

Unit-1

Natural language processing: Natural language processing (NLP) is a subfield of computer science and artificial intelligence (AI) that uses machine learning to enable computers to understand and communicate with human language.

NLP enables computers and digital devices to recognize, understand and generate text and speech by combining computational linguistics—the rule-based modelling of human language—together with statistical modelling, machine learning and deep learning.

NLP research has helped enable the era of generative AI, from the communication skills of large language models (LLMs) to the ability of image generation models to understand requests. NLP is already part of everyday life for many, powering search engines, prompting chat-bots for customer service with spoken commands, voice-operated GPS systems and question-answering digital assistants on smartphones such as Amazon's Alexa, Apple's Siri and Microsoft's Cortana.

NLP also plays a growing role in enterprise solutions that help streamline and automate business operations, increase employee productivity and simplify business processes. Natural language processing has heavily benefited from recent advances in machine learning, especially from deep learning techniques. The field is divided into the three parts:

1. Speech recognition—the translation of spoken language into text.
2. Natural language understanding—a computer's ability to understand language.
3. Natural language generation—the generation of natural language by a computer

Why Natural Language Processing Is Difficult?

Human language is special for several reasons. It is specifically constructed to convey the speaker/writer's meaning. It is a complex system, although little children can learn it pretty quickly.

Another remarkable thing about human language is that it is all about symbols. According to Chris Manning, a machine learning professor at Stanford, it is a discrete, symbolic, categorical signaling system. This means we can convey the same meaning in different ways (i.e., speech, gesture, signs, etc.) The encoding by the human brain is a continuous pattern of activation by which the symbols are transmitted via continuous signals of sound and vision.

Understanding human language is considered a difficult task due to its complexity. For example, there are an infinite number of different ways to arrange words in a sentence. Also, words can have several meanings and contextual information is necessary to correctly interpret sentences. Every language is more or less unique and ambiguous. Just take a look at the following newspaper headline “The Pope’s baby steps on gays.” This sentence clearly has two very different interpretations, which is a pretty good example of the challenges in natural language processing.

Benefits of Natural Language Processing:

NLP makes it easier for humans to communicate and collaborate with machines, by allowing them to do so in the natural human language they use every day. This offers benefits across many industries and applications.

1. Automation of repetitive tasks
 2. Improved data analysis and insights
 3. Enhanced search
 4. Content generation
1. Automation of repetitive tasks : NLP is especially useful in fully or partially automating tasks like customer support, data entry and document handling. For example, NLP-powered chatbots can handle routine customer queries, freeing up human agents for more complex issues. In document processing, NLP tools can automatically classify, extract key information and summarize content, reducing the time and errors associated with manual data handling. NLP facilitates language translation, converting text from one language to another while preserving meaning, context and nuances.
 2. Improved data analysis : NLP enhances data analysis by enabling the extraction of insights from unstructured text data, such as customer reviews, social media posts and news articles. By using text mining techniques, NLP can identify patterns, trends and sentiments that are not immediately obvious in large datasets. Sentiment analysis enables the extraction of subjective qualities—attitudes, emotions, sarcasm, confusion or suspicion—from text. This is often used for routing communications to the system or the person most likely to make the next response. This allows businesses to better understand customer preferences, market conditions and public opinion. NLP tools can also perform categorization and summarization of vast amounts of text, making it easier for analysts to identify key information and make data-driven decisions more efficiently.

3. Enhanced search : NLP benefits search by enabling systems to understand the intent behind user queries, providing more accurate and contextually relevant results. Instead of relying solely on keyword matching, NLP-powered search engines analyze the meaning of words and phrases, making it easier to find information even when queries are vague or complex. This improves user experience, whether in web searches, document retrieval or enterprise data systems.
4. Powerful content generation: NLP powers advanced language models to create human-like text for various purposes. Pre-trained models, such as GPT-4, can generate articles, reports, marketing copy, product descriptions and even creative writing based on prompts provided by users. NLP-powered tools can also assist in automating tasks like drafting emails, writing social media posts or legal documentation. By understanding context, tone and style, NLP sees to it that the generated content is coherent, relevant and aligned with the intended message, saving time and effort in content creation while maintaining quality.

NLP Use Cases

1. Customer Service : While NLP-powered chatbots and callbots are most common in customer service contexts, companies have also relied on natural language processing to power virtual assistants. These assistants are a form of conversational AI that can carry on more sophisticated discussions. And if NLP is unable to resolve an issue, it can connect a customer with the appropriate personnel.
2. Marketing : Gathering market intelligence becomes much easier with natural language processing, which can analyze online reviews, social media posts and web forums. Compiling this data can help marketing teams understand what consumers care about and how they perceive a business' brand.
3. Human Resources: Recruiters and HR personnel can use natural language processing to sift through hundreds of resumes, picking out promising candidates based on keywords, education, skills and other criteria. In addition, NLP's data analysis capabilities are ideal for reviewing employee surveys and quickly determining how employees feel about the workplace.
4. E-Commerce: Natural language processing can help customers book tickets, track orders and even recommend similar products on e-commerce websites. Teams can also use data on customer purchases to inform what types of products to stock up on and when to replenish inventories.
5. Finance: In finance, NLP can be paired with machine learning to generate financial reports based on invoices, statements and other documents.

