Introduction to Natural Language Processing

Course Structure

- Live session → Every Sunday (1100 - 1300)
- Hands on Practical Active Learning Method
- All Course Material including Lecture Slides, Recordings and Working Notebooks will be Provided and Assignments after end of class
- Learning Quiz and Assignment(s) will be shared
- Course Qualification Certificate depends on Attendance + Performance in Quizzes, Assignments and Projects
- Communication through Slack Channel. Link

Course Qualifying Criteria

Attendance

- 75% of attendance
- **Note:** The learners of the course must attend 75% of live sessions to qualify for the course certificate.

Evaluation/ Submission of Assessments

- Learning checks in class
- Performance in assignments and learning checks
- Submission of assignments/projects
- **Note:** To qualify for the certificate it is mandatory to pass the assessment with 60% of the grade

Revoking/Removal:

- In the following cases, learners will be removed from the course
- Two weeks of consecutive absences from the live session without any prior notice
- Poor performance in assignments and quizzes
- Not submission of assignments within due dates

Introduction to course content

Python Foundational Element

Introduction to Python

Understanding Python as a programming language

Variables and Data Types

Exploring different data types in Python (integers, floats, strings, etc.)

Expressions and Operators

Understanding Python's expression syntax and operators

Basic Input and Output

Obtain and display data in Python

Python Programming Logic

Conditional Statements

Learning how to use if, elif, and else statements for decision-making.

Loops (For and While)

Understanding loops for repetitive tasks.

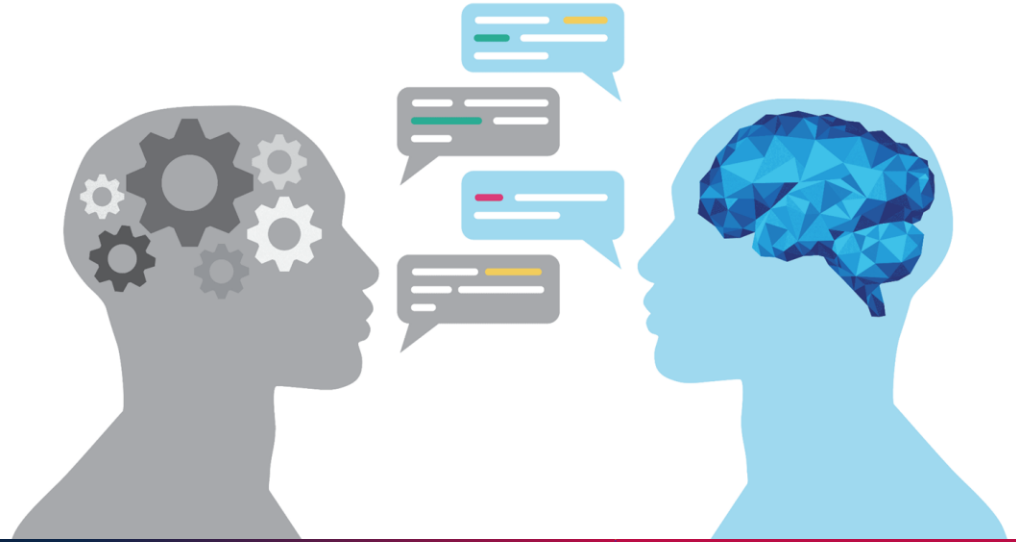
Functions and Modularity

Defining and using functions to organize code.

Introduction to Objects and Classes

Exploring object-oriented programming concepts in Python.

What is NLP?
Discussion Time(5 mins)



I. 1.Definition of Natural Language Processing

NLP is the field of study that focuses on how computers understand,, and process human language in a way that is similar to how humans do. Juinterpretst like how we, as humans, use language to communicate, express thoughts, and convey meaning, NLP enables computers to perform similar tasks with human language.

Applications of NLP:

It involves tasks such as **text classification**, **sentiment analysis**, **machine translation**, named entity recognition, and many more. NLP technology is used in various applications like chatbots, voice assistants, language translation, social media analysis, and sentiment analysis, among others..

2.Tokenization

Tokenization, in the realm of Natural Language Processing (NLP) and machine learning, refers to the process of converting a sequence of text into smaller parts, known as tokens. These tokens can be as small as characters or as long as words.
Breaking text into smaller units (e.g., letters, words, phrases, characters) for computer processing.

