

**Please note:** This is just only a reference material. Please read the prescribed books in the syllabus and any other resources for knowledge enhancement in this subject ( NLP )

## Machine Translation

**Machine Translation** is the process of using Artificial Intelligence to automatically translate content from one language to another without human input.

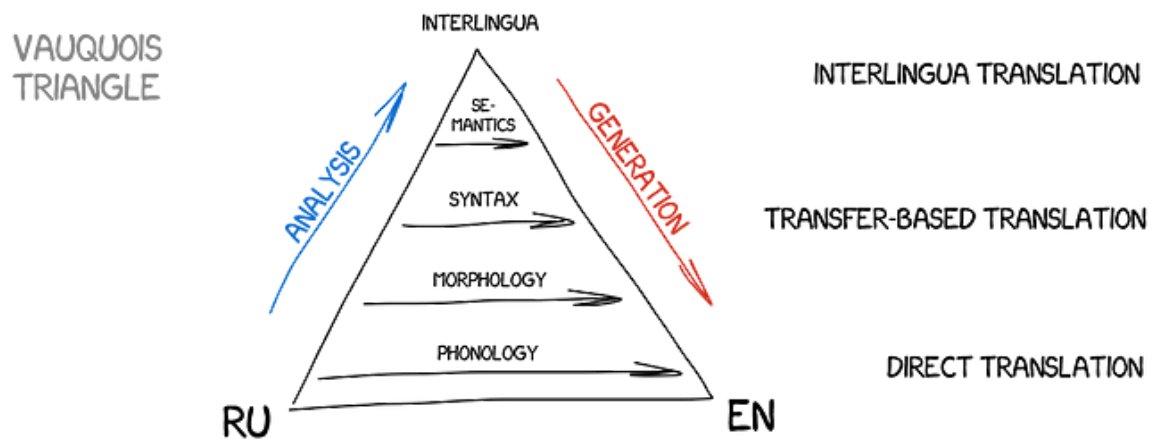
The **main approaches** used for machine translation are

- **Rule-Based** Machine Translation
- **Data-Driven** Approach
- **Neural Based** Machine Translation

### Rule-Based Machine Translation

This model is based on linguistic rules, it requires a large number of lexicons with morphological, syntactic, and semantic information, and

large sets of rules. The code first passes through the text and then by using the syntactic rules it extracts the data and that is used to generate text in the target language.



**Direct Machine Translation:** In this approach, the text is divided into words, then all the words are individually translated into the target language, then the sentence structure is corrected. This approach does not many good results. This approach is not used today.

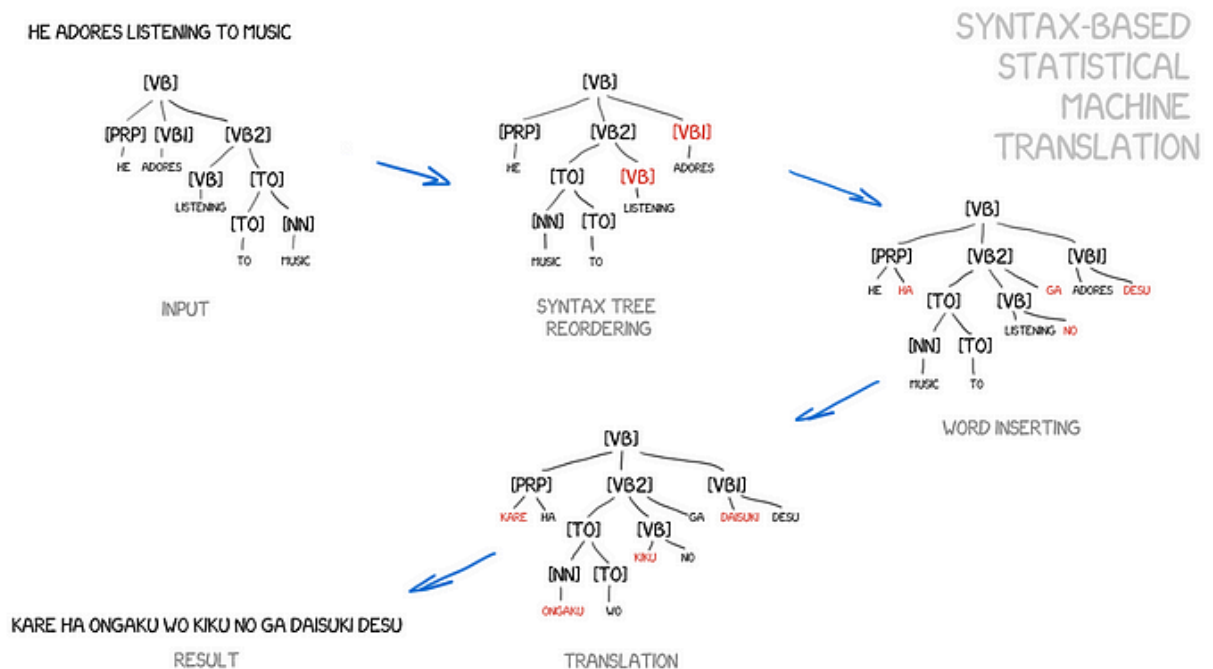
**Transfer Based Machine Translation:** On passing the text this method translated the words syntactically and semantically and lexically. This method will first parse the sentence of the source language then will apply rules that map the grammatical segments of the source sentence to a representation in the target language.

**Interlingual Translation:** In this approach, the text is first translated into an intermediate representation (which is language independent), next to that intermediate representation is converted to the target language. This method requires an analyzer for each word in the source language and generator for each word in the target language.

## **Data-Driven Approach**

**Example-Based Approach:** In this approach, Example of the source language and their translation is fed to Machine. By this, the Machine translates new examples and gives the same answer for

previous fed sentences. The Machine can be designed in such a way that for every new translation, it stores the sentence(or paragraph) of the source language as well as its translation in the target language. It makes faster translation time for previous sentences.



**Statistical Approach:** This approach utilizes translation models whose parameters are learned from the analysis of monolingual and bilingual corpora. There categories of it depending on the part of the sentence that it uses.

- **Word-based SMT:** In this approach, the sentence is broken down into words and does analysis on the word's stat. It assumes that each word covers the same concept in the target and the source language.
- **Phrase-Based SMT:** It is basically based on the same principles (statistics, reordering) as word-based SMT. In these models the translation is done on the phrases.
- **Syntax Based SMT:** In this model the machine learns to first convert syntactic units into a different language and then translate them by words or phrases.