Financial analysts can also employ natural language processing to predict stock market trends by analyzing news articles, social media posts and other online sources for market sentiments.

6. Insurance: Insurance companies can assess claims with natural language processing since this technology can handle both structured and unstructured data. NLP can also be trained to pick out unusual information, allowing teams to spot fraudulent claims.
7. Education: NLP-powered apps can check for spelling errors, highlight unnecessary or misapplied grammar and even suggest simpler ways to organize sentences. Natural language processing can also translate text into other languages, aiding students in learning a new language.
8. Healthcare: Healthcare professionals can develop more efficient workflows with the help of natural language processing. During procedures, doctors can dictate their actions and notes to an app, which produces an accurate transcription. NLP can also scan patient documents to identify patients who would be best suited for certain clinical trials.
9. Manufacturing: With its ability to process large amounts of data, NLP can inform manufacturers on how to improve production workflows, when to perform machine maintenance and what issues need to be fixed in products. And if companies need to find the best price for specific materials, natural language processing can review various websites and locate the optimal price.
10. Cybersecurity: IT and security teams can deploy natural language processing to filter out suspicious emails based on word choice, sentiment and other factors. This makes it easier to protect different departments from spam, phishing scams and other cyber attacks. With its ability to understand data, NLP can also detect unusual behavior and alert teams of possible threats.

Challenges of NLP :

Even state-of-the-art NLP models are not perfect, just as human speech is prone to error. As with any AI technology, NLP comes with potential pitfalls. Human language is filled with ambiguities that make it difficult for programmers to write software that accurately determines the intended meaning of text or voice data. Human language might take years for humans to learn—and many never stop learning. But then programmers must teach natural language-powered applications to recognize and understand irregularities so their applications can be accurate and useful. Associated risks might include:

- Biased training: As with any AI function, biased data used in training will skew the answers. The more diverse the users of an NLP function, the more significant this risk becomes, such as in government services, healthcare and

HR interactions. Training datasets scraped from the web, for example, are prone to bias.

- **Misinterpretation:** As in programming, there is a risk of garbage in, garbage out (GIGO). Speech recognition, also known as speech-to-text, is the task of reliably converting voice data into text data. But NLP solutions can become confused if spoken input is in an obscure dialect, mumbled, too full of slang, homonyms, incorrect grammar, idioms, fragments, mispronunciations, contractions or recorded with too much background noise.
- **New vocabulary:** New words are continually being invented or imported. The conventions of grammar can evolve or be intentionally broken. In these cases, NLP can either make a best guess or admit it's unsure—and either way, this creates a complication.
- **Tone of voice:** When people speak, their verbal delivery or even body language can give an entirely different meaning than the words alone. Exaggeration for effect, stressing words for importance or sarcasm can be confused by NLP, making the semantic analysis more difficult and less reliable.

Levels of NLP:

The process of NLP can be divided into five distinct phases: Lexical Analysis, Syntactic Analysis, Semantic Analysis, Discourse Integration, and Pragmatic Analysis. Each phase plays a crucial role in the overall understanding and processing of natural language.

First Phase of NLP: Lexical and Morphological Analysis

Tokenization: The lexical phase in Natural Language Processing (NLP) involves scanning text and breaking it down into smaller units such as paragraphs, sentences, and words. This process, known as tokenization, converts raw text into manageable units called tokens or lexemes. Tokenization is essential for understanding and processing text at the word level.

In addition to tokenization, various data cleaning and feature extraction techniques are applied, including:

Lemmatization: Reducing words to their base or root form.

Stopwords Removal: Eliminating common words that do not carry significant meaning, such as "and," "the," and "is."

Correcting Misspelled Words: Ensuring the text is free of spelling errors to maintain accuracy.

These steps enhance the comprehensibility of the text, making it easier to analyze and process.

Morphological Analysis: Morphological analysis is another critical phase in NLP, focusing on identifying morphemes, the smallest units of a word that carry

meaning and cannot be further divided. Understanding morphemes is vital for grasping the structure of words and their relationships.

Types of Morphemes

Free Morphemes: Text elements that carry meaning independently and make sense on their own. For example, "bat" is a free morpheme.

Bound Morphemes: Elements that must be attached to free morphemes to convey meaning, as they cannot stand alone. For instance, the suffix "-ing" is a bound morpheme, needing to be attached to a free morpheme like "run" to form "running."