There are some types of tokenization:

2.1 Letter Tokenization

Letters can be represented by numbers using an encoding scheme called ASCII.If we tokenize by letters, we might have same list of tokens for two of more words. So it makes hard to understand the sentiment of a word and would not convey the intended meaning of the sentence.

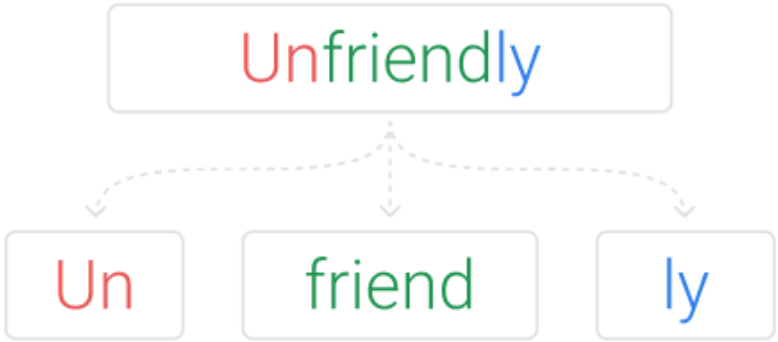
While tokenizing by letters, the word STOP & POST have same encoding just in different order, which makes hard to understand the sentiment of a word.

2.2 Word Tokenization

This is the most basic and commonly used type, where text is split into individual words based on whitespace or punctuation.

2.3 Subword tokenization.

Striking a balance between word and letter tokenization, this method breaks text into units that might be larger than a single character but smaller than a full word. For instance, "Unfriendly" could be tokenized into "Un" ,"friend" and "ly". This approach is especially useful for languages that form meaning by combining smaller units or when dealing with out-of-vocabulary words in NLP tasks.



Google colab notebook - [Tokenization]

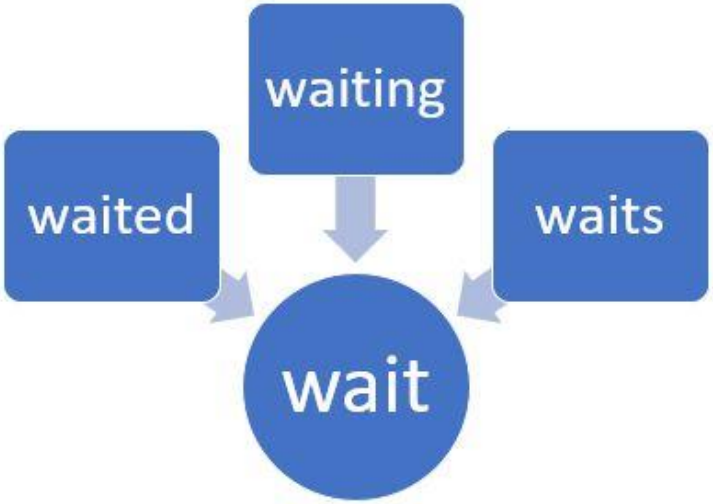
3.Stemming and Lemmatization

tesnsorflow / keras does not support

Task of reducing each word to its root . For example “Walk” is the root for words like “Walks”, “Walking”, “Walked”. Usually the root may hold significantly more meaning than the tense itself.

3.1 Stemming

Stemming is a method in text processing that eliminates prefixes and suffixes from words, transforming them into their fundamental or root form, The main objective of stemming is to streamline and standardize words, enhancing the effectiveness of the natural language processing tasks.It helps in reducing the vocabulary present in the documents, which saves a lot of computation.

