21CSE356T – Natural Language Processing

"Enabling Computers to Understand Natural Language like Humans"

Department of Computational Intelligence

School of computing

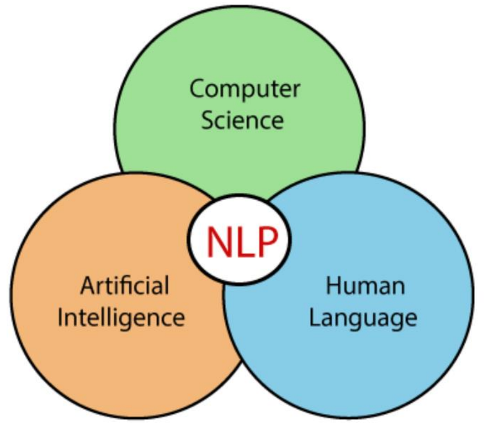
SRM Institute of Science and Technology

Introduction to NLP

- Motive of learning a language is to communicate and share the information successfully to others.
- According to the industry the estimation is only 21% of the available data is in the structured form.
- Data is being generated ,send messages on WhatsApp and Facebook or various social media.
- Majority of the datas exists in textual format which is highly unstructured form.
- Now in order to produce significant and actionable insights from this data it is important to get acquainted with the techniques of text analysis and natural language processing.

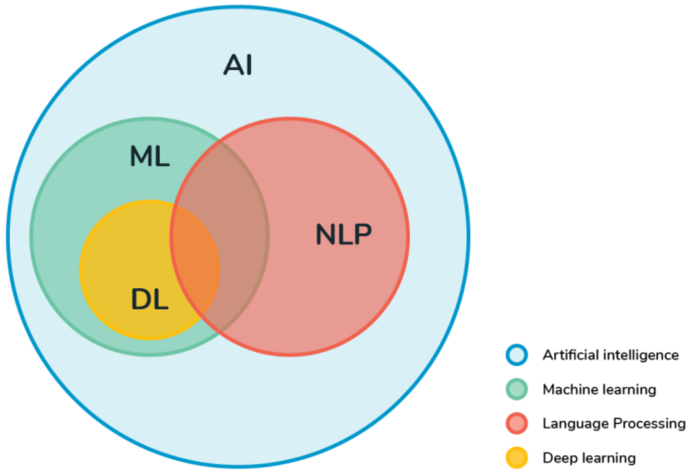
Introduction to NLP

- Text analytics/mining is the process of deriving meaningful information from natural language text.
- It usually involves the process of structuring the input text, deriving patterns and evaluating and interpreting the output.
- Natural language processing is a part of computer science and artificial intelligence which deals with human languages.



What is NLP?

- Natural language processing (NLP) is a branch of artificial intelligence (AI) that enables machines to understand human language.
- The main intention of NLP is to build systems that are able to make sense of text and
- Then automatically execute tasks like spell-check, text translation, topic classification, etc.



Need of NLP

- The need to study Natural Language Processing (NLP)
  - **Increasing role of language in human-computer interaction**
  - **Vast amount of unstructured textual data available.**

- Now to make interactions between computers and humans, computers need to understand natural languages used by humans.

- Natural language processing is all about making computers **learn, process, and manipulate natural languages**.

Need of NLP

- **Processing text data** is an essential task as there is an abundance of text available everywhere.
- Text data can be found in various sources such as **books, websites, social media, news articles, research papers, emails, and more.**
- However, text data is often **unstructured**, meaning it lacks a predefined format or organization.
- To harness the valuable information contained within text data, it is necessary to **process and analyze** it effectively.