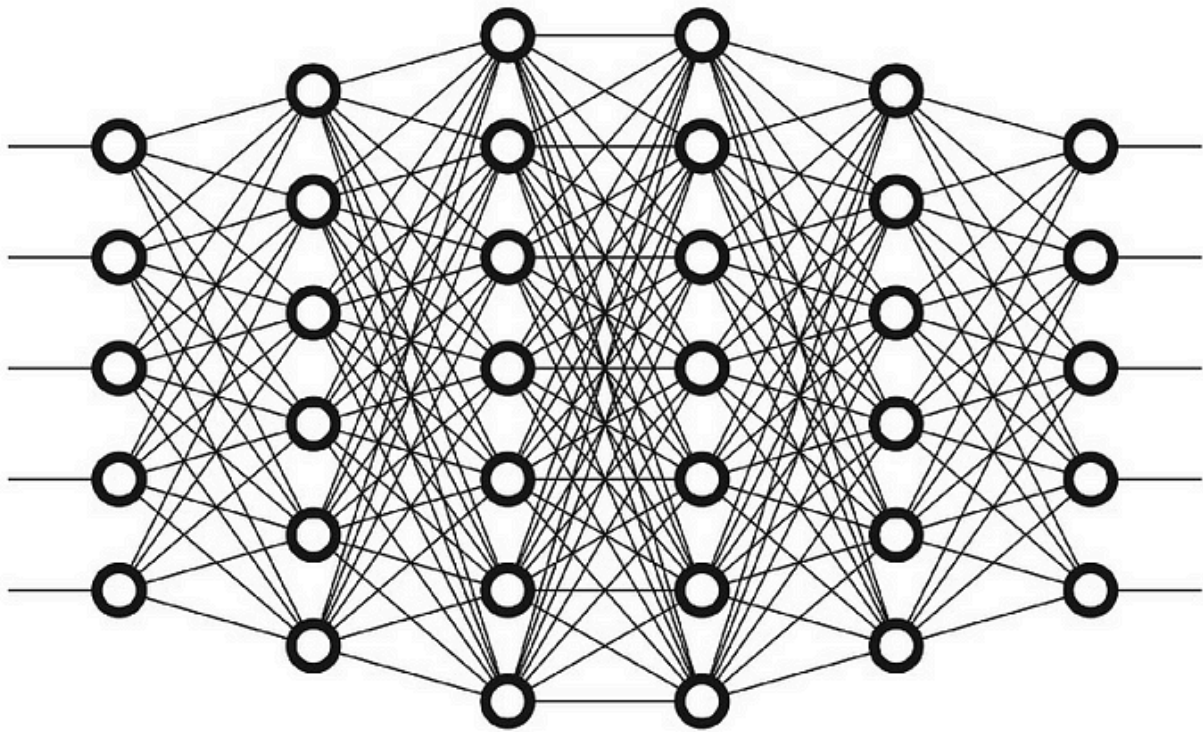
## **Neural Based Machine Translation:**

We all know the beauty of Deep learning. The critical distinction between the deep learning and classical neural networks is in the

ability to search for those specific features, without any idea of their nature. Deep learning works well in machine translation, as well.

This approach uses a neural network for its translation. Google was the first company to use this approach.

In Neural Network, source language's sentence(a set of features of that sentence) is fed to the input layer; it encodes this input. Then it processes the information in the hidden layer and encodes it internally. On the outer layer, it decodes the output in the target language.

































# Word Sense Disambiguation

Word sense disambiguation (WSD) is the task of determining the correct meaning of a word in a given context. It is a common problem in natural language processing (NLP) because many words have

multiple meanings, and the meaning of a word can change depending on the context in which it is used.

For example, the word “bass” can refer to a type of fish or a musical instrument. Disambiguating the word “bass” in the sentence “I caught a bass while fishing” would involve determining that it refers to a type of fish, while in the sentence “I play the bass in a band,” it would refer to the musical instrument.

SAME WORD - DIFFERENT MEANINGS			SAME WORD - DIFFERENT MEANINGS			SAME WORD - DIFFERENT MEANINGS		
	<b>sink</b>			<b>plant</b>			<b>seal</b>	
	<b>nail</b>			<b>bark</b>			<b>nut</b>	
	<b>note</b>			<b>letter</b>			<b>crane</b>	
	<b>fan</b>			<b>bat</b>			<b>bow</b>	
	<b>button</b>			<b>table</b>			<b>trunk</b>	
<small>www.englishforkidz.com</small>			<small>www.englishforkidz.com</small>			<small>www.englishforkidz.com</small>		



For example, consider the two examples of the distinct sense that exist for the word “bass” –

- I can hear bass sound.
- He likes to eat grilled bass.

The occurrence of the word **bass** clearly denotes the distinct meaning. In first sentence, it means **frequency** and in second, it means **fish**. Hence, if it would be disambiguated by WSD then the correct meaning to the above sentences can be assigned as follows –

- I can hear bass/frequency sound.
- He likes to eat grilled bass/fish.

### Example 1

```
import nltk

nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('wordnet')

from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.tokenize import word_tokenize

# Sample text and ambiguous word
text = "He leaves the room when she enters."
```

```
ambiguous_word = "leaves"

# Tokenize the text

tokenized_text = word_tokenize(text)

# Perform WSD using Lesk algorithm

sense = lesk(tokenized_text, ambiguous_word)

print(f"The sense of the word '{ambiguous_word}' in the context of  
the text is: {sense.definition()}")
```

The sense of the word 'leaves' in the context of the text is: the period of time during which you are absent from work or duty

## Example 2

```
import nltk

nltk.download('punkt')

nltk.download('punkt_tab')

nltk.download('wordnet')

from nltk.corpus import wordnet as wn

from nltk.wsd import lesk

from nltk.tokenize import word_tokenize
```

```
# Sample text and ambiguous word

text = "The player hit the ball with his bat."

ambiguous_word = "bat"


# Tokenize the text

tokenized_text = word_tokenize(text)


# Perform WSD using Lesk algorithm

sense = lesk(tokenized_text, ambiguous_word)


print(f"The sense of the word '{ambiguous_word}' in the context of
the text is: {sense.definition()}")
```

The sense of the word 'bat' in the context of the text is: strike with, or as if with a  
baseball bat

### Example 3

```
import nltk

nltk.download('punkt')

nltk.download('punkt_tab')

nltk.download('wordnet')

from nltk.corpus import wordnet as wn

from nltk.wsd import lesk

from nltk.tokenize import word_tokenize
```

```
# Sample text and ambiguous word

text = "The band played rock music at the concert."
ambiguous_word = "rock"


# Tokenize the text

tokenized_text = word_tokenize(text)


# Perform WSD using Lesk algorithm

sense = lesk(tokenized_text, ambiguous_word)


print(f"The sense of the word '{ambiguous_word}' in the context of
the text is: {sense.definition()}")
```

The sense of the word 'rock' in the context of the text is: a genre of popular music originating in the 1950s; a blend of black rhythm-and-blues with white country-and-western

## **Information Retrieval & its Design Features**

**Information retrieval (IR) is a process that facilitates the effective and efficient retrieval of relevant information from large collections of unstructured or semi-structured data. IR systems assist in searching for, locating, and presenting information that matches a user's search query or information need.**

Information retrieval (IR) is the branch of computing that deals with models for finding material of an unstructured nature that satisfies an information need from within large collections of documents usually stored on the web or computers.

The term unstructured data refers to data that does not have a clear format, semantically tough to understand and is not easy for a computer program to understand with simple, obvious rules like text documents, images, video files or graphs also sometimes.

- An information retrieval system is a software system that provides access to books, journals and other documents, and stores and manages those documents. Web search engines are the most visible information retrieval applications.

The general objective of an Information Retrieval System is to minimize the difficulty of a user locating the information they need from the system.

- This difficulty or overhead can be expressed as the time a user spends in all of the steps leading to reading an item containing the needed information. Examples of the steps include such as query generation, query execution, scanning results of the query to select items to read, reading non-relevant items and further downstream tasks, if any.
- The main problem statement in information retrieval is that decisions must be made for every document or information object regarding whether or not to show it to the person who is retrieving the information.
- Information retrieval needs to find relevance of the documents on based on inputs such as keyword or example documents.

We use the word document as a general term that could also include non-textual information, such as multimedia objects.

### **Defining Relevance of Items in Information Retrieval System :**

The term *relevant* item is used in information retrieval to represent all the items containing the needed information for the user making the query. From the perspective of the user, both relevant and needed can be considered synonymous.

- Relevant document contains the information that a person was looking for when they submitted the query to the information retrieval system.
- Topical relevance: This dimension of measuring relevance is about on the topic or on the subject, this will tell us whether the topic of the information retrieved matches the topic of the request.
- User relevance: This measure places the focus of relevance with respect to individual's perceptions of information and information environment not in information as represented in a document or some other concrete form.

### **Types of Information Retrieval Models**

The classification of information retrieval models is done based on type of interaction between documents and queries:

- The way IR models represent the documents and query statements.
- How the information retrieval system matches the query with the documents in the corpus to find out the related documents
- How the ranking for the documents is implemented in the system.