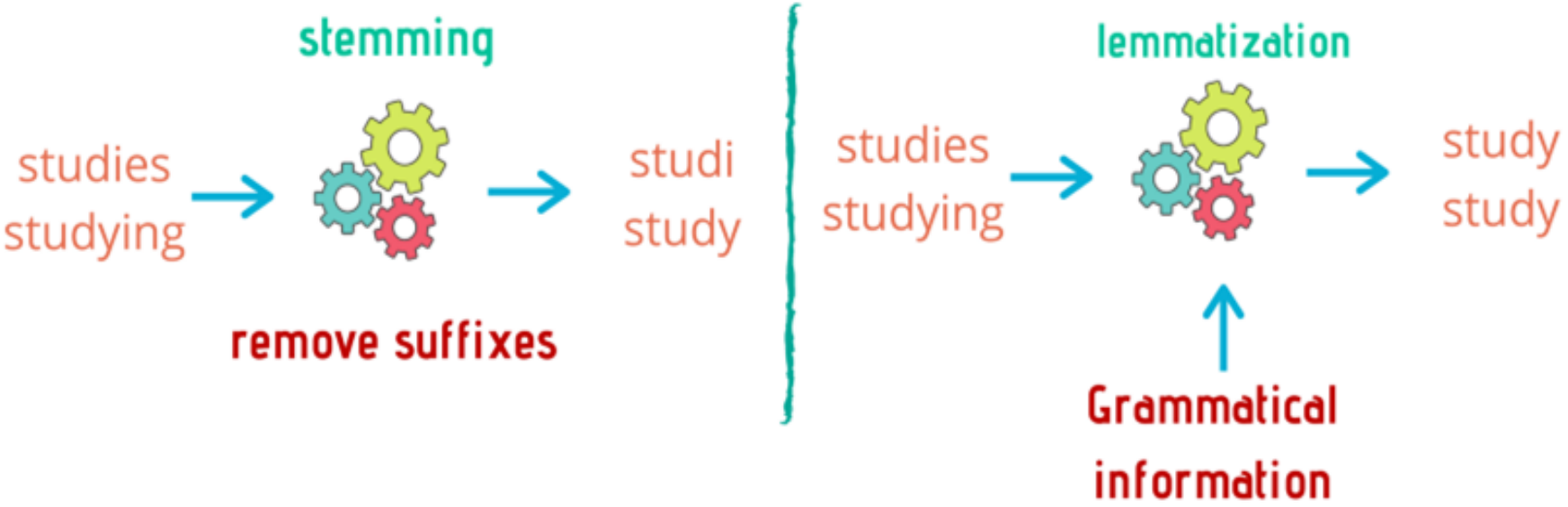


Stemming

3.2 Lemmatization

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. So, it links words with similar meanings to one word.

STEMMING VS. LEMMATIZATION



4.Stop Words

Stopwords are words that appear frequently in almost every document, contributing little semantic value. Examples include “The,” “is,” and “am.” These words may seem trivial, but they play a crucial role in shaping the efficiency and accuracy of NLP tasks.

Stopwords can be categorized into two main types:

1. Generic Stopwords: These are language-specific and ubiquitous words that are commonly found in various contexts. For the English language, examples include ‘a,’ ‘and,’ ‘the,’ ‘all,’ ‘do,’ and ‘so.’ They hold little significance on their own.
2. Domain-Specific Stopwords: These are words specific to a particular domain, such as education, health, sports, or politics. For instance, in the education domain, words like ‘paper,’ ‘class,’ ‘study,’ and ‘book’ might be considered domain-specific stopwords.

5. Bag-of-words (BOW)

We need a way to represent text data for machine learning algorithm and the bag-of-words model helps us to achieve that task. The bag-of-words model is simple to understand and implement. It is a way of extracting features from the text for use in machine learning algorithms.

The bag-of-words model, which allows us to represent text as numerical feature vectors.The two ideas behind the bag-of-words

1. We create a vocabulary of unique tokens
2. We construct a feature vector from each document that contains the counts of how often each word occurs in the particular document

The unique words in each document represent only a small subset of all the words in the bag-of-words vocabulary, the feature vectors will mostly consist of zeros, which is why we call them **sparse**.

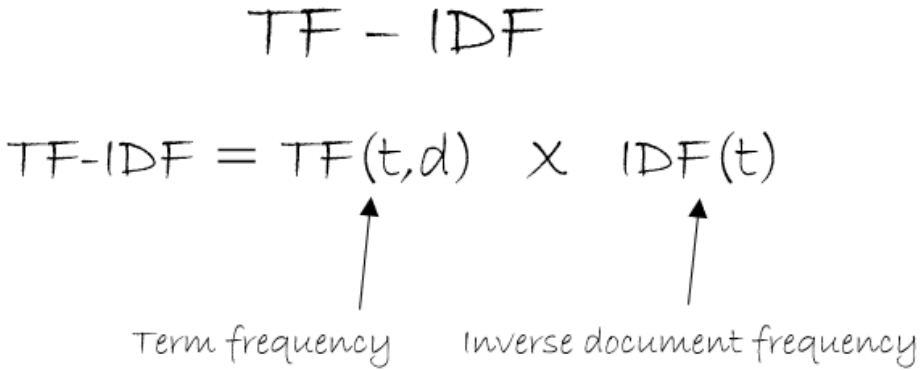
Document	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

