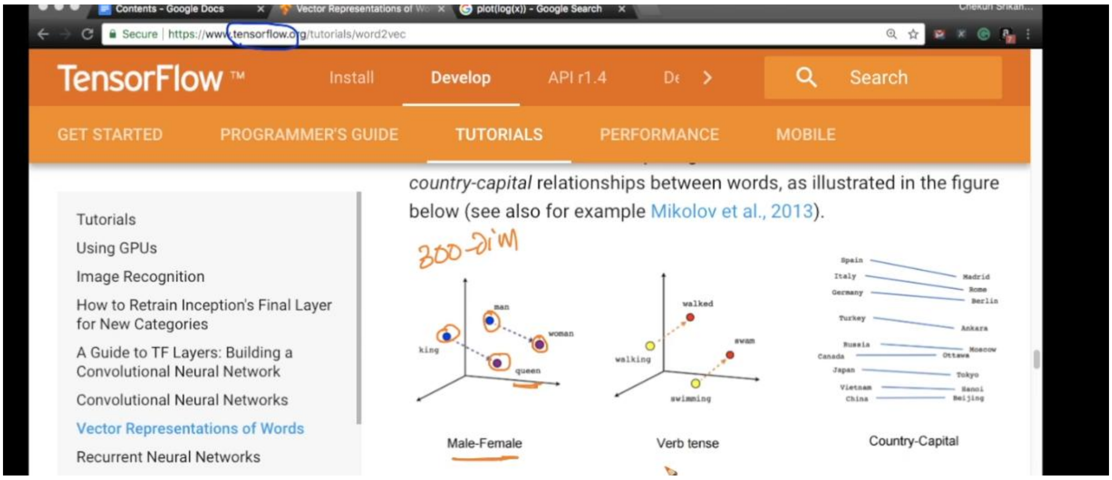
It will not work for sentence like BOW it will create vectors for words .

As much as dimension our accuracy will be more but to take 300 dimension we need millions of data .

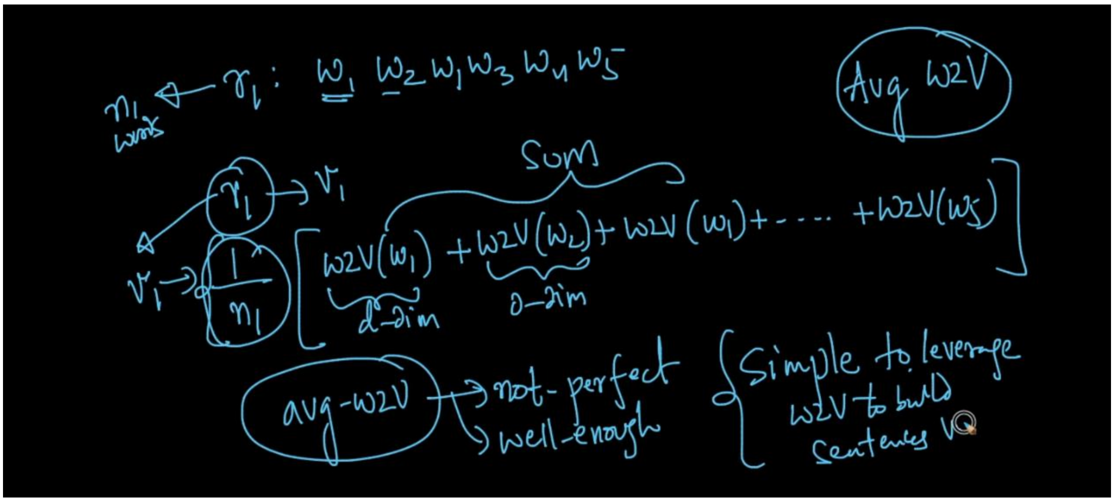


What it will do is see above image
tasty ,delicious ,ball are three words so word2vec will create 3 vectors lets say v1 ,v2 and v3 and will see how close they are .