Importance of Morphological Analysis

Morphological analysis is crucial in NLP for several reasons:

- **Understanding Word Structure:** It helps in deciphering the composition of complex words.
- **Predicting Word Forms:** It aids in anticipating different forms of a word based on its morphemes.
- **Improving Accuracy:** It enhances the accuracy of tasks such as part-of-speech tagging, syntactic parsing, and machine translation.
- **By identifying and analyzing morphemes,** the system can interpret text correctly at the most fundamental level, laying the groundwork for more advanced NLP applications.

Second Phase of NLP: Syntactic Analysis (Parsing)

Syntactic analysis, also known as parsing, is the second phase of Natural Language Processing (NLP). This phase is essential for understanding the structure of a sentence and assessing its grammatical correctness. It involves analyzing the relationships between words and ensuring their logical consistency by comparing their arrangement against standard grammatical rules.

Role of Parsing: Parsing examines the grammatical structure and relationships within a given text. It assigns Parts-Of-Speech (POS) tags to each word, categorizing them as nouns, verbs, adverbs, etc. This tagging is crucial for understanding how words relate to each other syntactically and helps in avoiding ambiguity. Ambiguity arises when a text can be interpreted in multiple ways due to words having various meanings. For example, the word "book" can be a noun (a physical book) or a verb (the action of booking something), depending on the sentence context.

Examples of Syntax: Consider the following sentences:

Correct Syntax: "John eats an apple."

Incorrect Syntax: "Apple eats John an."

Despite using the same words, only the first sentence is grammatically correct and makes sense. The correct arrangement of words according to grammatical rules is what makes the sentence meaningful.

Assigning POS Tags: During parsing, each word in the sentence is assigned a POS tag to indicate its grammatical category. Here's an example breakdown:

Sentence: "John eats an apple."

POS Tags:

John: Proper Noun (NNP)

eats: Verb (VBZ)

an: Determiner (DT)

apple: Noun (NN)

Assigning POS tags correctly is crucial for understanding the sentence structure and ensuring accurate interpretation of the text.

Importance of Syntactic Analysis

By analyzing and ensuring proper syntax, NLP systems can better understand and generate human language. This analysis helps in various applications, such as machine translation, sentiment analysis, and information retrieval, by providing a clear structure and reducing ambiguity.

Third Phase of NLP: **Semantic Analysis**

Semantic Analysis is the third phase of Natural Language Processing (NLP), focusing on extracting the meaning from text. Unlike syntactic analysis, which deals with grammatical structure, semantic analysis is concerned with the literal and contextual meaning of words, phrases, and sentences.

Semantic analysis aims to understand the dictionary definitions of words and their usage in context. It determines whether the arrangement of words in a sentence makes logical sense. This phase helps in finding context and logic by ensuring the semantic coherence of sentences.

Key Tasks in Semantic Analysis

Named Entity Recognition (NER): NER identifies and classifies entities within the text, such as names of people, places, and organizations. These entities belong to predefined categories and are crucial for understanding the text's content.

Word Sense Disambiguation (WSD): WSD determines the correct meaning of ambiguous words based on context. For example, the word "bank" can refer to a financial institution or the side of a river. WSD uses contextual clues to assign the appropriate meaning.

Examples of Semantic Analysis: Consider the following examples:

Syntactically Correct but Semantically Incorrect: "Apple eats a John."

This sentence is grammatically correct but does not make sense semantically. An apple cannot eat a person, highlighting the importance of semantic analysis in ensuring logical coherence.

Literal Interpretation: "What time is it?"

This phrase is interpreted literally as someone asking for the current time, demonstrating how semantic analysis helps in understanding the intended meaning.

Importance of Semantic Analysis

Semantic analysis is essential for various NLP applications, including machine translation, information retrieval, and question answering. By ensuring that sentences are not only grammatically correct but also meaningful, semantic analysis enhances the accuracy and relevance of NLP systems.

Fourth Phase of NLP: **Discourse Integration**

Discourse Integration is the fourth phase of Natural Language Processing (NLP). This phase deals with comprehending the relationship between the current sentence and earlier sentences or the larger context. Discourse integration is crucial for contextualizing text and understanding the overall message conveyed.

Role of Discourse Integration

Discourse integration examines how words, phrases, and sentences relate to each other within a larger context. It assesses the impact a word or sentence has on the structure of a text and how the combination of sentences affects the overall meaning. This phase helps in understanding implicit references and the flow of information across sentences.

Importance of Contextualization

In conversations and texts, words and sentences often depend on preceding or following sentences for their meaning. Understanding the context behind these words and sentences is essential to accurately interpret their meaning.

Example of Discourse Integration: Consider the following examples:

Contextual Reference: "This is unfair!"

To understand what "this" refers to, we need to examine the preceding or following sentences. Without context, the statement's meaning remains unclear.

Anaphora Resolution: "Taylor went to the store to buy some groceries. She realized she forgot her wallet."

In this example, the pronoun "she" refers back to "Taylor" in the first sentence. Understanding that "Taylor" is the antecedent of "she" is crucial for grasping the sentence's meaning.