Creating Bag of Words Model

Google colab notebook - Bag-of-word

6. TF-IDF

TF-IDF (Term Frequency — Inverse Document Frequency) is a statistical measure for text mining, NLP, Machine Learning. It is a measure of importance of a word / term within a document relative to a collection of documents (a.k.s corpus)
TF-IDF is type of text vectorization process where a words or terms within a document are transformed into importance numbers. TF-IDF scores of a word is calculated by multiplying the words Term Frequency (TF) and Inverse Document Frequency (IDF).



6.1 TF (Term Frequency):

Term Frequency [tf(t,d)] is a relative frequency of a term(t) of interest within a document(d)

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

where $f_{t,d}$ is the raw count of a term in a document

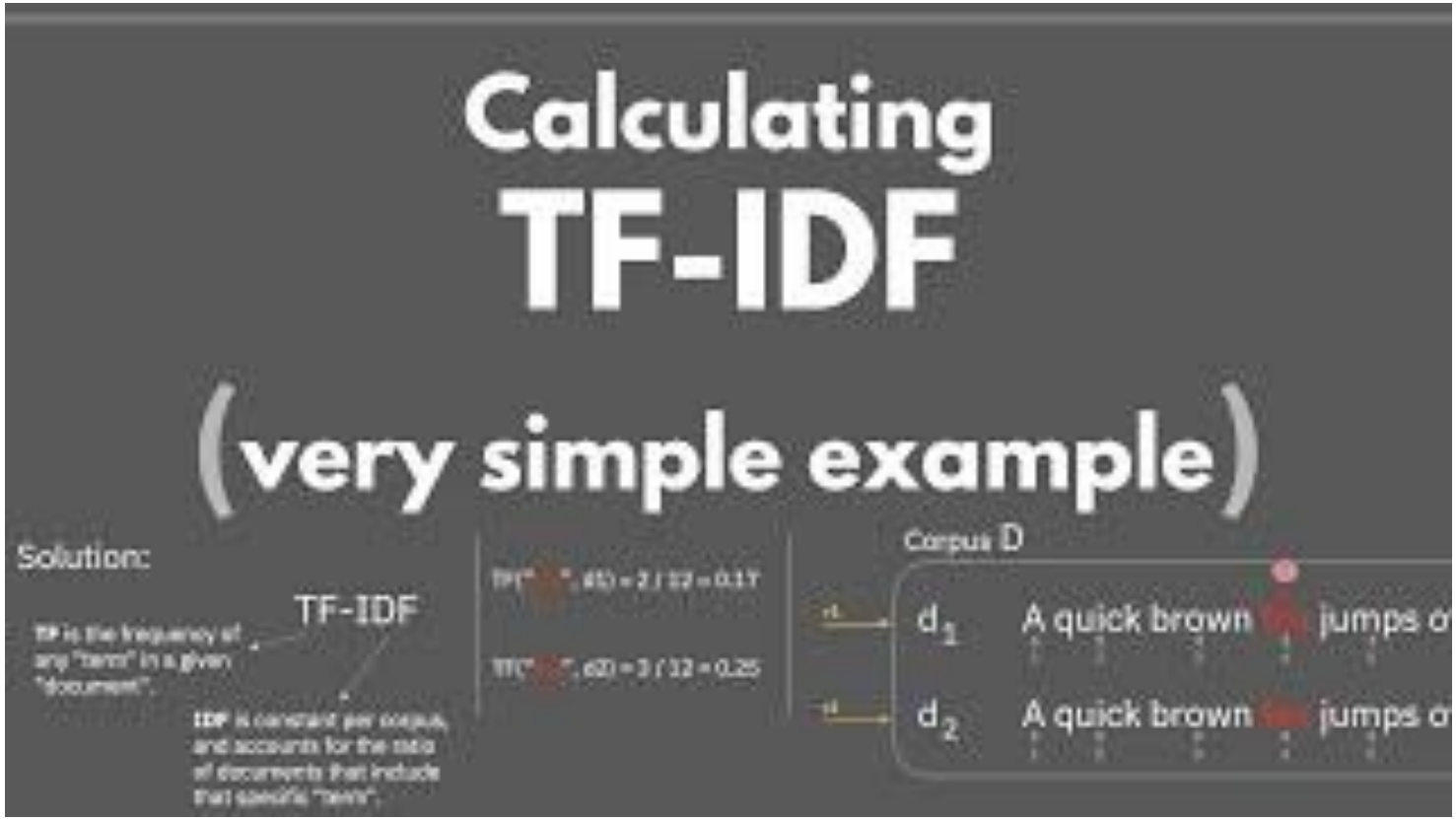
6.2 **IDF (Inverse Document Frequency)**