The IR models are mainly categorized as Classical Information Retrieval models, Non-Classical Information Retrieval models and Alternative models. Let us learn about them further.

**Classical IR model:** Classical models in information retrieval are the simplest and easy to implement IR models. The models are mainly based on mathematical knowledge which is easily recognized and understood.

- The meaning of the term classic in the name of classical information retrieval systems denotes that they use foundational techniques for documents without any extra information about the structure or content of a document
- Boolean, Vector, and Probabilistic are the three main classical information retrieval models.

**Non-classical IR model:** Non-classical information retrieval models are based on principles of information logic model, situation theory model, and interaction model.

- The non-classical model does not base include any concepts from classical models like similarity, probability, Boolean operations, etc.

**Alternative IR model:** Alternative IR models are the advanced classical information retrieval models making use of specific techniques from other fields like the Cluster model, fuzzy model, and latent semantic indexing (LSI) models.

## Design features of Information Retrieval (IR) Systems

Let us look at some design features of information retrieval systems.

### Inverted Index

An inverted index is an index data structure storing a mapping from content (content can be words or numbers) to its locations in a document or a set of documents.

- We can also say the inverted index is a hashmap-like data structure that directs the user from a word to a document or a web page. The inverted index is also the primary data structure of most information retrieval systems.
- Inverted index as a data structure lists for every word, all documents that contain it, and frequency of the occurrences in the document hence making it easy to search for hits of a query word.
- Types of inverted indexes: Mainly two types record level inverted index, and word level inverted index.
  - Record level inverted index contains a list of references to documents for each word.
  - Word level inverted index additionally contains the positions of each word within a document. This form of the inverted index also offers more functionality but needs more processing power and space to be created.
- Advantages of inverted index: The main utility of inverted index is that it allows fast full-text searches at the cost of increased processing when a document is added to the database.
  - Inverted index is easy to develop.
  - Inverted index is also the most popular data structure used in document retrieval systems used on a large scale for example, in search engines.
- Disadvantage of inverted index: Inverted index has a large storage overhead with high maintenance costs for the update, delete, and insert statements.



## Stop Word Elimination

Stop words are high-frequency words that are deemed unlikely to be useful for searching inside the documents of the information retrieval system.

- All the words in the corpus with less semantic weights are kept in a list called a stop list.
- Example: Articles like a, an, the, and prepositions like in, of, for, at, etc. are examples of stop words.
- Size reduction of the inverted index using stop list: One main pro of eliminating stop words is that the size of the inverted index can be significantly reduced by a stop list.
  - As per Zipf's law, a stop list covering a few dozen words reduces the size of the inverted index by almost half.
- One disadvantage of stop word elimination is that sometimes it may cause the elimination of the term that is useful for searching.
  - Example: If we eliminate the alphabet A from Vitamin A, then the word will lose its significance.

## Stemming

Stemming is the heuristic process of extracting the base form of words by chopping off the ends of words. It is the process of producing morphological variants of a root or base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. Stemming is one of the important steps in information retrieval systems like search engines.

- For example, the words laughing, laughs, and laughed would be stemmed from the root word laugh.
- Usage of stemming in information retrieval system with an example: If we want to search for the word *chocolate* in a collection of documents, we want to see all the documents that have information about the word chocolate.

- It may so happen that the words *chocolates*, *chocolatey*, and *choco* may be present in many documents instead of *chocolate*.
- To relate these many words, we can stem these words into their root word *chocolate* again so that we can retrieve all the documents containing this base word no matter the way it is represented across documents.
- There are many standard tools for performing this reduction (of stemming into root word) like the Porter's Stemmer, the Snowball stemmer, the Lancaster stemmer, etc.

## Conclusion

- Information retrieval is the task of ranking a list of documents or search results in response to a query.
- The relevance of the results can be measured in terms of precision and recall either from the perspective of documents or users.
- Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances & Recall (sensitivity) is the fraction of relevant instances that were retrieved.
- Classical information retrieval models are based on mathematical knowledge that is easily recognized and understood.
- Non-Classical Information Retrieval models are opposite to the classical IR models and are based on principles like the information logic model, situation theory model, and interaction models.
- Information retrieval systems need to be based on effective design principles like inverted index (primary data structure), stop word removal and stemming for enhanced performance.

## **Sentiment Analysis and its Various Types (Levels):**

Sentiment analysis is the process of analyzing people's opinions, thoughts, and impressions about a topic, product, or service. It can be performed at different levels, depending on the objective of the analysis and the problem being considered:

- **Document-level:** Determines the overall opinion of a document by classifying it as positive, negative, or neutral
- **Sentence-level:** Analyzes each sentence for polarity, or whether it expresses an opinion
- **Aspect-level:** Assigns polarity to each aspect of a sentence, and is the most fine-grained level of analysis
- **Phrase level:** Analyzes opinion words at the phrase level
- **Sub-sentence level:** Obtains the sentiment of sub-expressions within a sentence

The level of analysis to choose depends on the objective of the analysis. For example, document-level analysis is sufficient for some use cases, but not for others.

Sentiment analysis is used in customer care, market research, and reputation management. It can help businesses understand their customers' sentiments towards their brand.

Sentiment analysis at sentence level		
Text	Sentiment	
<i>The touch screen is cool</i>	Positive	
Sentiment analysis at document level		
Text	Sentiment	
<i>I bought an iPhone a few days ago. It is such a nice phone, although a little large. The touch screen is cool. The voice quality is clear too. I simply love it!"</i>	Positive	
Sentiment analysis at Aspect level		
Text	Aspect	Sentiment
<i>The iPhone's <u>call quality</u> is good, but its <u>battery life</u> is short.</i>	<i>call quality</i>	Positive
	<i>battery life</i>	Negative

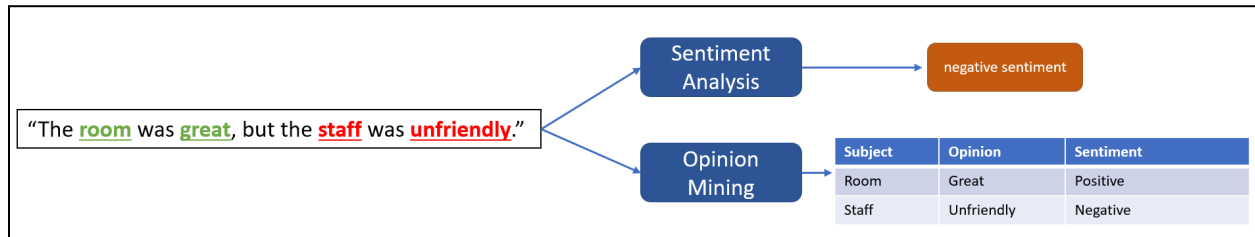
## Difference between Sentiment Analysis and Opinion Mining

Sentiment analysis and opinion mining are two ways of detecting positive and negative sentiment. Using sentiment analysis, you can get sentiment labels (such as "negative", "neutral" and "positive") and confidence scores at the sentence and document-level. Opinion Mining provides granular information about the opinions related to words (such as the attributes of products or services) in the text.

Opinion Mining is a feature of Sentiment Analysis. Also known as Aspect-based Sentiment Analysis in Natural Language Processing (NLP), this feature provides

more granular information about the opinions related to attributes of products or services in text.

For example, if a customer leaves feedback about a hotel such as "The room was great, but the staff was unfriendly.", Opinion Mining will locate targets (aspects) in the text, and their associated assessments (opinions) and sentiments. Sentiment Analysis might only report a negative sentiment.



Opinion Mining will locate targets (nouns or verbs) in the text, and their associated assessment (adjective). For example, the sentence "*The restaurant had great food and our server was friendly*" has two targets: *food* and *server*. Each target has an assessment. For example, the assessment for *food* would be *great*, and the assessment for *server* would be *friendly*.