Application of Discourse Integration

Discourse integration is vital for various NLP applications, such as machine translation, sentiment analysis, and conversational agents. By understanding the

relationships and context within texts, NLP systems can provide more accurate and coherent responses.

Fifth Phase of NLP: **Pragmatic Analysis**

Pragmatic Analysis is the fifth and final phase of Natural Language Processing (NLP), focusing on interpreting the inferred meaning of a text beyond its literal content. Human language is often complex and layered with underlying assumptions, implications, and intentions that go beyond straightforward interpretation. This phase aims to grasp these deeper meanings in communication.

Role of Pragmatic Analysis: Pragmatic analysis goes beyond the literal meanings examined in semantic analysis, aiming to understand what the writer or speaker truly intends to convey. In natural language, words and phrases can carry different meanings depending on context, tone, and the situation in which they are used.

Importance of Understanding Intentions: In human communication, people often do not say exactly what they mean. For instance, the word "Hello" can have various interpretations depending on the tone and context in which it is spoken. It could be a simple greeting, an expression of surprise, or even a signal of anger. Thus, understanding the intended meaning behind words and sentences is crucial.

Examples of Pragmatic Analysis: Consider the following examples:

Contextual Greeting: "Hello! What time is it?"

"Hello!" is more than just a greeting; it serves to establish contact.

"What time is it?" might be a straightforward request for the current time, but it could also imply concern about being late.

Figurative Expression: "I'm falling for you."

The word "falling" literally means collapsing, but in this context, it means the speaker is expressing love for someone.

Application of Pragmatic Analysis: Pragmatic analysis is essential for applications like sentiment analysis, conversational AI, and advanced dialogue systems. By interpreting the deeper, inferred meanings of texts, NLP systems can understand human emotions, intentions, and subtleties in communication, leading to more accurate and human-like interactions.

Regular Expression: Regular expression is a sequence of characters that forms a pattern which is mainly used to find or replace patterns in a string.

These are supported by many languages such as python, java, R etc. Most common uses of regular expressions are:

1. Finding patterns in a string or file.(Ex: find all the numbers present in a string)
2. Replace a part of the string with another string.
3. Search substring in string or file.

4. Split string into substrings.
5. Validate email format.

Meta Characters: These are the characters which have special meaning. The following are some of the meta characters with their uses.

	Meta character	Description
1	[](Square brackets)	This matches any single character in this bracket with the given string.
2	.(Period)	This matches all the characters except the newline. If we pass this as a pattern in the findall() function it will match with all the characters present in the string except newline characters.
3	^(Carret)	This matches the given pattern at the start of the string. This is used to check if the string starts with a particular pattern or not.
4	\$(Dollar)	This matches the given pattern at the end of string. This is used to check if the string ends with a pattern or not.
5	*(Star)	This matches 0 or more occurrences of the pattern to its left.
6	+(Plus)	This matches 1 or more occurrences of the pattern to its left.
7	?(Question mark)	This matches 0 or 1 occurrence of the pattern to its left.
8	{ } (Braces)	This matches the specified number of occurrences of pattern present in the braces.
9	(Alternation)	This works like 'or' condition. In this we can give two or more patterns. If the string contains at least one of the given patterns this will give a match.
10	() (Group)	This is used to group various regular expressions together and then find a match in the string.
11	\ (Backslash)	This is used to match special sequences or can be used as escape characters also.

Special Sequences in Python RegEx:

SNo	Sequence	Description
1	\A	This gives a match if the characters to the right of this are at the beginning of the string.
2	\b	This gives a match if the characters to the right are at the beginning of a word or the characters to the left are at the end of a word in the given string.
3	\B	This gives a match if the characters to the right or left of \B are not

		present at the beginning or end of a word in the given string.
4	<code>\d</code>	This gives a match if the string contains a digit.
5	<code>\D</code>	This gives a match if the string contains only non digit characters.
6	<code>\s</code>	This gives a match if the string contains a white space character.
7	<code>\S</code>	This gives a match if the string contains only characters other than white space character.
8	<code>\w</code>	This gives a match if the string contains any character in a-z, A-Z, 0-9 and underscore(_).
9	<code>\W</code>	This gives a match if the string contains characters other than a-z, A-Z, 0-9 and underscore(_).
10	<code>\Z</code>	This gives a match if the characters to the left of <code>\Z</code> are present at the end of the string.