IDF [idf(t,D)] is a measure of how much information the term(t) provides across all documents(D)

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

where N : total number of documents in the corpus

Calculating TF_IDF



7. Countvectorizer

Machines cannot understand characters and words. So when dealing with text data we need to represent it in numbers to be understood by the machine. Countvectorizer is a method to convert text to numerical data.

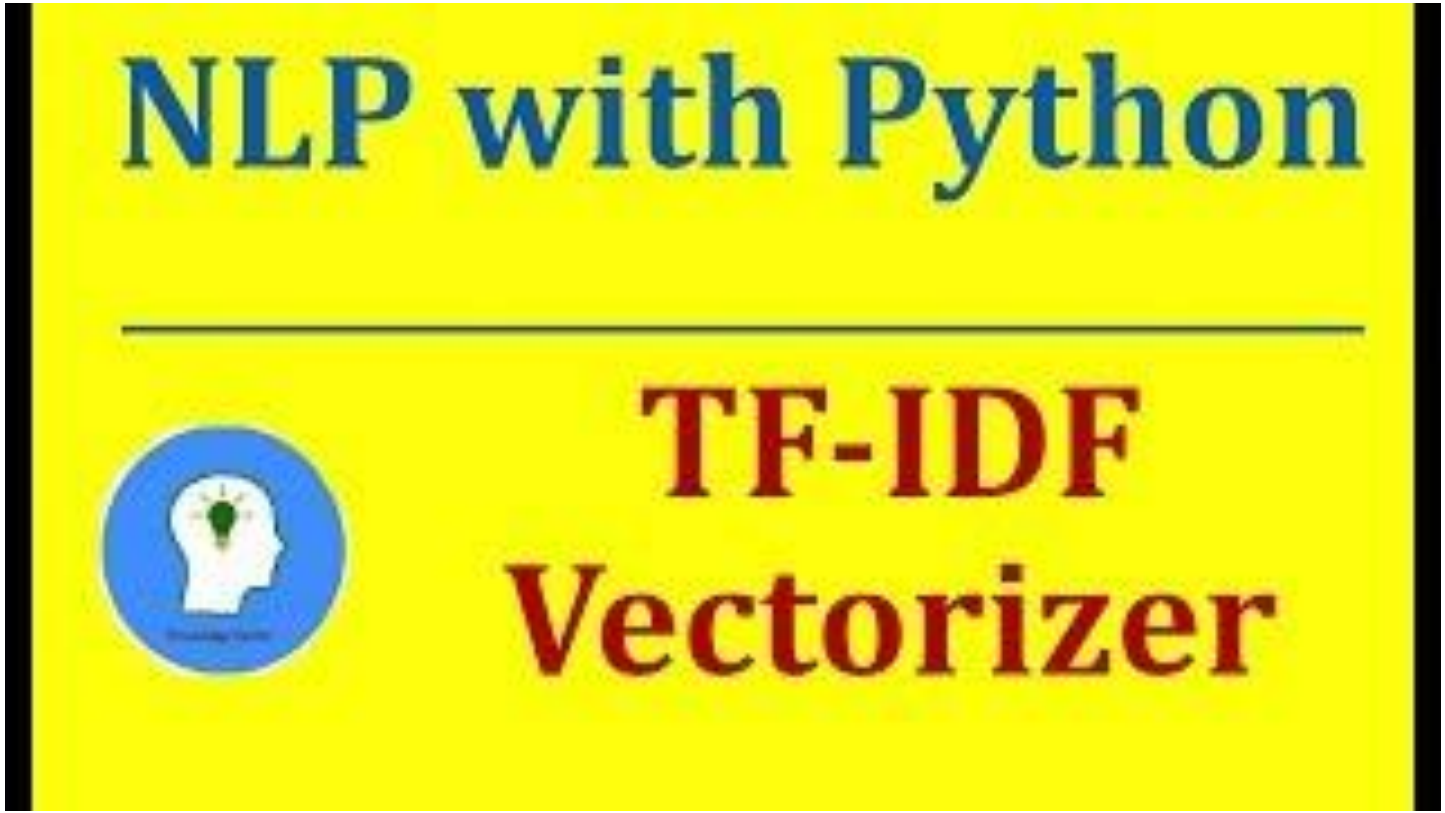
8. TfidfTransformer

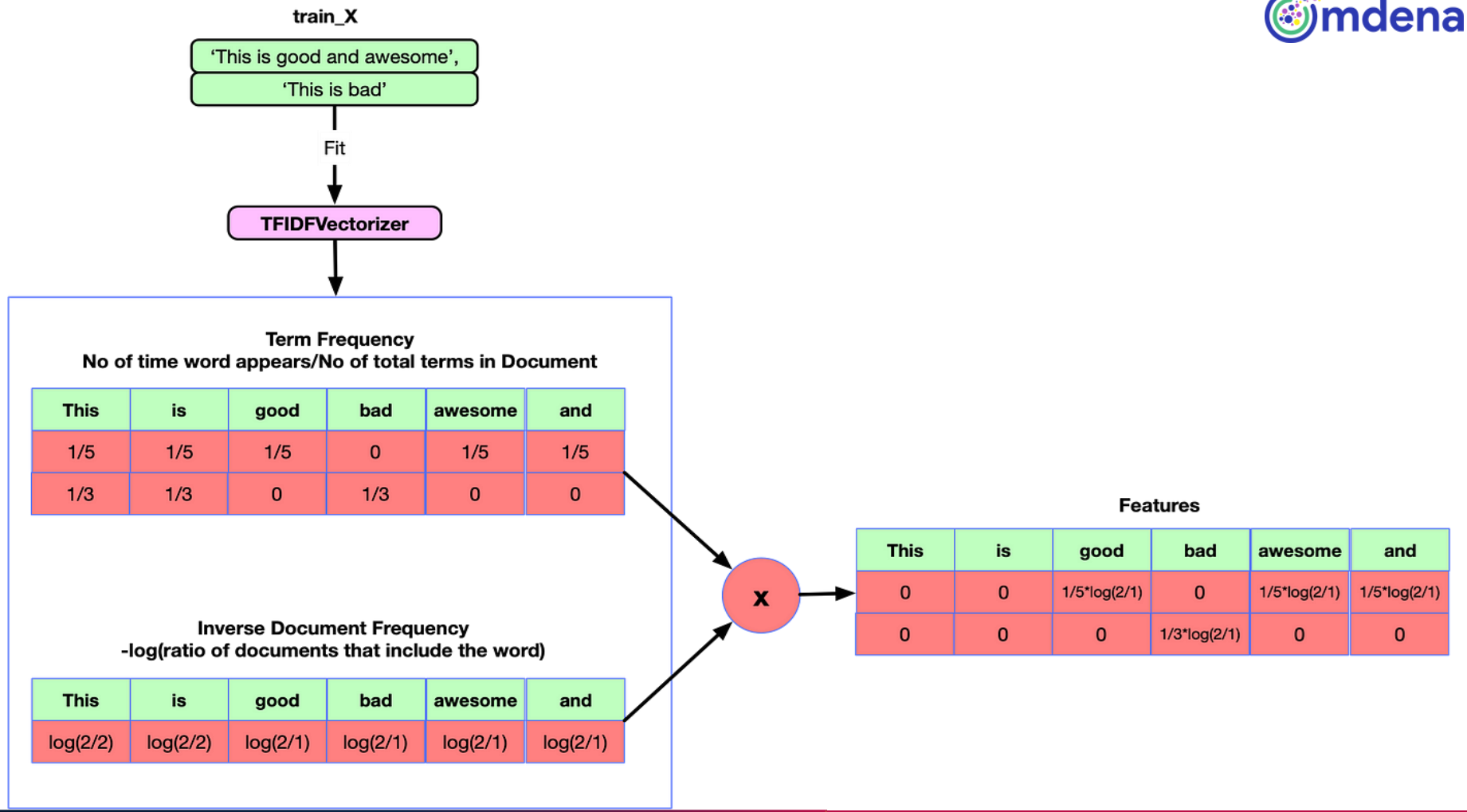
It is a powerful tool for transforming count-based representations of text data (such as Bag of Words) into TF-IDF representations, which provide a more nuanced view of the importance of terms within documents and across the corpus.