	Opinion Mining	Sentiment Analysis
Definition	The process of extracting opinions, targets, and aspects from text.	The process of determining the sentiment polarity( positive, negative, neutral )
Focus	Extracts what is being talked about and the associated opinion	Determines how the writer feels about the topic.
Granularity	Focuses on aspects, entities, or topics.	Focuses on overall or fine grained sentiment classification.
Output	Opinion - related entities, like aspect-sentiment pairs.	Sentiment polarity or emotional tone.

<b>Example</b>	“The camera is great, and the battery is awesome.” .. Extracts opinion about camera as great and about battery as awesome.	“The camera is great, and the battery is awesome.” .. Overall sentiment is positive.
<b>Applications</b>	Product Reviews, surveys, and customer feedback analysis.	Brand monitoring, social media analysis and sentiment tracking.

## Named Entity Recognition and its Applications

*Named Entity Recognition (NER) is a natural language processing (NLP) task that involves identifying and categorizing named entities within a body of text into predefined categories such as persons' names, organizations, locations, dates, numerical expressions, and other types of entities.*

The goal of NER is to extract and label relevant entities in a text corpus automatically.

For example, given the sentence “**Apple Inc. was founded by Steve Jobs and Steve Wozniak in California on April 1, 1976,**”

a NER system would identify **“Apple Inc.” as an organization, “Steve Jobs” and “Steve Wozniak” as persons, “California” as a location, and “April 1, 1976” as a date.**

NER systems typically utilize machine learning algorithms, particularly **supervised learning techniques**, to recognize and classify named entities. These algorithms are trained on labeled datasets where each word or phrase in the text is tagged with its corresponding entity type.

Named Entity Recognition (NER) represents different **types** of entities that a model is trained to recognize and classify within a text.

**Here’s an explanation of some common predefined categories:**

### **1. PERSON:**

Represents names of people, including first names, last names, and full names.

## **2. ORG:**

Stands for organizations. This category includes names of companies, institutions, government agencies, and other organized groups.

## **3. GPE (Geopolitical Entity):**

Represents geopolitical entities such as countries, cities, states, provinces, and territories.

## **4. DATE:**

Represents specific points in time or date ranges. This category includes dates, months, years, weekdays, and other temporal expressions.

## **5. TIME:**

Represents specific times of day or time intervals.

## **6. MONEY:**

Represents monetary expressions, including currency symbols, amounts, and monetary units.



## **7. PERCENT:**

Represents percentages or proportions expressed as numeric values followed by the percent symbol (%).

## **8. QUANTITY:**

Represents quantities or measurements, such as distances, weights, volumes, or counts.

## **9. ORDINAL:**

Represents ordinal numbers, indicating rank or order in a sequence (e.g., first, second, third).

## **10. CARDINAL:**

Represents cardinal numbers, which denote quantity or count (e.g., one, two, three).

```
import spacy
```

```
from spacy import displacy
```

```
# Load the English language model
```

```
nlp = spacy.load("en_core_web_sm")

# Define your text

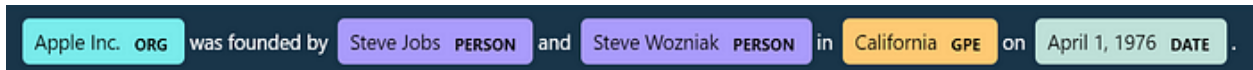
text = "Apple Inc. was founded by Steve Jobs and Steve Wozniak in California on April 1, 1976."

# Process the text with spaCy

doc = nlp(text)

displacy.render(doc, style="ent")
```

Output:



## How Named Entity Recognition (NER) work?

### 1. Tokenization:

The input text is split into individual words or tokens. This step breaks down the text into its basic units for further analysis.

## **2. Part-of-Speech (POS) Tagging:**

Each token is assigned a part-of-speech tag that indicates its grammatical role in the sentence. This step helps identify the words that are likely to represent named entities.

## **3. Named Entity Recognition:**

The NER model analyzes the tokens and their associated POS tags to identify sequences of tokens that form named entities. This involves classifying each identified entity into one of several predefined categories such as persons' names, organizations, locations, dates, and more.

## **4. Entity Classification:**

Once named entities are identified, they are classified into specific categories based on their semantic meaning. For example, a sequence of tokens representing a person's name would be classified as a "PERSON" entity, while a sequence representing a company name would be classified as an "ORG" entity.

## **5. Post-processing:**

After identifying and classifying named entities, post-processing steps may be applied to refine the results. This may include resolving ambiguities, handling nested entities, or normalizing entity representations.

## **6. Output:**

The final output of the NER process is typically a structured representation of the text, where each identified named entity is tagged with its category label.

# **Applications of Name Entity Recognition**

## **1. Information Extraction:**

NER is used to extract structured information from unstructured text. For example, extracting names of people, organizations, locations, and dates from news articles or social media posts.

## **2. Question Answering:**

NER assists in identifying relevant entities mentioned in questions and finding corresponding answers in text documents. For instance, in a medical question answering system, NER can identify diseases, treatments, and symptoms mentioned in a question.

## **3. Text Summarization:**

NER aids in identifying key named entities in a document, which can be used to generate summaries by focusing on important entities and their relationships.

## **4. Sentiment Analysis:**

NER can be used to identify named entities associated with sentiment in text, such as product names or brands in product reviews. This information can be used to analyze sentiment towards specific entities.

## **5. Geographical Information Systems (GIS):**

NER helps in extracting geographic entities like place names, addresses, and landmarks from text data for mapping, navigation, and geospatial analysis.

## **Disadvantages of Name Entity Recognition**

### **1. Ambiguity:**

NER systems have trouble with words that have more than one meaning, like “apple.” This word could mean the fruit or the company that makes iPhones. Figuring out which meaning is right can be hard because it depends on the sentence

### **2. Out-of-Vocabulary Entities:**

NER models may fail to recognize named entities that are not present in their training data. This can happen with newly coined terms, slang, abbreviations, or names of entities not commonly encountered in the training corpus.

### **3. Entity Boundary Detection:**

Identifying the boundaries of named entities accurately can be difficult, especially in languages with complex morphology or in texts with irregular punctuation and formatting. This can lead to incorrect entity recognition.

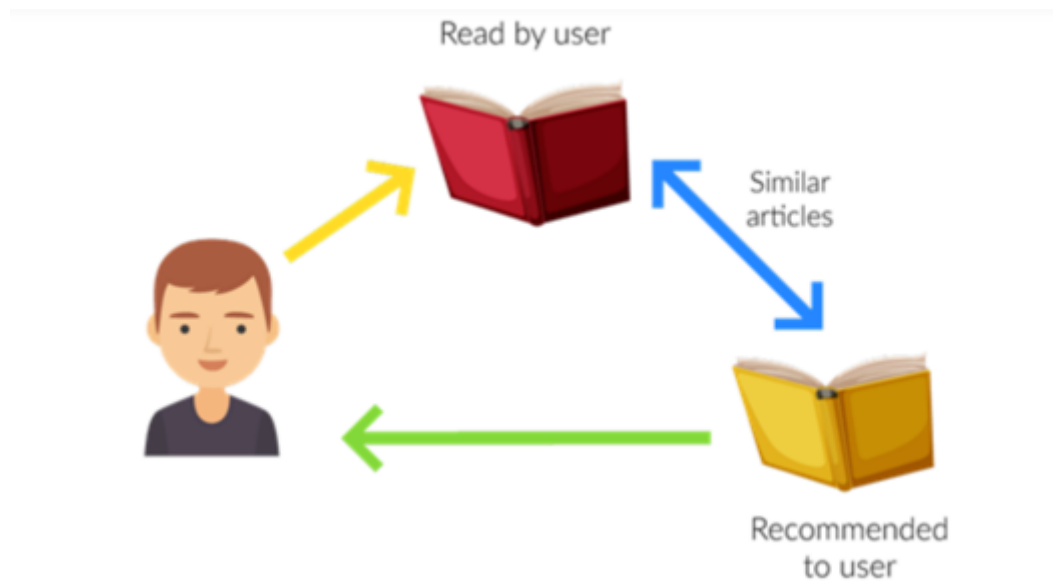
#### **4. Limited Coverage :**

NER systems might not recognize every type of named entity, or they might not do well with certain types of entities. For instance, they might not be great at identifying entities related to specific topics or languages because they didn't see enough examples of those in their training data. This can make them less accurate for those types of entities.