Regular Expressions and Their Purpose in Text Processing:

Regular expressions, often abbreviated as “regex” or “regexp,” are powerful and flexible patterns used in text processing to search for, match, and manipulate text based on specific criteria. They serve the following purposes in text processing:

1. **Pattern Matching:** Regular expressions allow us to define patterns or templates that describe sets of strings with common characteristics. We can use regex to search for occurrences of these patterns within text data.
2. **Search and Extraction:** We can use regular expressions to search for specific substrings or patterns within a larger text and extract or capture those matches. This is useful for tasks like finding email addresses, phone numbers, or dates in text.
3. **Validation:** Regular expressions are frequently used to validate whether a given string conforms to a particular format or structure. For example, we can use regex to validate input forms, such as checking if a user-provided email address is valid.
4. **Text Manipulation:** Regular expressions allow us to replace or transform text based on patterns. For instance, we can find and replace all instances of a word in a document or format dates in a consistent way.
5. **Text Parsing:** Regular expressions are instrumental in parsing structured data formats like CSV, JSON, or XML. We can use regex to identify and extract data fields within these formats.
6. **Text Cleaning:** Regular expressions are useful for text preprocessing tasks like removing whitespace, stripping HTML tags from web content, or stripping unwanted characters.
7. **Tokenization:** In natural language processing (NLP), regex can be used for tokenization, which involves splitting text into meaningful units like words or sentences.

8. Pattern Validation: Regex can be employed to enforce specific patterns or constraints, such as password strength rules, in applications.

Advantages of Regular Expressions: Regular expressions offer several advantages.

1. Pattern Flexibility: Regular expressions allow us to define highly flexible patterns for matching text. We can specify patterns that match a wide range of variations, making them suitable for handling diverse input data.
2. Concise Notation: Regular expressions provide a concise and expressive way to represent complex patterns. Instead of writing custom code to handle different cases, we can often achieve the same result with a single regex pattern.
3. Pattern Modularity: We can build complex patterns by combining simpler patterns, which promotes code modularity and reusability. This makes it easier to understand and maintain regular expressions.
4. Compact Code: Using regular expressions often results in shorter and more compact code compared to equivalent procedural code for text processing tasks. This can lead to more readable and maintainable codebases.
5. Efficiency: In many cases, regex engines are highly optimized for pattern matching, which can lead to efficient and fast text processing, especially when dealing with large datasets.
6. Consistent Syntax: Regex syntax is standardized across many programming languages and text editors. Once we learn regular expressions, we can apply our knowledge to various contexts without having to learn new syntax.

Term frequency: TF measures the frequency of a term within a document. It is calculated as the ratio of the number of times a term occurs in a document to the total number of terms in that document. The goal is to emphasize words that are frequent within a document.

$$TF(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, “Data Science is awesome!” One way to start out is to eliminate documents that don’t contain all three words: “Data,” “is,” “Science” and “awesome,” but this still leaves too many documents. To further distinguish them, we might count the number of times each term occurs in each document. The number of times a term occurs in a document is called its term frequency.

The weight of a term that occurs in a document is simply proportional to the term frequency.

Inverse Document Frequency (IDF): IDF measures the rarity of a term across a collection of documents. It is calculated as the logarithm of the ratio of the total number of documents to the number of documents containing the term. The goal is to penalize words that are common across all documents.

$$IDF(t, D) = \log \left(\frac{\text{Total number of documents in the corpus } N}{\text{Number of documents containing term } t} \right)$$

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

TF-IDF in NLP (Term Frequency Inverse Document Frequency): In the realm of Natural Language Processing (NLP), TF-IDF (Term Frequency-Inverse Document Frequency) is a powerful technique used to analyze and understand the importance of words in a document corpus. TF-IDF plays a crucial role in tasks such as text mining, information retrieval, and document classification. Let’s delve into the concepts and applications of TF-IDF in NLP.

TF-IDF is a numerical statistic that reflects the significance of a word within a document relative to a collection of documents, known as a corpus. The idea behind TF-IDF is to quantify the importance of a term in a document with respect to its frequency in the document and its rarity across multiple documents. Words within a text document are transformed into importance numbers by a text vectorization process. There are many different text vectorization scoring schemes, with TF-IDF being one of the most common.

TF-IDF is useful in many natural language processing applications. For example, Search Engines use TF-IDF to rank the relevance of a document for a query. TF-IDF is also employed in text classification, text summarization, and topic modeling.

Combining TF and IDF: TF-IDF

The TF-IDF score for a term in a document is obtained by multiplying its TF and IDF scores.