9.TfidfVectorizer

The TfidfVectorizer in scikit-learn efficiently converts a collection of raw documents into a sparse matrix of TF-IDF features. It performs tokenization of the input text, calculating both the term frequency (TF) and inverse document frequency (IDF) for each term across the corpus. The resulting TF-IDF matrix represents the importance of each term in each document relative to its occurrence in the entire corpus.

Tf-idf vectorizer




train_X
'This is good and awesome',
'This is bad'
Fit
TFIDFVectorizer
Term Frequency
No of time word appears/No of total terms in Document
This | is | good | bad | awesome | and
1/5 | 1/5 | 1/5 | 0 | 1/5 | 1/5
1/3 | 1/3 | 0 | 1/3 | 0 | 0
Inverse Document Frequency
-log(ratio of documents that include the word)
This | is | good | bad | awesome | and
log(2/2) | log(2/2) | log(2/1) | log(2/1) | log(2/1) | log(2/1)
X
Features
This | is | good | bad | awesome | and
0 | 0 | 1/5*log(2/1) | 0 | 1/5*log(2/1) | 1/5*log(2/1)
0 | 0 | 0 | 1/3*log(2/1) | 0 | 0

II. Creating NLP Project Using With Machine Learning

1. Data Downloading & Exploring

- Downloaded the dataset from Kaggle and used Python's pandas library to read the dataset into a DataFrame.
- Began exploring the dataset to gain insights into its structure and content:
  Viewed initial rows: df.head()
  Checked data types: df.dtypes
  Got descriptive statistics: df.describe()
  Checked for missing values: df.isnull().sum()

2. Data Cleaning & Preparing for analysis

- General Cleaning: cleaned the raw data file which has many unnecessary things such as
  Special Characters: @, #, $, %, ^, &, *, (,), etc.
  Punctuation: ,, ., ;, :, !, ?
  Text formatting: bold, italics, underline
  URLs and email addresses: http:\\

- Text Cleaning:

Stop words: remove common stop words like "the", "a", "an", "is", "are", "to", "of", etc.
Lemmatization or stemming: reduce words to their base form (e.g., "working" to "work") to improve text matching and prevent redundancy.

- Tokenization:

nltk.download('punkt') : contain a pre-trained tokenizer for English text. A tokenizer is a tool divides text into smaller units, such as words, sentences, or phrases.

3. Encoding and Splitting

- Encoding :

Import the LabelEncoder class from the scikit-learn library, which is used for encoding categorical variables into numerical values.
Eg. "Arts" : 1 , "Automation Testing" : 2 , "Blockchain" : 3 etc

- Tfidfvectorizer:

Import the TfidfVectorizer class from scikit-learn, a powerful tool for converting text documents into numerical representations suitable for machine learning tasks.

4. Splitting

Import the train_test_split function from the sklearn.model_selection module in the scikit-learn library.This function is essential for splitting a dataset into two subsets: a training set and a test set.

5. Modeling

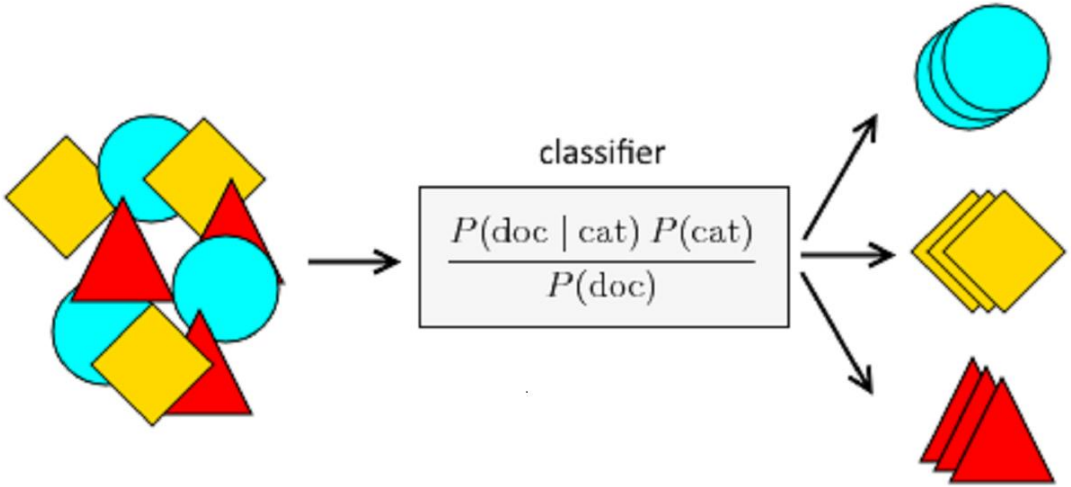
KNeighbors Classifier

It is a type of instance-based or lazy learning algorithm that stores all available data points and classifies new data points based on their similarity to existing data points.