## **Recommendation System and its Types :**

**A recommendation system is a type of machine learning system that provides personalized recommendations to users based on their past behaviors, preferences, and patterns. It is a subclass of information filtering systems**

**that use algorithms to recommend items to users based on their interests or behaviors.**



**Recommendation systems are widely used in e-commerce, social media, entertainment, and other online platforms to increase user engagement and retention, improve customer satisfaction, and drive sales and revenue.**

**How does the Recommendation System work?**



**Here are the four steps of how recommendation systems work:**

- 1. Collecting user data:** The first step in building a recommendation system is to collect user data. This can include user ratings, reviews, clickstream data, purchase history, and other behavioral data. The data can be collected either explicitly, through user surveys or feedback forms, or implicitly, through user interactions with the platform.
- 2. Storing the data:** Once the user data is collected, it needs to be stored in a database or data warehouse for analysis. The data can be stored in a structured or unstructured format, depending on the type and volume of the data.
- 3. Analyzing the data:** The next step is to analyze the user data to identify patterns and trends. This can be done using various data analysis techniques like

**clustering, classification, and regression. The goal is to understand the user's preferences, behaviors, and interests, and to use this information to make personalized recommendations.**

**4. Filtering and recommending: The final step is to filter the data and make recommendations to the user. This can be done using various recommendation algorithms, such as collaborative, content-based, and hybrid filtering. The algorithm uses the user data and the analysis results to generate a list of recommended items the user will likely be interested in. The recommendations are then presented to the user in a personalized way, such as through a recommendation widget, email, or push notification.**

**These four steps are the basic components of most recommendation systems, and the specific implementation**

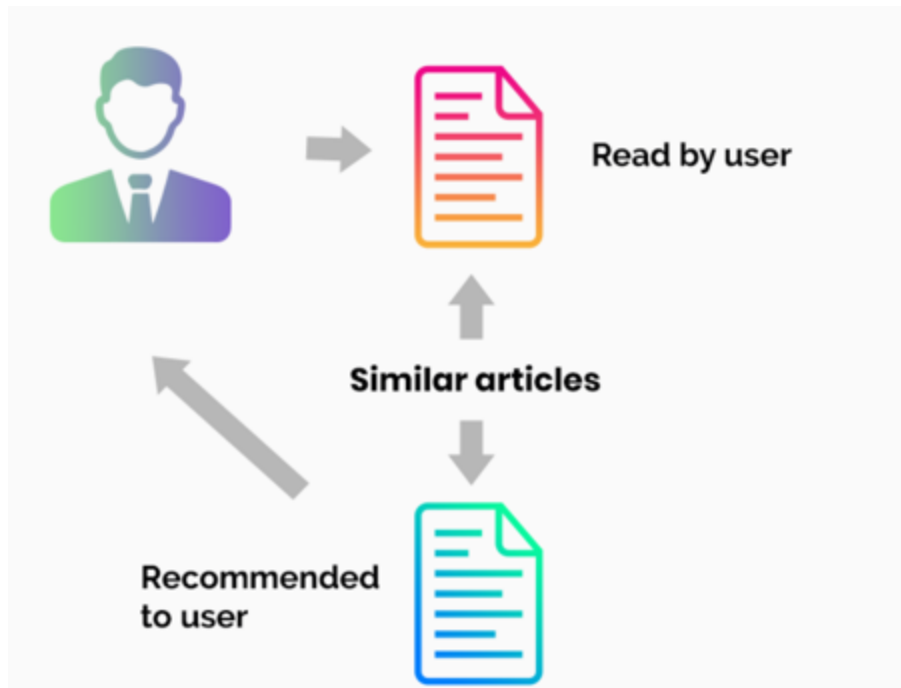
**details may vary depending on the type of system and the application domain.**

## **Types of Recommendation Systems**

**There are three main types of recommendation systems**

→ **Content-Based Filtering**

**Content-based recommendation systems recommend items to users based on their past preferences and behaviors. This type of system analyzes the user's historical data, such as their search history, browsing history, or purchase history, and recommends items that are similar to the ones the user has interacted with before.**

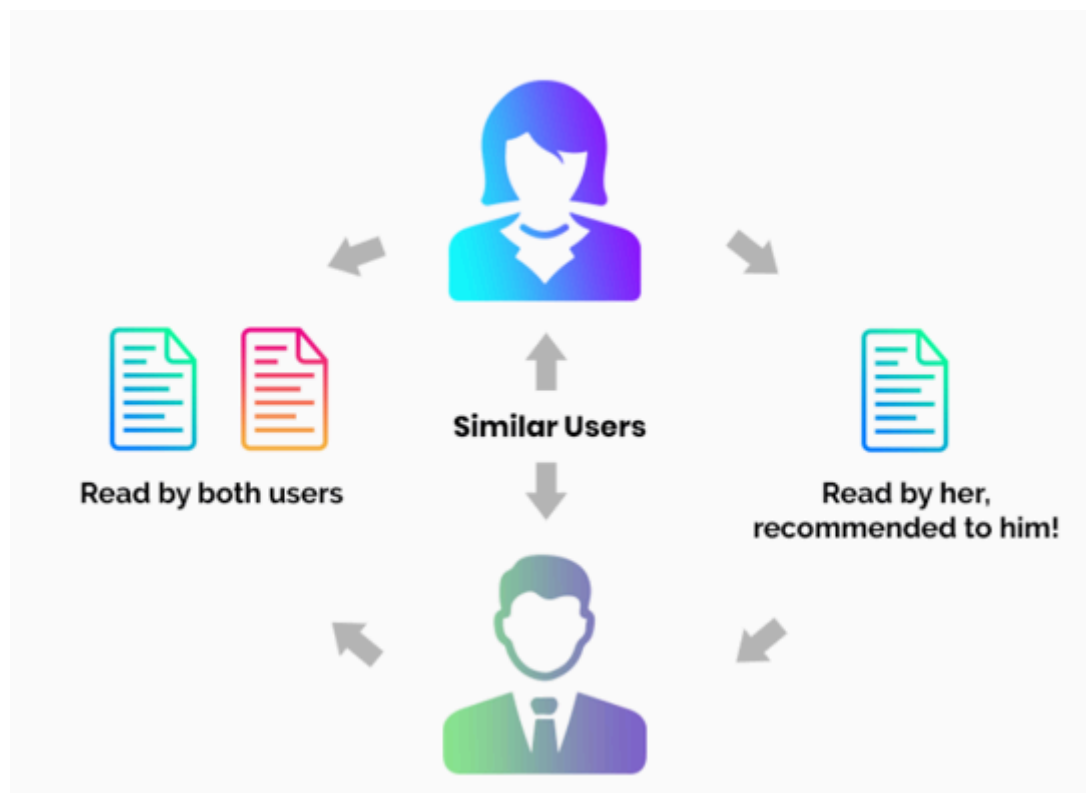


For example, if a user has watched several action movies in the past, a content-based recommendation system might recommend similar action movies to the user. if a user likes to watch movies such as Iron Man, the recommender system recommends movies of the superhero genre or films describing Tony Stark.