$$\text{TF-IDF}(t,d,D)=\text{TF}(t,d)\times\text{IDF}(t,D)$$

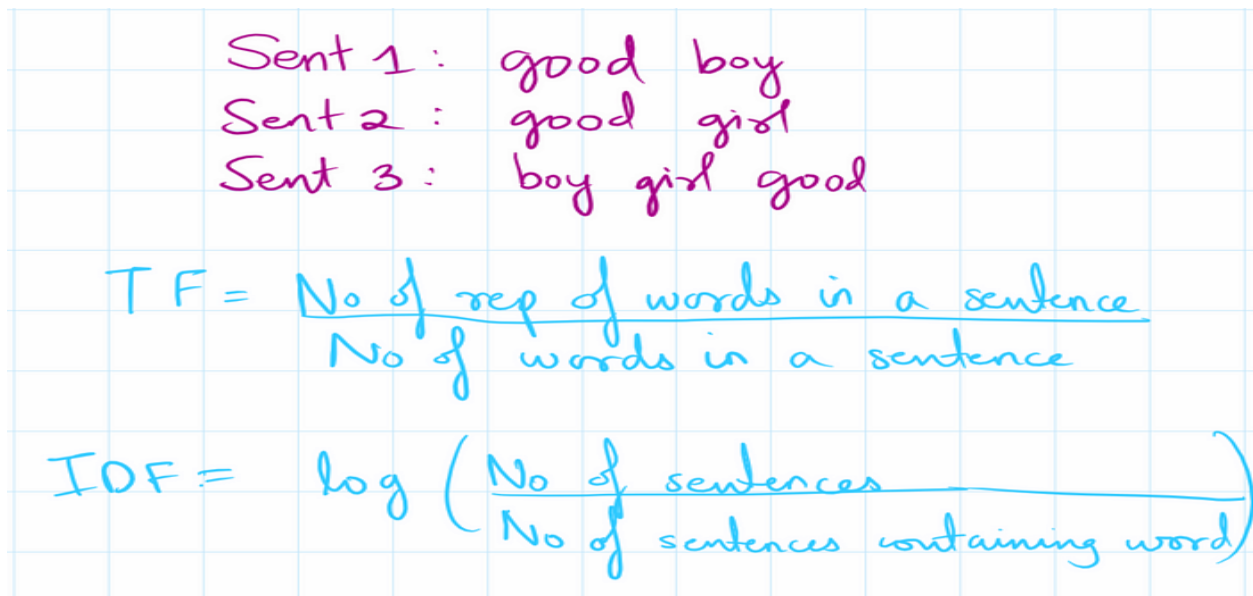
CODE:

```
from sklearn.feature_extraction.text import TfidfVectorizer
corpus = ["good boy", "good girl", "good boy girl"]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)
print(X.toarray())
```

Output:

```
[[0.78980693 0.        0.61335554]
 [0.        0.78980693 0.61335554]
 [0.61980538 0.61980538 0.48133417]]
```

Inside this array of array the first number represents boy, the second number represents girl and the third number represents good
The length of the words in the corpus is 3 and they are arranged in ascending order.



Sent 1: good boy
Sent 2: good girl
Sent 3: boy girl good

$$\text{TF} = \frac{\text{No of rep of words in a sentence}}{\text{No of words in a sentence}}$$
$$\text{IDF} = \log \left(\frac{\text{No of sentences}}{\text{No of sentences containing word}} \right)$$

	S1	S2	S3
good	1/2	1/2	1/3
boy	1/2	0	1/3
girl	0	1/2	1/3

good	$\log(3/3) = 0$
boy	$\log(3/2)$
girl	$\log(3/2)$

	good	boy	girl
Sent 1	0	$\frac{1}{2} \times \log(\frac{3}{2})$	0
Sent 2	0	0	$\frac{1}{2} \log(\frac{3}{2})$
Sent 3	0	$\frac{1}{3} \times \log(\frac{3}{2})$	$\frac{1}{3} \log(\frac{3}{2})$

$$\frac{1}{2} \log\left(\frac{3}{2}\right) = 0.0880$$

$$\frac{1}{3} \log\left(\frac{3}{2}\right) = 0.058$$

NOTE

The theoretical value and the actual value is being different by a significant amount.

It is because inside of the scikit library the formula that is being used for IDF is

- For each term, calculate the inverse document frequency across the entire corpus.
- $IDF(t, D) = \log\left(\frac{\text{Total number of documents in the corpus } N}{\text{Number of documents containing term } t}\right) + 1$

Part Of Speech Tagging – POS Tagging in NLP:

Part of Speech is the classification of words based on their role in the sentence. The major POS tags are Nouns, Verbs, Adjectives, Adverbs. This category provides more details about the word and its meaning in the context. A sentence consists of words with a sensible Part of Speech structure.

For example: Book the flight!

This sentence contains Noun (Book), Determinant (the) and a Verb (flight).

POS tagging refers to the automatic assignment of a tag to words in a given sentence. It converts a sentence into a list of words with their tags. (word, tag). Since this task involves considering the sentence structure, it cannot be done at the Lexical level. A POS tagger considers surrounding words while assigning a tag.

For example, the previous sentence, “Book the flight”, will become a list of each word with its corresponding POS tag – [(“Book”, “Verb”), (“the”, “Det”), (“flight”, “Noun”)].

Difficulties in POS Tagging:

Similar to most NLP problems, POS tagging suffers from ambiguity. In the sentences, “Book the flight” and “I like to read books”, we see that book can act as a Verb or Noun. Similarly, many words in the English dictionary has multiple possible POS tags.

- This (Preposition) is a car
- This (Determiner) car is red
- You can go this (Adverb) far only.

These sentences use the word “This” in various contexts. However, how can one assign the correct tag to the words?

POS tagging techniques.