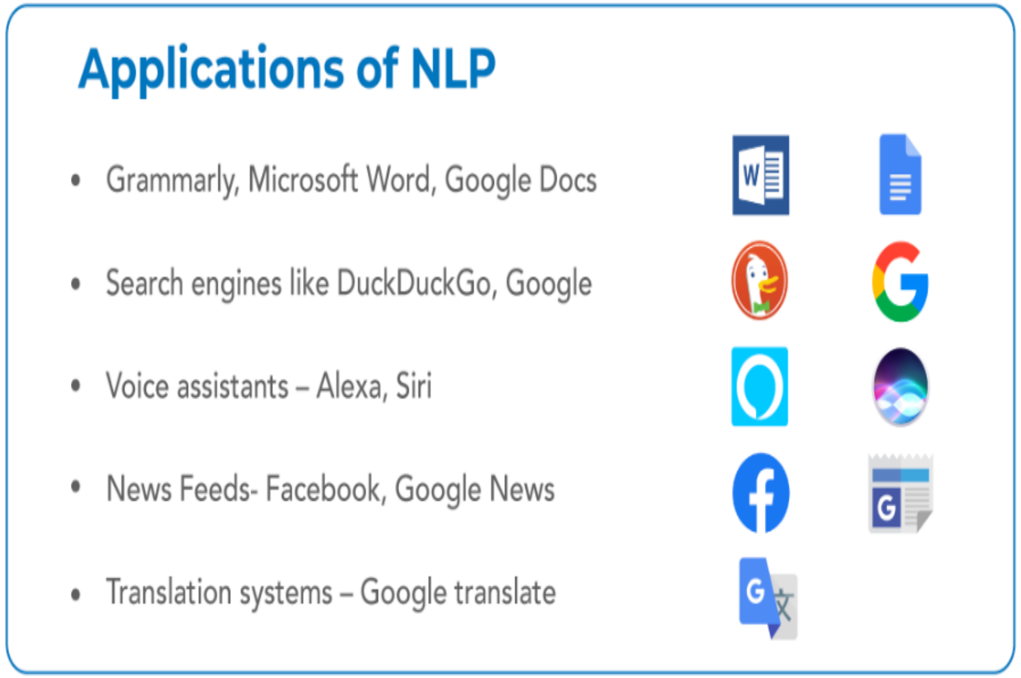
Need of NLP

- To extract insights, identify patterns, perform sentiment analysis, categorize documents, automate text generation, and enable information retrieval.
- By processing text data, **valuable knowledge can be derived**, enabling businesses, researchers, and individuals to make **informed decisions, gain insights, improve customer experiences, develop intelligent systems, and drive innovation across various industries**.

Use of NLP

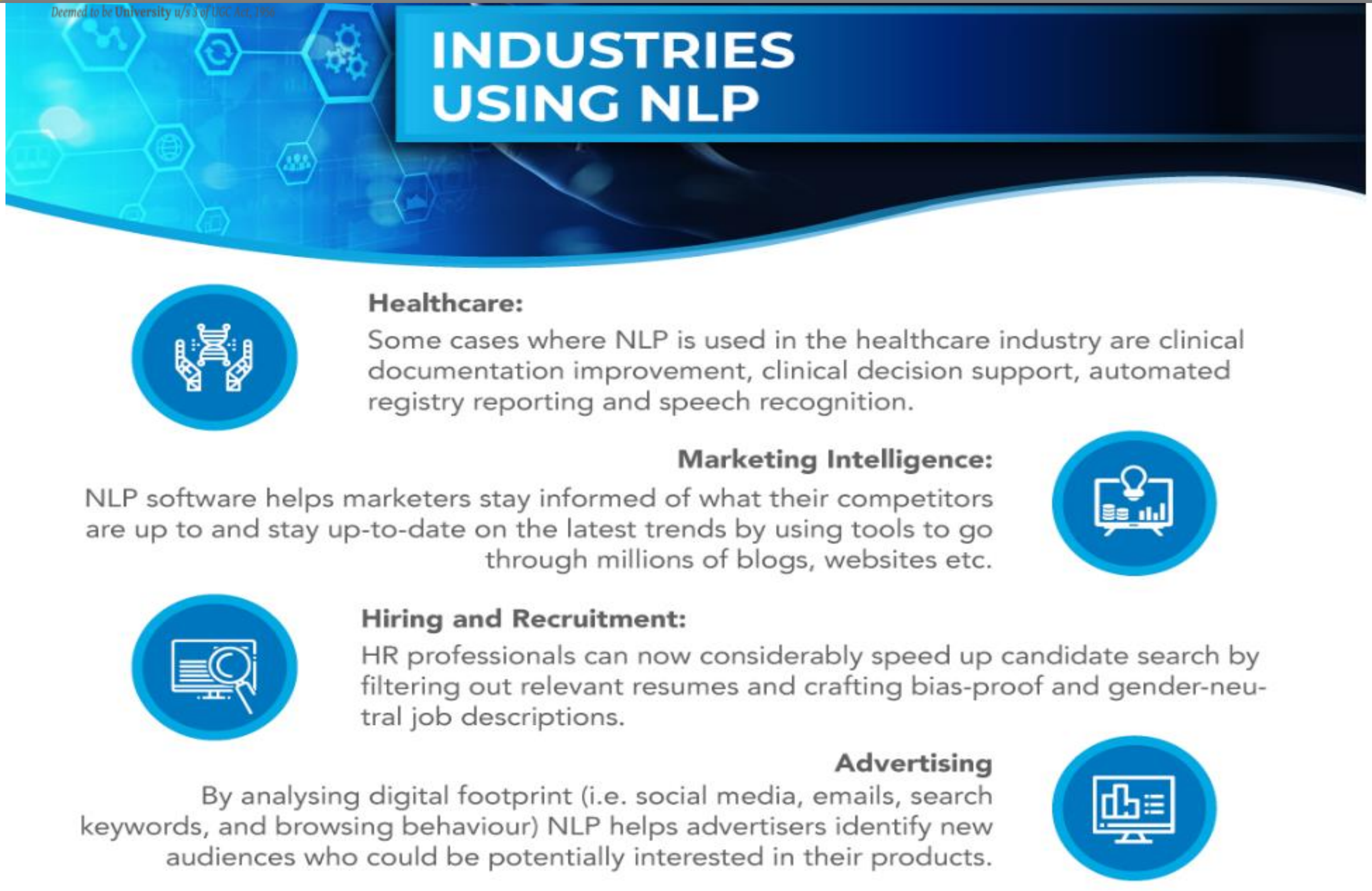
1. **User-Friendly Interfaces:** NLP allows for intuitive and user-friendly interfaces using natural language, *reducing the need for complex programming syntax*.
2. **Accessibility and Inclusivity:** NLP makes technology accessible to a wider audience, including those with *limited technical expertise or disabilities*.
3. **Conversational Systems:** NLP enables the development of conversational agents, *enhancing user interaction and system efficiency*.
4. **Data Extraction and Analysis:** NLP extracts insights from unstructured text data, enabling sentiment analysis, information retrieval, and text summarization.
5. **Voice-based Interaction:** NLP powers voice assistants and speech recognition systems for hands-free and natural interaction.
6. **Human-Machine Collaboration:** NLP enables seamless communication and collaboration between humans and machines.
7. **Natural Language Understanding:** NLP allows machines to comprehend context, semantics, and intent, enabling advanced applications and personalized experiences.