Like that they will understand meaning of semantic words .



Avg-Word2Vec :



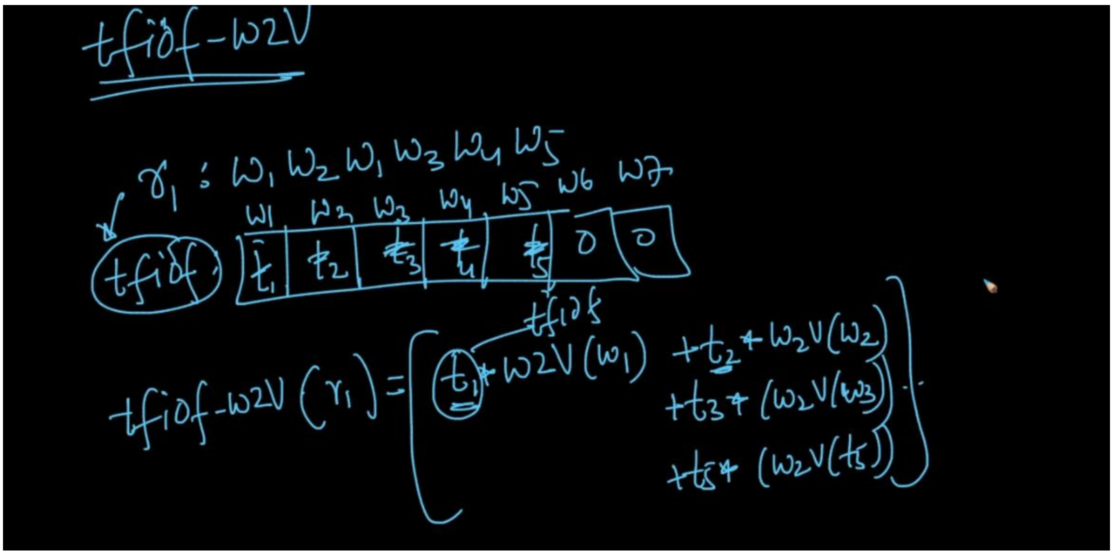
Here lets say we have one review and in that 5 words so we can convert this sentence in word2vec using above technique .

What we do is we will find word2vec for all individual words in a sentence and then we will add all that words vectors and then we divide it by n1 = no of words (5 in our case) .

TF IDF Word2Vec :

1. Here we will find TF IDF for lets say review 1 .
2. Then we will find Word2vec for all words in a sentence or review 1 .

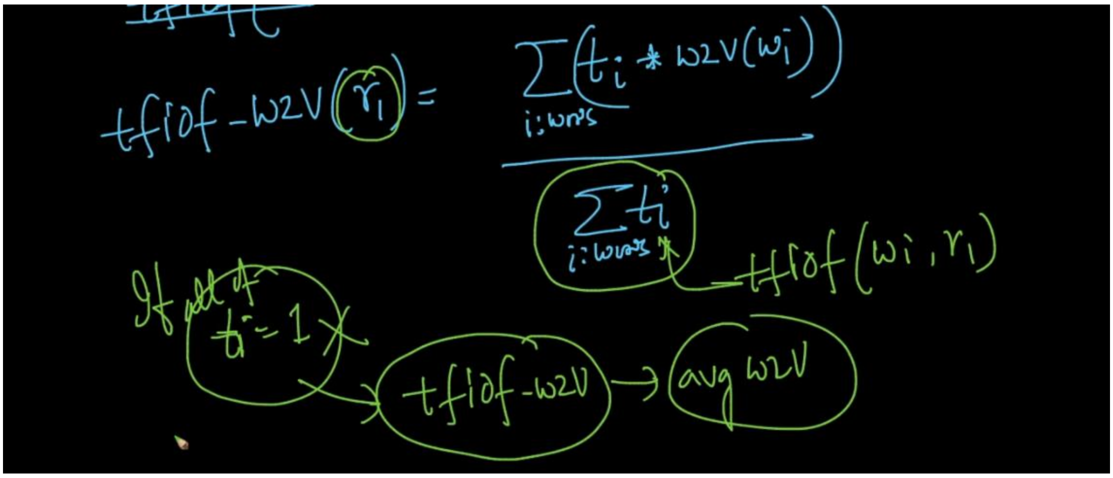
Then do multiplication in step 1 and 2 .