→ **Collaborative Filtering**

**Collaborative filtering recommendation systems**  
**recommend items to users based on the preferences and**

behaviors of other similar users. This type of system analyzes the user's historical data, as well as the data of other users with similar preferences, and recommends items that similar users have liked or interacted with before. For example, if two users have similar purchase histories, a collaborative filtering recommendation system might recommend items that one user has purchased to the other user.



**For example, if user A likes Apples, Bananas, and Mango while user B likes Apples, Bananas, and Jackfruit, they have similar interests. So, it is highly likely that A would like Jackfruit and B would enjoy Mango. This is how collaborative filtering takes place.**

**Two kinds of collaborative filtering techniques used are:**

- User-User collaborative filtering**
- Item-Item collaborative filtering**

**User-User collaborative filtering is a type of recommendation system that makes predictions for a user based on the preferences of similar users. It works by finding users with similar tastes and recommending items they liked to the target user. Item-Item collaborative filtering, on the other hand, recommends items to a user based on the preferences for similar items. It works by**

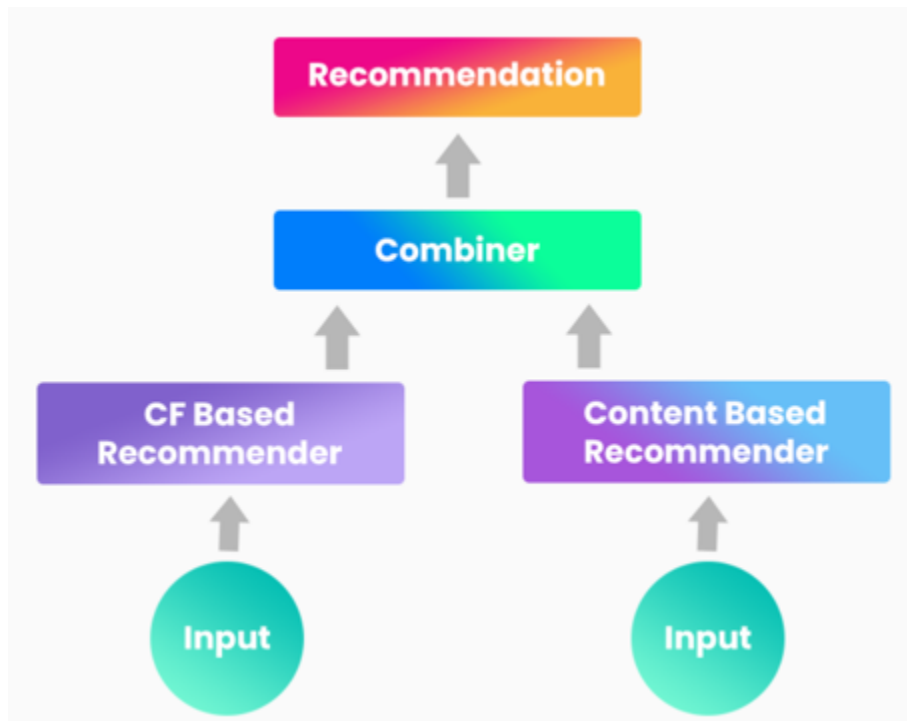
**identifying items that are similar to the ones a user has liked in the past and recommending them to the user.**

→ **Hybrid Recommendation Systems**

**Hybrid recommendation systems combine both content-based and collaborative filtering techniques to provide more accurate and diverse recommendations. This type of system uses a combination of user data, item data, and other contextual information to generate recommendations.**

**hybrid recommendation system might use content-based filtering to recommend items that are similar to the ones the user has interacted with before, and collaborative filtering to recommend items that other similar users have liked or interacted with. By combining the strengths of both approaches, hybrid recommendation systems can provide**

**more accurate and diverse recommendations than either content-based or collaborative filtering alone.**



**Netflix is an excellent case in point for a hybrid recommendation system. It makes recommendations by juxtaposing users' watching and searching habits and finding similar users on that platform. This way, Netflix uses collaborative filtering.**



**By recommending such shows/movies that share similar traits with those rated highly by the user, Netflix uses content-based filtering. They can also veto the common issues in recommendation systems, such as cold start and data insufficiency issues.**

## **Question Answering System and its types :**

**Question Answering models are able to retrieve the answer to a question from a given text. This is useful for searching for an answer in a document. Depending on the model used, the answer can be directly extracted from text or generated from scratch.**

### **Use cases**

**Question Answering (QA) models are often used to automate the response to frequently asked questions by using a**

**knowledge base (e.g. documents) as context. As such, they are useful for smart virtual assistants, employed in customer support or for enterprise FAQ bots (i.e. directed towards enterprise employees).**

**Moreover, many search systems augment their search results with instant answers, which provide the user with immediate access to information relevant to their query.**

## **Question Answering Types :**

**QA systems differ in the way answers are created.**

- Extractive QA: The model extracts the answer from a context and provides it directly to the user. It is usually solved with BERT-like models.**
- Generative QA: The model generates free text directly based on the context. It leverages Text Generation models.**

Moreover, QA systems differ in where answers are taken from.

- **Open QA:** The answer is taken from a context.
- **Closed QA:** No context is provided and the answer is completely generated by a model.

### **Sample code :**

You can infer with QA models with the Hugging Face `transformers` library using the `question-answering` pipeline, which by default will be initialized with the `distilbert-base-cased-distilled-squad` model (which is a model for extractive open QA). This pipeline takes a question and a context from which the answer will be extracted and returned.

First, let's install the `transformers` library using *pip* as usual.

Then, we create a pipeline object with the `question-answering` task, and use it by providing a *question* and a *context*.

The model returns a dictionary containing the keys:

- ***answer***: The text extracted from the context, which should contain the answer.
- ***start***: The index of the character in the context that corresponds to the start of the extracted answer.
- ***end***: The index of the character in the context that corresponds to the end of the extracted answer.
- ***score***: The confidence of the model in extracting the answer from the context.

**Fast Question Answering over many documents:**

Running the QA model over many documents can be slow.

To speed up the search, you can first use passage ranking

**models to see which documents might contain the answer to the question and iterate over them with the QA model.**

## **Question Answering Datasets :**

**The dataset that is used the most as an academic benchmark for extractive question answering is SQuAD (The Stanford Question Answering Dataset). SQuAD is a reading comprehension dataset, consisting of questions posed by crowd-workers on a set of Wikipedia articles, where the answer to every question is a segment of text from the corresponding reading passage. It contains 100,000+ question-answer pairs on 500+ articles.**

## **Developing QA System using BERT**

**BERT, which stands for Bidirectional Encoder**

**Representations from Transformers developed by**

**researchers at Google in 2018, is based on Transformers, a**

**deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection.**

**It is designed to pre-train deep bidirectional representations from an unlabeled text by jointly conditioning on both the left and right contexts. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.**

**Install the transformers library,**

```
!pip install transformers
```

**Load the BertForQuestionAnswering model and the tokenizer.**

```

import torch

from transformers import BertForQuestionAnswering

from transformers import BertTokenizer

#Model

model =
BertForQuestionAnswering.from_pretrained('bert-large-uncased-whole-word-ma
sking-finetuned-squad')

#Tokenizer

tokenizer =
BertTokenizer.from_pretrained('bert-large-uncased-whole-word-masking-finet
uned-squad')

```

Create a QA example and use function `encode_plus()` to encode the example. The function `encode_plus()` returns a dictionary that contains `input_ids`, `token_type_ids`, and

attention mask but we only need input\_ids and token\_type\_ids for the QA task.

```
question = '''What is Machine Learning?'''
```

```
paragraph = ''' Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to progressively improve their performance on a specific task. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in the applications of email filtering, detection of network intruders, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics. '''
```

```
encoding = tokenizer.encode_plus(text=question, text_pair=paragraph)
```

```
inputs = encoding['input_ids'] #Token embeddings
```



```
sentence_embedding = encoding['token_type_ids'] #Segment embeddings

tokens = tokenizer.convert_ids_to_tokens(inputs) #input tokens
```

**Run the QA example through the loaded model.**

```
start_scores, end_scores = model(input_ids=torch.tensor([inputs]),
token_type_ids=torch.tensor([sentence_embedding]), return_dict=False)
```

**We can get both the start index and the end index and use both the indices for span prediction.**

```
start_index = torch.argmax(start_scores)

end_index = torch.argmax(end_scores)
```

```
answer = ' '.join(tokens[start_index:end_index+1])  
  
print(answer)
```

## Output:

```
The answer is : the scientific study of algorithms and  
statistical models
```

# Shingling

Shingling is the process of “Converting documents to sets”.