1. **Rule-Based Tagging:** Rule-based tagging is the oldest tagging approach where we use contextual information to assign tags to unknown or ambiguous words.

The rule-based approach uses a dictionary to get possible tags for tagging each word. If the word has more than one possible tag, then rule-based taggers use hand-written rules to identify the correct tag.

Since rules are usually built manually, therefore they are also called Knowledge-driven taggers. We have a limited number of rules, approximately around 1000 for the English language.

One of example of a rule is as follows:

Sample Rule: If an ambiguous word “X” is preceded by a determiner and followed by a noun, tag it as an adjective;

A nice car: nice is an ADJECTIVE here.

Limitations/Disadvantages of Rule-Based Approach:

- High development cost and high time complexity when applying to a large corpus of text
- Defining a set of rules manually is an extremely cumbersome process and is not scalable at all

2 **Stochastic POS Tagging:** Stochastic POS Tagger uses probabilistic and statistical information from the corpus of labeled text (where we know the actual tags of words in the corpus) to assign a POS tag to each word in a sentence.

This tagger can use techniques like Word frequency measurements and Tag Sequence Probabilities. It can either use one of these approaches or a combination of both. Let’s discuss these techniques in detail.

Word Frequency Measurements

The tag encountered most frequently in the corpus is the one assigned to the ambiguous words(words having 2 or more possible POS tags).

Let's understand this approach using some example sentences :

Ambiguous Word = "play"

Sentence 1 : I play cricket every day. POS tag of play = VERB

Sentence 2 : I want to perform a play. POS tag of play = NOUN

The word frequency method will now check the most frequently used POS tag for "play". Let's say this frequent POS tag happens to be VERB; then we assign the POS tag of "play" = VERB

The main drawback of this approach is that it can yield invalid sequences of tags.

Tag Sequence Probabilities: In this method, the best tag for a given word is determined by the probability that it occurs with "n" previous tags.

Simply put, assume we have a new sequence of 4 words,

w1w2w3w4

w1w2w3w4

And we need to identify the POS tag of w4

If n = 3, we will consider the POS tags of 3 words prior to w4 in the labeled corpus of text

Let's say the POS tags for

w1= NOUN,

w2= VERB ,

w3 = DETERMINER

In short, N, V, D: NVD

Then in the labeled corpus of text, we will search for this NVD sequence.

Let's say we found 100 such NVD sequences. Out of these -

10 sequences have the POS of the next word is NOUN 90 sequences have the POS of the next word is VERB

Then the POS of the word

w 4= VERB

The main drawback of this technique is that sometimes the predicted sequence is not Grammatically correct.

Now let's discuss some properties and limitations of the Stochastic tagging approach :

- This POS tagging is based on the probability of the tag occurring (either solo or in sequence)
- It requires labeled corpus, also called training data in the Machine Learning lingo
- There would be no probability for the words that don't exist in the training data
- It uses a different testing corpus (unseen text) other than the training corpus

- It is the simplest POS tagging because it chooses the most frequent tags associated with a word in the training corpus

Transformation-Based Learning Tagger: TBL

Transformation-based tagging is the combination of Rule-based & stochastic tagging methodologies. In Layman's terms; The algorithm keeps on searching for the new best set of rules given input as labeled corpus until its accuracy saturates the labeled corpus.

Algorithm takes following Input:

- a tagged corpus
- a dictionary of words with the most frequent tags
- Output : Sequence of transformation rules

Example of sample rule learned by this algorithm:

Rule : Change Noun(NN) to Verb(VB) when previous tag is To(TO)

E.g.: race has the following probabilities in the Brown corpus -

Probability of tag is NOUN given word is race $P(NN | \text{race}) = 98\%$

Probability of tag is VERB given word is race $P(VB | \text{race}) = 0.02$

Given sequence: is expected to race tomorrow

First tag race with NOUN (since its probability of being NOUN is 98%)

Then apply the above rule and re-tag the POS of race with VERB (since just the previous tag before the "race" word is TO)

The Working of the TBL Algorithm

- Step 1: Label every word with the most likely tag via lookup from the input dictionary.
- Step 2: Check every possible transformation & select one which most improves tagging accuracy.

Similar to the above sample rule, other possible (maybe worst transformations) rules could be -

Change Noun(NN) to Determiner(DT) when previous tag is To(TO)

Change Noun(NN) to Adverb(RB) when previous tag is To(TO)

Change Noun(NN) to Adjective(JJ) when previous tag is To(TO)

etc.....

- Step 3: Re-tag corpus by applying all possible transformation rules
- Repeat Step 1,2,3 as many times as needed until accuracy saturates or you reach some predefined accuracy cutoff.

Advantages: We can learn a small set of simple rules, and these rules are decent enough for basic POS tagging

- Development, as well as debugging, is very easy in TBL because the learned rules are easy to understand
- Complexity in tagging is reduced because, in TBL, there is a cross-connection between machine-learned and human-generated rules

Drawbacks: Despite being a simple and somewhat effective approach to POS tagging, TBL has major disadvantages.