Applications of NLP



Applications of NLP

- Grammarly, Microsoft Word, Google Docs
- Search engines like DuckDuckGo, Google
- Voice assistants – Alexa, Siri
- News Feeds- Facebook, Google News
- Translation systems – Google translate



INDUSTRIES USING NLP

Healthcare:
Some cases where NLP is used in the healthcare industry are clinical documentation improvement, clinical decision support, automated registry reporting and speech recognition.

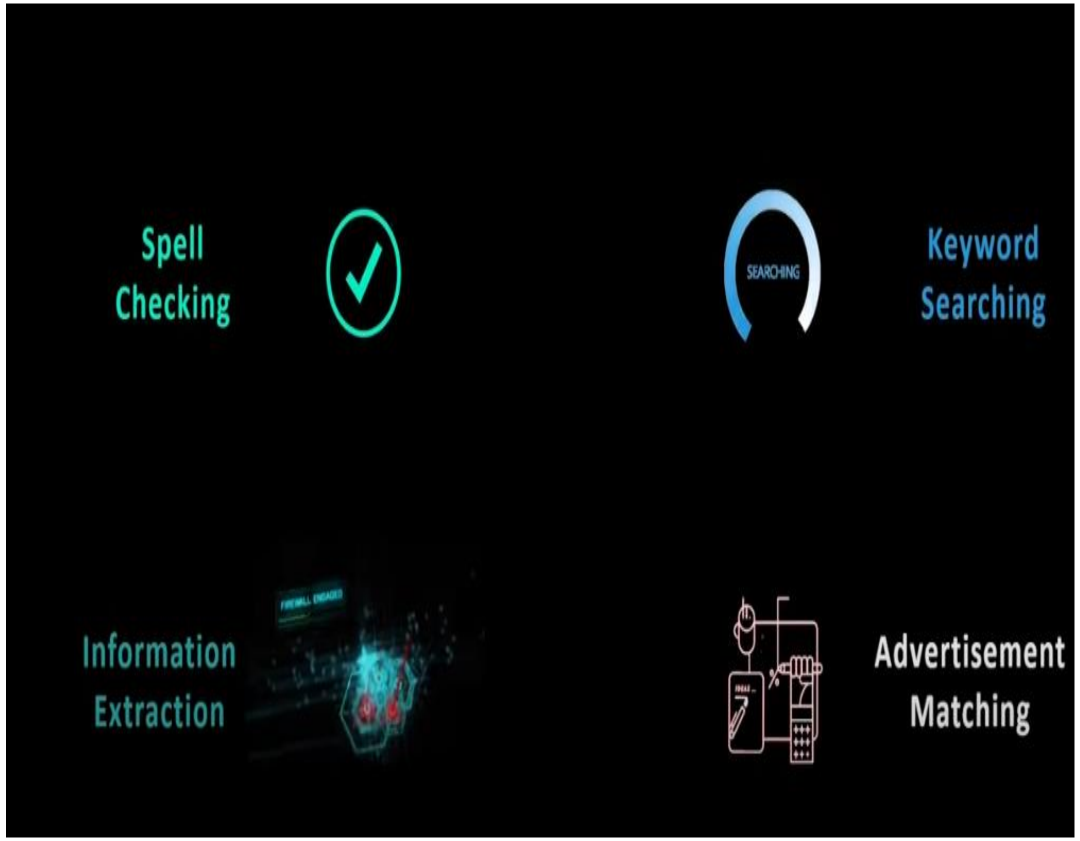
Marketing Intelligence:
NLP software helps marketers stay informed of what their competitors are up to and stay up-to-date on the latest trends by using tools to go through millions of blogs, websites etc.