See above image we convert sentence into TF IDF t1,t2…. t5 .

Then we are taking word2vec for 5 words then simply we multiply each word TF IDF with word2vec .

See below formula .



BOW code example in Notebook :

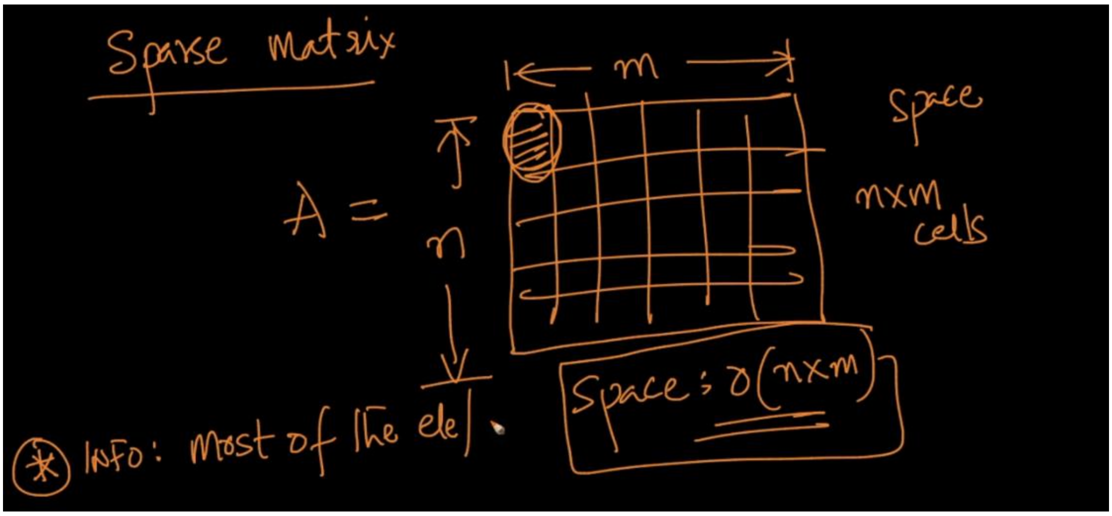
Now we learn concept of Sparse matrix .

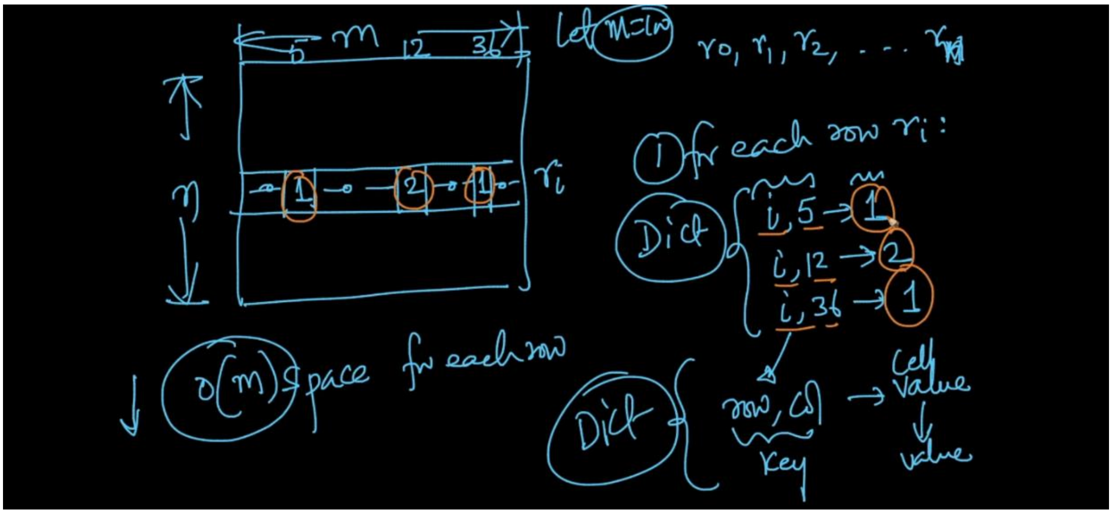
We know that sparse matrices learned in BOW chapter the matrix which has lots of 0 value are called sparse Matrices .

