**TF-IDF/Term Frequency Technique:**

OR How to find meaning of sentences and documents

TF-IDF or ( Term Frequency(TF) — Inverse Dense Frequency(IDF) )is a technique which is used to find meaning of sentences consisting of words and cancels out the incapabilities of Bag of Words technique which is good for text classification or for helping a machine read words in numbers. However, it just blows up in your face when you ask it to understand the meaning of the sentence or the document.

Let’s say a machine is trying to understand meaning of this —

***Today is a beautiful day***

This sentence talks about **today**, it also tells us that today is a **beautiful day**. The mood is **happy/positive**, anything else cowboy?

Beauty is clearly the adjective word used here. From a BoW approach all words are broken into count and frequency with no preference to a word in particular, all words have same frequency here (1 in this case)and obviously there is no emphasis on beauty or positive mood by the machine.

The words are just broken down and if we were talking about importance, ‘a’ is as important as ‘day’ or ‘beauty’.

But is it really that ‘a’ tells you more about context of a sentence compared to ‘beauty’ ?

No, that’s why Bag of words needed an upgrade.

Also, another major drawback is say a document has 200 words, out of which ‘a’ comes 20 times, ‘the’ comes 15 times etc.

Many words which are repeated again and again are given more importance in final feature building and we miss out on context of less repeated but important words like Rain, beauty, subway , names.

So it’s easy to miss on what was meant by the writer if read by a machine and it presents a problem that TF-IDF solves, so now we know **why do we use TF-IDF.**

**Let’s now see how does it work, okay?**

TF-IDF is useful in solving the major drawbacks of Bag of words by introducing an important concept called **inverse document frequency.**

It’s a score which the machine keeps where it is evaluates the words used in a sentence and measures it’s usage compared to words used in the entire document. In other words, it’s a score to highlight each word’s relevance in the entire document. It’s calculated as -

*IDF =Log[(# Number of documents) / (Number of documents containing the word)] and*

*TF = (Number of repetitions of word in a document) / (# of words in a document)*

okay, for now let’s just say that TF answers questions like — **how many times is beauty used in that entire document**, give me a probability and IDF answers questions like how important is the word beauty in the entire list of documents, **is it a common theme in all the documents**.

So using TF and IDF machine makes sense of important words in a document and important words throughout all documents.

**Answer me this —**

*Imagine there’s a document full of sentences, what is the best way to break it so that a machine can make some sense of what it is ?*

*1. Break it in words*

*2. Break it in letters*

*3. Break it in sentences*

*4. Break it in bytes*

Can you answer it ?

**Times up.**

*The current answer is option 3*. Break it in sentences .

Why ? cuz when you break a document in multiple sentences, each sentence has multiple words which represent provide some context to sentences and these sentences as a whole provide some context to the document and then we can ask the machine questions like,

what documents are similar to each other Siri?

By evaluating TF-IDF or a number of *“the words used in a sentence vs words used in overall document”,*we understand -

1. how useful a word is to a sentence (which helps us understand the importance of a word in a sentence).
2. how useful a word is to a document (which helps us understand the important words with more frequencies in a document).
3. helps us ignore words that are misspelled (using n-gram technique) an example of which I am covering below

Imagine in a document you misspelled ‘example’ as ‘exaple’ and you forgot to go back and change it before giving it to a machine to read -

In case of BOW, both ‘example’ and ‘exaple’ would be treated as different words and given the same importance because their frequency is same.

But in case of TD-IDF because of a score of IDF, this mistake is corrected because we know example as a word is more important than exaple, so we treat it like a non useful word.

Now because of these scores our machine has a better understanding of these documents and can be asked to compare these documents, find similar documents, find opposite documents, find similarities in document and can be used by machine to recommend you what to read next, cool right?

Now, I am guessing you need a minute to go back and grasp this concept again before I tell you how to do it, ofcourse I’ll take up an example so if you’re conceptually hazy but almost clear you’ll definitelly be alright once you practise with the example.

**What is the way of finding TF-IDF of a document?**

The process to find meaning of documents using TF-IDF is very similar to Bag of words,

1. Clean data / Preprocessing — Clean data (standardise data) , Normalize data( all lower case) , lemmatize data ( all words to root words ).
2. Tokenize words with frequency
3. Find TF for words
4. Find IDF for words
5. Vectorize vocab

**Let’s cover an example of 3 documents -**

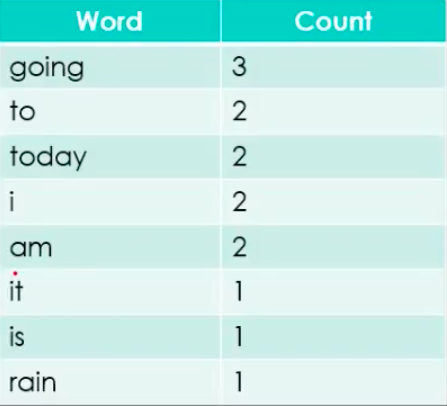
**Document 1**It is going to rain today.

**Document 2**Today I am not going outside.

**Document 3**I am going to watch the season premiere.

To find TF-IDF we need to perform the steps we laid out above, let’s get to it.