Hiring and Recruitment:
HR professionals can now considerably speed up candidate search by filtering out relevant resumes and crafting bias-proof and gender-neutral job descriptions.

Advertising
By analysing digital footprint (i.e. social media, emails, search keywords, and browsing behaviour) NLP helps advertisers identify new audiences who could be potentially interested in their products.

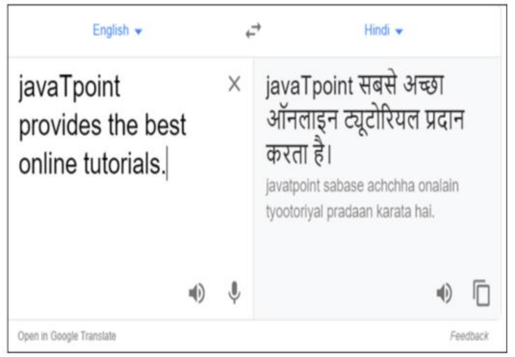
Applications of NLP

Sentimental Analysis
Chatbot
Speech Recognition
Machine Translation



Applications of NLP

- Twitter sentimental analysis or the Facebook sentiment as it's being used heavily now.
- Customer chat services provided by various companies and the process behind all of that is because of the NLP.
- Speech recognition are also talking about divorce assistants like Google assistant and alexa.
- NLP to translate data from one language to another
- Advertisement matching basically recommendation of ads based on your history.


English ▾ ⇄ Hindi ▾

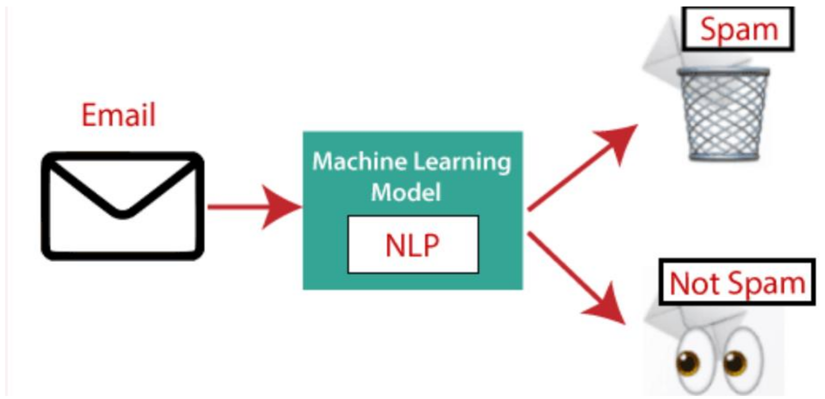
javaTpoint provides the best online tutorials.

javaTpoint सबसे अच्छा ऑनलाइन ट्यूटोरियल प्रदान करता है।

javatpoint sabase achchha onalain tyootoriyal pradaan karata hai.