A ***k*-shingle** for a document is a **sequence of *k* tokens** that appears in the document. **Tokens can be characters, words** or something else, depending on the application. **One important point to note is that a document's *k*-shingle set should be unique.**

**Example:**

**Compute  $k=2$  character shingles for document  $D_1 =$**  `ab cab`

Set of 2 character shingles:  **$S(D_1) =$**  `{ab, bc, ca}`

Note that the substring `ab` appears twice within  $D_1$ , but **appears only once** as a shingle.

## Similarity Metric for Shingles

**A natural similarity measure for Shingles is the Jaccard similarity:**

$$\text{sim}(D_1, D_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

Similarity measure between documents  $D_1$  and  $D_2$  where  $C_1$  and  $C_2$  are the  $k$ -shingles respectively

How to convert Documents to Sets ?

The first option is the bag of words model, where each document is treated as an unordered set of words. A more general approach is to shingle the document. This takes consecutive words and group them as a single object. A  $k$ -shingle is a consecutive set of  $k$  words. So the set of all 1-shingles is exactly the bag of words model. An alternative name to  $k$ -shingle is  $k$ -gram.

$D_1$  : I am Sam.

$D_2$  : Sam I am.

$D_3$  : I do not like green eggs and ham.

$D_4$  : I do not like them, Sam I am.

The ( $k = 1$ ) word shingles of  $D_1 \cup D_2 \cup D_3 \cup D_4$  are: {[I], [am], [Sam], [do], [not], [like], [green], [eggs], [and], [ham], [them]}.

The ( $k = 2$ ) word shingles of  $D1 \cup D2 \cup D3 \cup D4$  are:  $\{[I \text{ am}], [am \text{ Sam}], [Sam \text{ I}], [I \text{ do}], [do \text{ not}], [not \text{ like}], [like \text{ green}], [green \text{ eggs}], [eggs \text{ and}], [and \text{ ham}], [like \text{ them}], [them \text{ Sam}]\}$ .

The set of  $k$ -shingles of a document with  $n$  words is at most  $n - k$ .

### Example

Doc1: The night is dark and the moon is red.

Doc2: The moon in the night is red

Doc3: I can see moon is red, the night is dark.

Compute  $k=3$  word shingles of  $D1 \cup D2 \cup D3$

The ( $k = 3$ ) word shingles of  $D1 \cup D2 \cup D3$  are:  $\{[the \text{ night is}], [night \text{ is dark}], [is \text{ dark and}], [dark \text{ and the}], [and \text{ the moon}], [the \text{ moon is}], [moon \text{ is red}], [the \text{ moon in}], [moon \text{ in the}], [in \text{ the night}], [night \text{ is red}], [i \text{ can see}], [can \text{ see moon}], [see \text{ moon is}], [is \text{ red the}], [red \text{ the night}]\}$ .

# Document Classification

*Document classification refers to the process of categorizing documents into different classes or categories based on their content or attributes. It can be done manually or algorithmically using AI, ML, and NLP techniques.*

The purpose of document classification is to facilitate storage, management, and analysis of documents. It is widely used in various fields, including computer science, information science, and library science.

Automated document classification techniques leverage machine learning algorithms, such as neural networks, Naive Bayes classifiers, and logistic regression, to classify documents based on patterns and features extracted from the text.

## What is Intelligent Document Processing(IDP)?

***Intelligent Document Processing*** is a broader initiative that encompasses document classification and goes beyond it. IDP combines Artificial intelligence (AI), Machine learning (ML), Natural language processing(NLP), and Optical character recognition (OCR) technologies to capture, extract, and process data from various document formats



## **Types of Document Classification Methods:**

Document classification can be categorized in different ways based on the attributes or approaches used. Here are some types of document classification:

1. **Subject-based Classification:** This type of classification categorizes documents based on their subjects or topics. It focuses on the content of the documents and assigns them to relevant categories.
2. **Rule-based Classification:** Rule-based classification involves writing classification rules manually. These rules define the criteria for assigning documents to specific categories.



### 3. **Supervised Machine Learning Classification:** In

supervised classification, a machine learning model is trained using a labeled dataset. The model learns from the labeled examples and then applies that knowledge to classify new, unseen documents.

### 4. **Unsupervised Machine Learning Classification:**

Unsupervised classification groups documents together based on similarities in their content, without the need for pre-labeled training data. Clustering algorithms are commonly used for unsupervised document classification.

### 5. **Visual Classification:** Visual classification analyzes the visual structure of documents without reading their text. It identifies patterns and layouts specific to different document types to classify them accordingly.

### 6. **Text Classification:** Text classification focuses on defining the type, genre, or theme of the text based on its content.

Natural Language Processing (NLP) techniques can be used

to analyze words, phrases, and their context to understand the semantics of the text.

## **Auto-Classification of Documents:**

*Auto-classification of documents refers to the process of automatically assigning categories or labels to documents without human intervention. This can be achieved using machine learning algorithms, natural language processing techniques, or a combination of both.*

*Auto-classification can save time and effort by eliminating the need for manual categorization of documents.*

## **How does Document Classification Work?**

Document classification typically involves the following steps:

1. **Data Preparation:** *The documents to be classified are collected and preprocessed. This may involve tasks such as removing irrelevant information, tokenizing the text, and converting it into a suitable format for analysis.*
2. **Feature Extraction:** *Relevant features are extracted from the documents. These features can include words, phrases, or other linguistic elements that capture the essence of the document's content.*
3. **Training:** *In supervised classification, a machine learning model is trained using a labeled dataset. The model learns to associate the extracted features with the corresponding document categories.*
4. **Classification:** *Once the model is trained, it can be used to classify new, unseen documents. The model applies the learned patterns and rules to assign the most appropriate category or label to each document.*

5. **Evaluation and Refinement:** *The performance of the classification model is evaluated using metrics such as accuracy, precision, recall, and F1 score. The model can be refined by adjusting its parameters or using techniques like cross-validation.*

## **Automated Document Classification Techniques:**

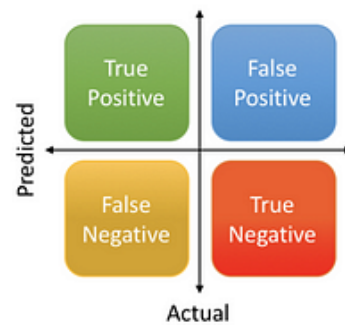
*Automated document classification techniques leverage advanced technologies such as machine learning, deep learning, and natural language processing. These techniques enable the automatic categorization of documents based on their content, eliminating the need for manual intervention.*

- Automated document classification uses AI, ML, and NLP to categorize documents automatically.
- It saves time and improves consistency in document classification.

- Machine learning algorithms can be trained to classify documents based on patterns and features extracted from the text.
- NLP techniques analyze document content, extract relevant features, and identify patterns for classification.
- Computer vision techniques can be used for documents with images or scanned text.
- Data labeling is important for training accurate classification models.
- Automated document classification has applications in spam filtering, content moderation, lead generation, and more.