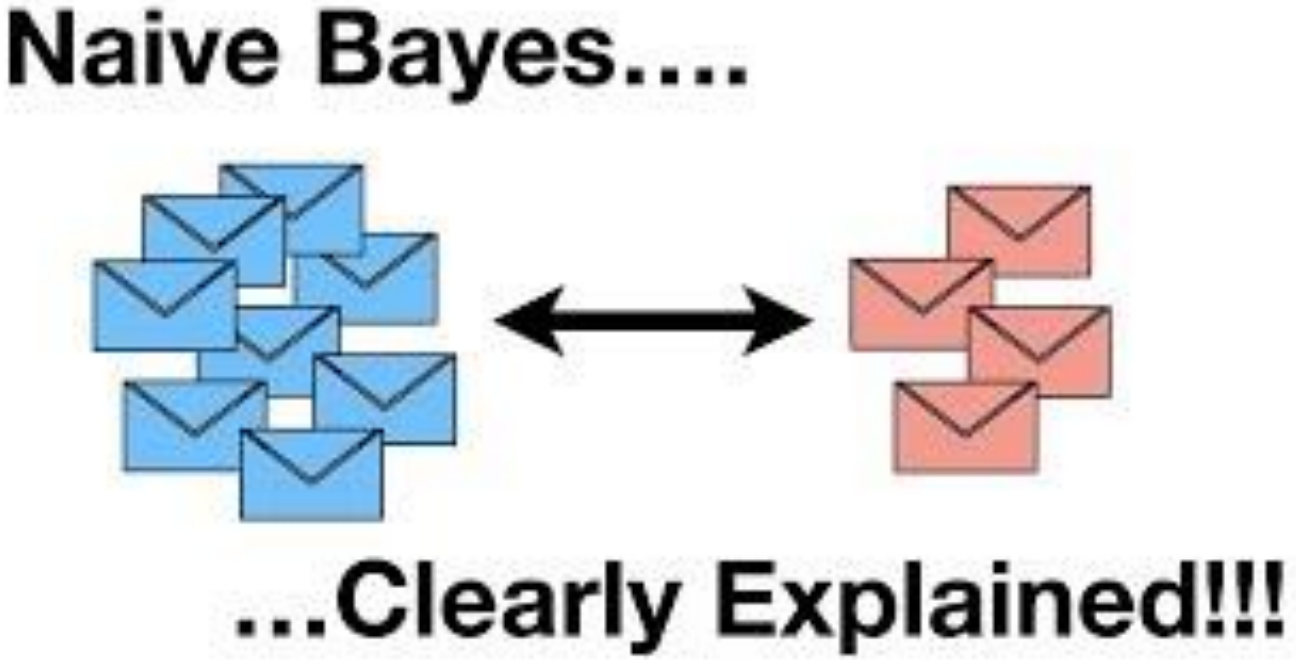
- It belongs to the family of "lazy" algorithms because it doesn't build a model during training; instead, it memorizes the training instances and makes predictions based on their similarity to the input data.
- When a new data point is to be classified, the algorithm finds the K nearest neighbors (data points) in the training set based on some distance metric (e.g., Euclidean distance, cosine similarity) and assigns the majority class label among those neighbors to the new data point.

Naive Bayes Model

- Naive Bayes is based on Bayes' theorem, which describes the probability of an event, given prior knowledge of conditions that might be related to the event.
- Despite its simplicity, Naive Bayes classifiers are often surprisingly effective, especially in text classification tasks.
- The "Naive" assumption in Naive Bayes is that features (words in the case of NLP) are conditionally independent given the class label. This assumption simplifies the computation and allows for efficient training and inference.



Naive Bayes



Google notebook - [Resume screening]

THANK YOU