Open in Google Translate

Feedback



Real life use cases of NLP

1. **Gmail** - when u r typing any sentence in gmail you will notice that it tries to auto complete. Auto completion is done using NLP.
2. **Spam filters** - This emails didn't have spam filters then you will be so much worried you will get so much mails/headache. Using NLP we can filter them and using keywords take them out of your inbox.
3. **Language translation** - Translate sentence from one language to another language.
4. **Customer service chat bot** - ex: in bank service chat bot if you type in a message and many times there is no human on the other hand. Your chat bot can interpret your language it can derive intent out of it and it can respond to your question on its own and sometimes it doesn't work well then they connect it to human beings.
5. **Voice assistants** at amazon, alexa and google assistant
6. **Google search** - BERT special language model will give a correct solution for all search question.

Advantages of NLP

- NLP helps users to ask questions about any subject and get a direct response within seconds.
- NLP offers exact answers to the question means it does not offer unnecessary and unwanted information.
- NLP helps computers to communicate with humans in their languages.
- It is very time efficient.
- Most of the companies use NLP to improve the efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.

Disadvantages of NLP

- NLP may not show context.
- NLP is unpredictable
- NLP may require more keystrokes.
- NLP is unable to adapt to the new domain, and it has a limited function that's why NLP is built for a single and specific task only.