So this will increase space complexity too much because we store lots of non important values in a matrix .

Space complexity = O(n*m)

Now we want to reduce this as much as possible and we can do this using sparse matrix .





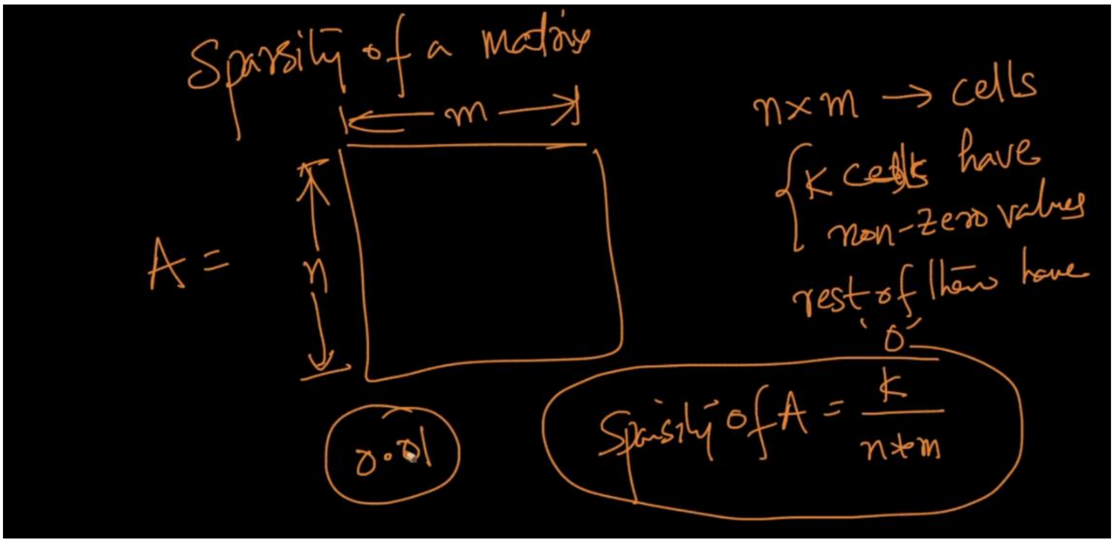
Just see above image what we are doing here is we are removing 0 values column .

Lets say we m = 100 columns in our data set . and n rows . now lets take row n5 and in that row only 3 values are non zero and rest 97 are 0 ..

So we will create new matrix with row,col,value .

So here we are reducing almost 10 times of matrix space how ?

Now we 3 column only and 3 values so we are storing only 9 values instead of 100 values this is called sparse matrix .



As low as sparsity that means we have too much efficient sparse matrix representation .

Means we have only 1% of values which are 0 values in above image .

Text Processing Code in notebook :

In n gram as n increases dimension also increases ..

```
final_bigram_counts = count_vect.fit_transform(final['Text'].values)

In [81]: final_bigram_counts.get_shape()
Out[81]: (364171, 2910192)
```

2.9M Dim

only Unigrams → 115K dim

See image after bi gram shape increases to 2.9 M values and for uni gram it was 115 k only .