**Step 1 Clean data and Tokenize**



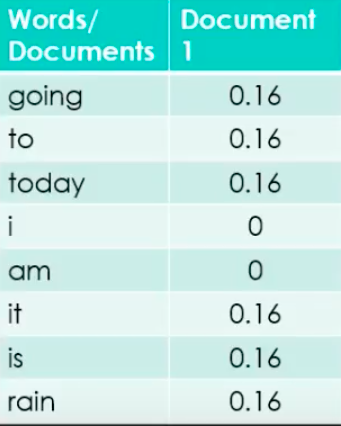
Vocab of document

**Step 2 Find TF**

**Document 1**—

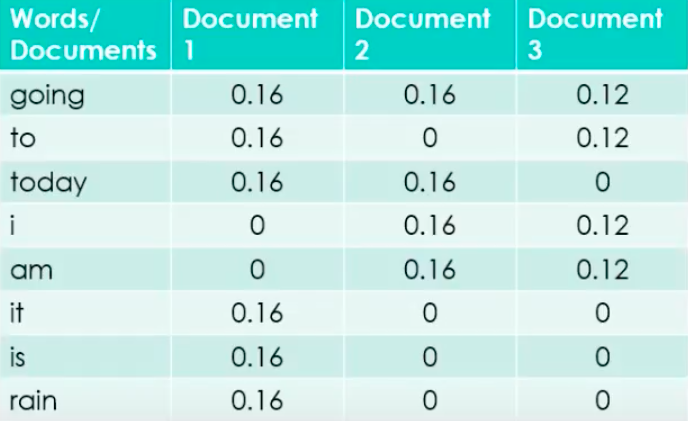
*It is going to rain today.*

Find it’s TF = (Number of repetitions of word in a document) / (# of words in a document)



TF for sentence 1

Continue for rest of sentences -



TF for the document

**Step 3 Find IDF**

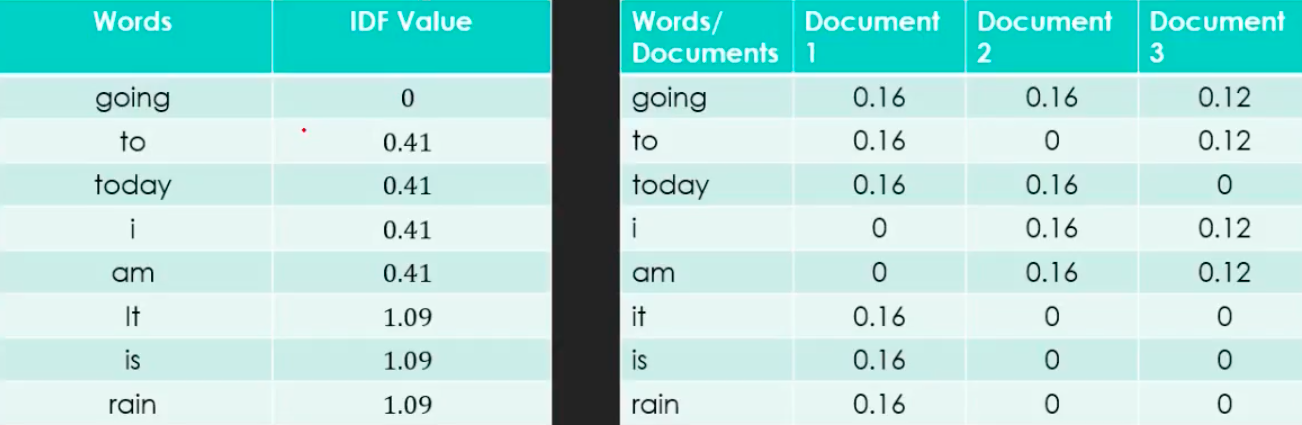
Find IDF for documents (we do this for feature names only/ vocab words which have no stop words )

IDF =Log[(Number of documents) / (Number of documents containing the word)]



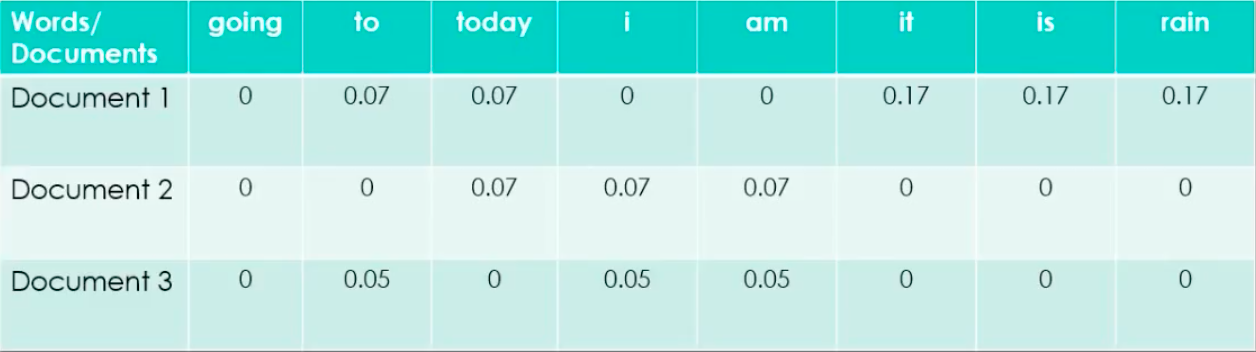
IDF for document

**Step 4 Build model i.e. stack all words next to each other —**



IDF Value and TF value of 3 documents.

**Step 5 Compare results and use table to ask questions**



Remember, the final equation = TF-IDF = TF \* IDF

You can easily see using this table that words like **‘it’,’is’,’rain’** are important for document 1 but not for document 2 and document 3 which means Document 1 and 2&3 are different w.r.t talking about rain.

You can also say that Document 1 and 2 talk about something **‘today’**, and document 2 and 3 discuss something about the writer because of the word **‘I’.**

This table helps you find similarities and non similarities btw documents, words and more much much better than BOW.