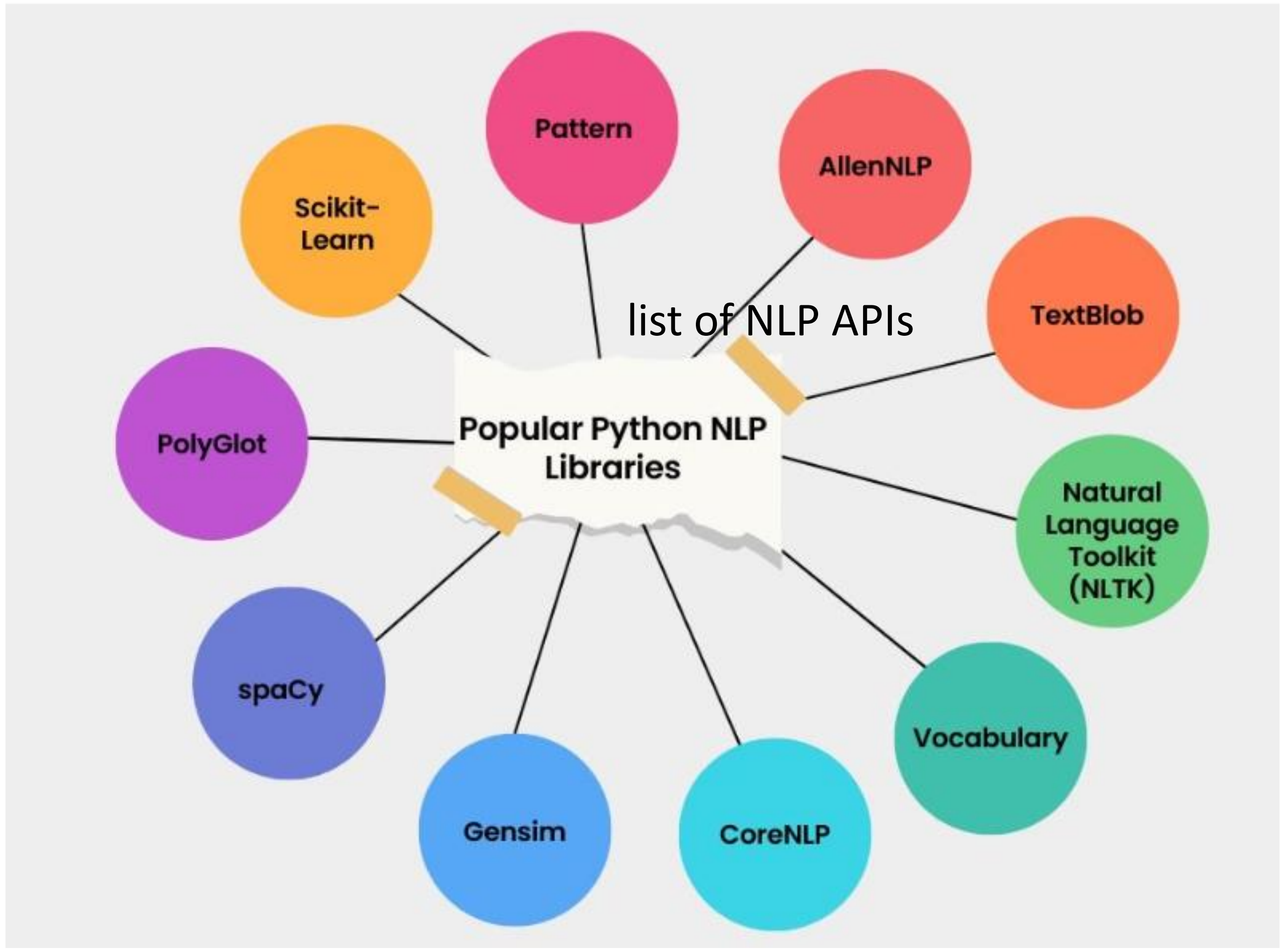
Components of NLP

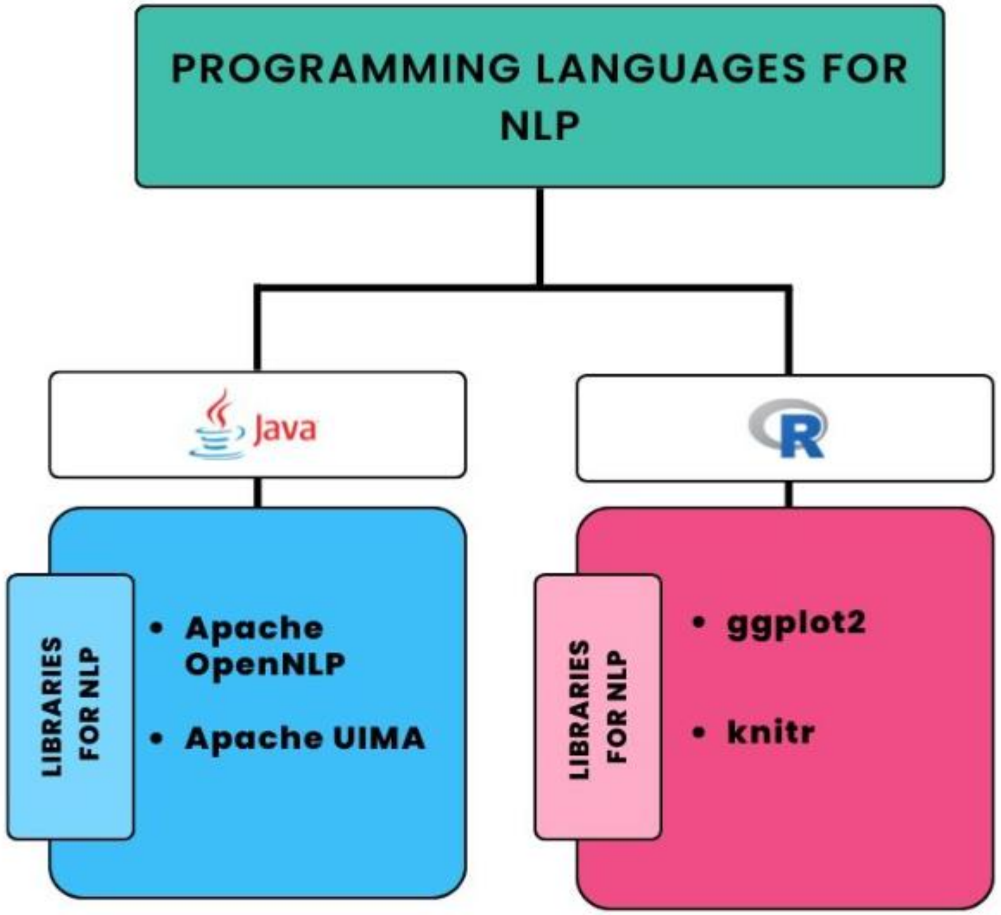
- **Natural Language Understanding (NLU)**
- Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.
- NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.
- NLU involves the following tasks -
- It is used to map the given input into useful representation.
- It is used to analyze different aspects of the language.

Components of NLP

- **2. Natural Language Generation (NLG)**
- Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation.
- It mainly involves Text planning, Sentence planning, and Text Realization.