- TBL algorithm training/learning time complexity is very high, and time increases multi-fold when corpus size increases
- TBL does not provide tag probabilities

Named Entity Recognition (NER):

Named Entity Recognition (NER) serves as a bridge between unstructured text and structured data, enabling machines to sift through vast amounts of textual information and extract nuggets of valuable data in categorized forms. By pinpointing specific entities within a sea of words, NER transforms the way we process and utilize textual data.

Purpose: NER's primary objective is to comb through unstructured text and identify specific chunks as named entities, subsequently classifying them into predefined categories. This conversion of raw text into structured information makes data more actionable, facilitating tasks like data analysis, information retrieval, and knowledge graph construction.

How it works: The intricacies of NER can be broken down into several steps:

- Tokenization. Before identifying entities, the text is split into tokens, which can be words, phrases, or even sentences. For instance, "Steve Jobs co-founded Apple" would be split into tokens like "Steve", "Jobs", "co-founded", "Apple".
- Entity identification. Using various linguistic rules or statistical methods, potential named entities are detected. This involves recognizing patterns, such as capitalization in names ("Steve Jobs") or specific formats (like dates).
- Entity classification. Once entities are identified, they are categorized into predefined classes such as "Person", "Organization", or "Location". This is often achieved using machine learning models trained on labeled datasets.

For our example, "Steve Jobs" would be classified as a "Person" and "Apple" as an "Organization".

- Contextual analysis. NER systems often consider the surrounding context to improve accuracy. For instance, in the sentence "Apple released a new iPhone", the context helps the system recognize "Apple" as an organization rather than a fruit.
- Post-processing. After initial recognition and classification, post-processing might be applied to refine results. This could involve resolving ambiguities, merging multi-token entities, or using knowledge bases to enhance entity data.
- The beauty of NER lies in its ability to understand and interpret unstructured text, which constitutes a significant portion of the data in the digital world, from web pages and news articles to social media posts and research papers. By identifying and classifying named entities, NER adds a layer of structure and meaning to this vast textual landscape.

Named Entity Recognition Methods

Named Entity Recognition (NER) has seen many methods developed over the years, each tailored to address the unique challenges of extracting and categorizing named entities from vast textual landscapes.

1 Rule-based Methods: Rule-based methods are grounded in manually crafted rules. They identify and classify named entities based on linguistic patterns, regular expressions, or dictionaries. While they shine in specific domains where entities are well-defined, such as extracting standard medical terms from clinical notes, their scalability is limited. They might struggle with large or diverse datasets due to the rigidity of predefined rules.

2 Statistical Methods: Transitioning from manual rules, statistical methods employ models like Hidden Markov Models (HMM) or Conditional Random Fields (CRF). They predict named entities based on likelihoods derived from training data. These methods are apt for tasks with ample labeled datasets at their disposal. Their strength lies in generalizing across diverse texts, but they're only as good as the training data they're fed.

3. Machine Learning Methods: Machine learning methods take it a step further by using algorithms such as decision trees or support vector machines. They learn from labeled data to predict named entities. Their widespread adoption in modern NER systems is attributed to their prowess in handling vast datasets and intricate patterns. However, they're hungry for substantial labeled data and can be computationally demanding.

4 Deep Learning Methods: The latest in the line are deep learning methods, which harness the power of neural networks. Recurrent Neural Networks (RNN) and

transformers have become the go-to for many due to their ability to model long-term dependencies in text. They're ideal for large-scale tasks with abundant training data but come with the caveat of requiring significant computational might.

5 Hybrid Methods: Lastly, there's no one-size-fits-all in NER, leading to the emergence of hybrid methods. These techniques intertwine rule-based, statistical, and machine learning approaches, aiming to capture the best of all worlds. They're especially valuable when extracting entities from diverse sources, offering the flexibility of multiple methods. However, their intertwined nature can make them complex to implement and maintain.

Named Entity Recognition Use Cases: NER has found applications across diverse sectors, transforming the way we extract and utilize information. Here's a glimpse into some of its pivotal applications:

1. News aggregation. NER is instrumental in categorizing news articles by the primary entities mentioned. This categorization aids readers in swiftly locating stories about specific people, places, or organizations, streamlining the news consumption process.
2. Customer support. Analyzing customer queries becomes more efficient with NER. Companies can swiftly pinpoint common issues related to specific products or services, ensuring that customer concerns are addressed promptly and effectively.
3. Research. For academics and researchers, NER is a boon. It allows them to scan vast volumes of text, identifying mentions of specific entities relevant to their studies. This automated extraction speeds up the research process and ensures comprehensive data analysis.
4. Legal document analysis. In the legal sector, sifting through lengthy documents to find relevant entities like names, dates, or locations can be tedious. NER automates this, making legal research and analysis more efficient.