## **Performance and Correctness Measure**

$$\begin{aligned}
 \text{Precision} &= \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \\
 \text{Recall} &= \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \\
 \text{Accuracy} &= \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}
 \end{aligned}$$



Precision and recall are two extremely important model evaluation metrics. While precision refers to the percentage of your results which are relevant, recall refers to the percentage of total relevant results correctly classified by your algorithm. Unfortunately, it is not possible to maximize both these metrics at the same time, as one comes at the cost of another. For simplicity, there is another metric available, called F-1 score, which is a harmonic mean of precision and recall. For problems where both precision and recall are important, one can select a model which maximizes this F-1 score. For other problems, a trade-off is needed, and a decision has to be made whether to maximize precision, or recall.

$$\text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

**Regular expression**

# What are Regular Expressions

A regular expression (a.k.a. regex or regexp) is a sequence of characters that specifies a search pattern in the text. Usually, such patterns are used by string-searching algorithms for “find” or “find and replace” operations on strings, or for input validation.

When a data scientist comes across a text processing problem, whether it is searching for titles in names or dates of birth in a dataset, regular expressions are the right tool to use. They form part of the basic techniques in NLP and learning them will make you a more efficient programmer as well.

## Regular Expressions use cases

Regular Expressions are used in various tasks such as:

- Data pre-processing;
- Rule-based information Mining systems;
- Pattern Matching;



- Text feature Engineering;
- Web scraping;
- Data validation;
- Data Extraction.

An example use case is extracting all hashtags from a tweet, or getting email addresses or phone numbers from large unstructured text content.

## Regular Expressions with Python

Python provides a convenient built-in module for managing regular expressions: `re`.

### Basic Characters

First of all, it's possible to search for pattern occurrences in a string using the `search` function of the `re` module. This function returns a match object, containing the matched substring (or `None`, if it doesn't exist) and its position inside the original string.

When specifying the regular expression pattern, we usually prepend a `r` character, which means that the string is to be treated as a *raw string* and therefore that all escape codes will be ignored. For example, `"\n"` is a string with the `\n` character, whereas `r"\n"` is a string with the characters `\` and `n`.

Let's start with the basic regular expression characters and some examples:

- `^` : Matches the expression to its right, at the start of a string before it finds a line break;
- `$` : Matches the expression to its left, at the end of a string before it finds a line break;
- `.` : Matches any character except newline;
- `a` : Matches exactly one character `a` ;
- `ab` : Matches the string `ab` .

## Quantifiers

Let's see now how to use quantifiers:

- $a|b$  : Matches expression  $a$  or  $b$ . If  $a$  is matched first,  $b$  is not checked;
  - $+$  : Matches the expression to its left 1 or more times;
  - $*$  : Matches the expression to its left 0 or more times;
  - $?$  : Matches the expression to its left 0 or 1 times.
- 
- $\{p\}$  : Matches the expression to its left exactly  $p$  times;
  - $\{p, q\}$  : Matches the expression to its left  $p$  to  $q$  times;
  - $\{p, \}$  : Matches the expression to its left  $p$  or more times;
  - $\{, q\}$  : Matches the expression to its left up to  $q$  times.

## Character Classes

You can use regular expressions to match character classes such as words, numbers, punctuations, and so on.

- `\w` : Matches alphanumeric characters, that is a-z, A-Z, 0–9, and underscore(\_);
- `\W` : Matches non-alphanumeric characters, that is except a-z, A-Z, 0–9, and \_;
- `\d` : Matches digits, from 0–9;
- `\D` : Matches any non-digits;
- `\s` : Matches whitespace characters, which also include the `\t`, `\n`, `\r`, and space characters;
- `\S` : Matches non-whitespace characters.
- `\n` : Matches a newline character;
- `\t` : Matches a tab character;

## Sets

Regular expression sets can be used to match either a pattern or another pattern.

- `[abc]` : Matches either `a`, `b`, or `c`. It does not match `abc`;
- `[a-z]` : Matches any alphabet from `a` to `z`;
- `[A-Z]` : Matches any alphabets in capital from `A` to `Z`;
- `[a\ -p]` : Matches `a`, `-`, or `p`. It matches `-` because `\` escapes it;
- `[-z]` : Matches `-` or `z`;
- `[a-zA-Z0-9]` : Matches characters from `a` to `z` or from `0` to `9`.

## Examples of Regular Expression

To remove punctuation and special characters from text data, we aim to clean the text and retain only alphanumeric characters and possibly spaces between words. This step is essential in text preprocessing as

punctuation marks and special characters often do not carry significant semantic meaning and can introduce noise in the data.

## Code demonstrating Removing Punctuation and Special Characters.

```
import re

# Sample text with punctuation and special characters
text = "Natural ##language proc##essing## (NLP) is a %%fascinating field. It @@deals with how computers understand and interact with human language***. ***Sentence &*tokenization is **one of the *&*basic tasks in NLP."
```

```
# Remove punctuation and special characters using regular expressions
clean_text = re.sub(r'^a-zA-Z0-9\s', '', text)

# Print the cleaned text
print(clean_text)
```

Output:

```
Natural language processing NLP is a fascinating field It deals with how
computers understand and interact with human language Sentence
tokenization is one of the basic tasks in NLP
```

## Explanation of provided code.

1. We start by importing the `re` module, which allows us to work with regular expressions in Python.

2. We define a sample text containing punctuation marks and special characters.
3. Using the `re.sub()` function, we specify a regular expression pattern `r'[^a-zA-Z0-9\\s]'` to match any character that is not alphanumeric (`[^a-zA-Z0-9]`) or a whitespace character (`\\s`).  
The `^` inside the square brackets indicates negation.
4. We replace all occurrences of non-alphanumeric characters and special characters with an empty string `''`, effectively removing them from the text.
5. The cleaned text is stored in the variable `clean_text`.
6. Finally, we print the cleaned text.

Assuming the most common email formats are represented like

`[string1]@[string2].[2+ characters]` (e.g. `name.surname@gmail.com`,

name@yahoo.co.uk, name@outlook.com ), We can use the following regex for extracting these formats:

```
[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+
```

## Program

```
import re
```

```
text = """I know a set of email addresses that we can extract  
using expression1: abc.df@somecompany.co.uk, abc@gmail.com,  
xyz.ab@tpa.com, dfg.gh@dp.cp.net . But what about  
11.234.abc.ghy@tp.edu, let's check."""
```

```
emails =  
re.findall("([a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+" ,  
text)
```

```
print(emails)
```



## Output

```
['abc.df@somecompany.co.uk', 'abc@gmail.com', 'xyz.ab@tpa.com',  
'dfg.gh@dp.cp.net', '11.234.abc.ghy@tp.edu']
```

```
import re  
s = 'This is my tweet check it out http://tinyurl.com/blah and  
http://blabla.com'  
urls=re.findall(r'(https?://\S+)', s)  
print(urls)
```

```
['http://tinyurl.com/blah', 'http://blabla.com']
```

```
import re  
  
def validate_phone_number(regex, phone_number):  
    match = re.search(regex, phone_number)  
    if match:  
        return True  
    return False  
  
pattern =  
re.compile(r"(\+\d{1,3})?\s?(? \d{1,4})?[\s.-]?\d{3}[\s.-]?\d{4}  
{}")  
  
test_phone_numbers = [  
    "+1 (555) 123-4567",  
    "555-123-4567",  
    "555 123 4567",  
    "+44 (0) 20 1234 5678",
```

```
    "02012345678",  
    "invalid phone number"  
]
```

```
for number in test_phone_numbers:  
    print(f"{number}: {validate_phone_number(pattern, number)}")
```