Programming Languages for NLP





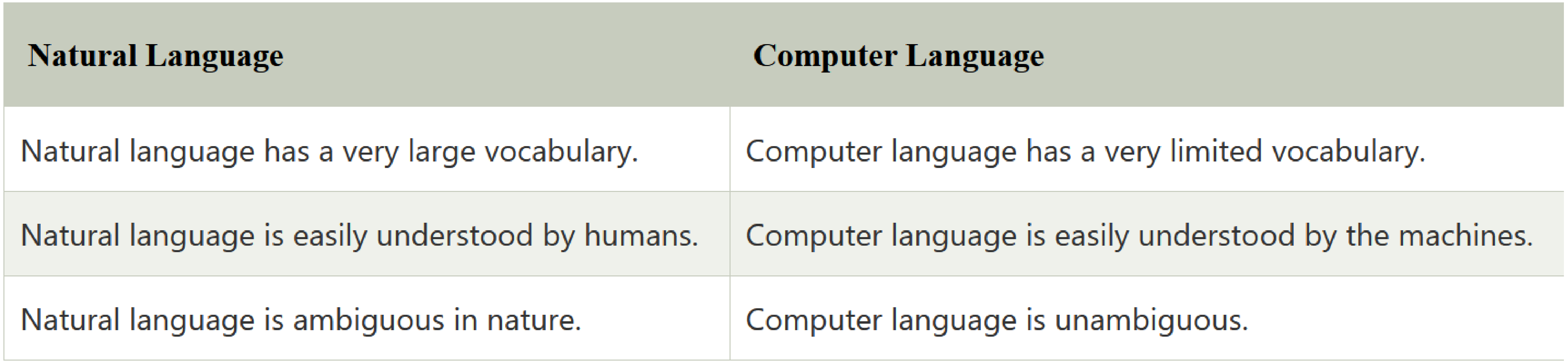
NLP Libraries

- **Scikit-learn:** It provides a wide range of algorithms for building machine learning models in Python.
- **Natural language Toolkit (NLTK):** NLTK is a complete toolkit for all NLP techniques.
- **Pattern:** It is a web mining module for NLP and machine learning.
- **TextBlob:** It provides an easy interface to learn basic NLP tasks like sentiment analysis, noun phrase extraction, or pos-tagging.
- **Quepy:** Quepy is used to transform natural language questions into queries in a database query language.
- **SpaCy:** SpaCy is an open-source NLP library which is used for Data Extraction, Data Analysis, Sentiment Analysis, and Text Summarization.
- **Gensim:** Gensim works with large datasets and processes data streams.

List of NLP APIs

- **IBM Watson API**
  IBM Watson API combines different sophisticated machine learning techniques to enable developers to classify text into various custom categories.
- **Chatbot API**
  Chatbot API allows you to create intelligent chatbots for any service.
- **Speech to text API**
  Speech to text API is used to convert speech to text
- **Text Analysis API by AYLIEN**
  Text Analysis API by AYLIEN is used to derive meaning and insights from the textual content.
- **Cloud NLP API**
  The Cloud NLP API is used to improve the capabilities of the application using natural language processing technology.

Difference between Natural language and Computer Language



Natural Language	Computer Language
Natural language has a very large vocabulary.	Computer language has a very limited vocabulary.
Natural language is easily understood by humans.	Computer language is easily understood by the machines.
Natural language is ambiguous in nature.	Computer language is unambiguous.

Student Evaluation MCQ

1)What is the field of Natural Language Processing (NLP)?
a) Computer Science b) Artificial Intelligence
c) Linguistics **d) All of the mentioned**

2)What is the main challenge/s of NLP?
a) Handling Ambiguity of Sentences b) Handling Tokenization
c) Handling POS-Tagging d) All of the mentioned

3) What is Machine Translation?
a) Converts one human language to another b) Converts human language to machine language
c) Converts any human language to English d) Converts Machine language to human language

4)Natural language processing is divided into the two subfields of -

A. symbolic and numeric B. algorithmic and heuristic

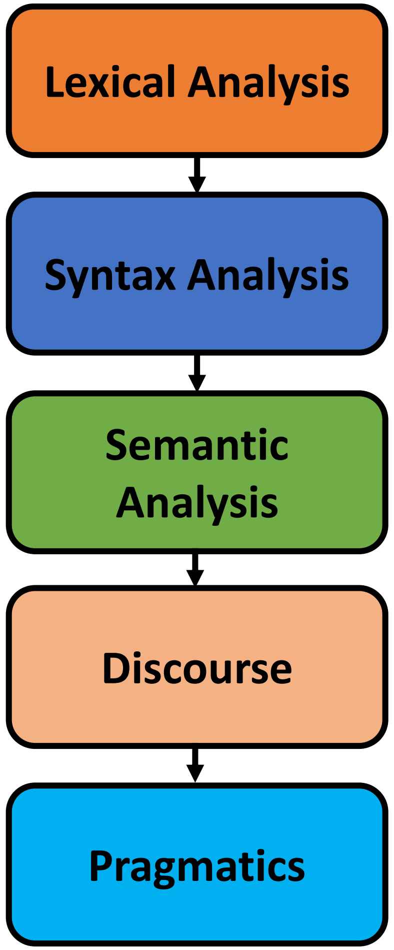
C. time and motion **D. understanding and generation**

5). The natural language is also known as ....

A. 3rd Generation language B. 4th Generation language

C. 5th Generation language D. 6th Generation language

Process of NLP



1.Morphological Analysis/ Lexical Analysis

2.Syntax Analysis

3.Semantic Analysis

4.Discourse

5.Pragmatics

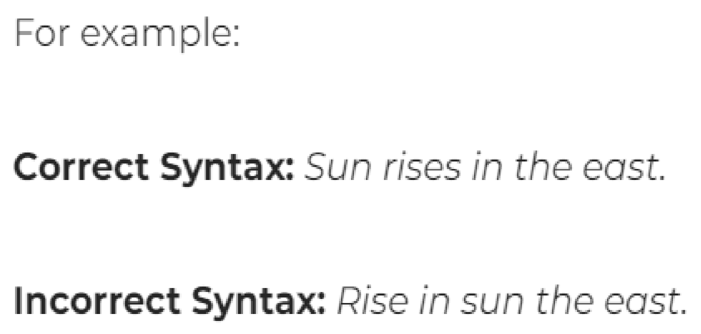
Morphological Analysis/ Lexical Analysis

- Morphological or Lexical Analysis deals with text at the individual word level.
- It looks for morphemes, the smallest unit of a word.
- The first phase of NLP is the Lexical Analysis.
- This phase scans the source code as a stream of characters and converts it into meaningful lexemes.
- It divides the whole text into paragraphs, sentences, and words.
- For example, irrationally can be broken into ir (prefix), rational (root) and -ly (suffix).
- Lexical Analysis finds the relation between these morphemes and converts the word into its root form.
- A lexical analyzer also assigns the possible Part-Of-Speech (POS) to the word.
- It takes into consideration the dictionary of the language.
- For example, the word “character” can be used as a noun or a verb.

Syntax Analysis

Syntax Analysis ensures that a given piece of text is **correct structure**. It tries to parse the sentence to check **correct grammar** at the sentence level.

Given the possible POS generated from the previous step, a syntax analyzer assigns POS tags based on the sentence structure.



For example:

Correct Syntax: *Sun rises in the east.*

Incorrect Syntax: *Rise in sun the east.*

Syntax

- Syntax refers to the arrangement of words in a very sentence specified they create grammatical sense.
- Syntax techniques

1. Lemmatization It entails reducing the various inflected forms of a word into a single form for easy analysis.
2. Morphological segmentation It involves dividing words into individual units called morphemes.
3. Word segmentation It involves dividing a large piece of continuous text into distinct units.
4. Part-of-speech tagging It involves identifying the part of speech for every word.
5. Parsing It involves undertaking a grammatical analysis for the provided sentence.
6. Sentence breaking It involves placing sentence boundaries on a large piece of text.
7. Stemming It involves cutting the inflected words to their root form.

Semantics

- Semantics refers to the meaning that is conveyed by a text. It involves the interpretation of words and how sentences are structured.

1. Named entity recognition (NER) It involves determining the parts of a text that can be identified and categorized into preset groups. Examples of such groups include names of individuals and names of places.
2. Word sense disambiguation It involves giving meaning to a word based on the context.

Semantic Analysis

- Consider the sentence: “The apple ate a banana”. Although the sentence is syntactically correct, it doesn’t make sense because apples can’t eat.
- Semantic analysis looks for meaning in the given sentence. It also deals with combining words into phrases.
- For example, “red apple” provides information regarding one object; hence we treat it as a single phrase.
- Similarly, we can group names referring to the same category, person, object or organization. “Robert Hill” refers to the same person and not two separate names – “Robert” and “Hill”

Discourse

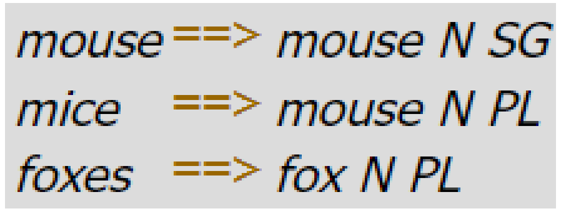
- Discourse deals with the effect of a previous sentence on the sentence in consideration. In the text, “Jack is a bright student. He spends most of the time in the library.” Here, discourse assigns “he” to refer to “Jack”.

Pragmatics

- The final stage of NLP, Pragmatics interprets the given text using information from the previous steps. Given a sentence, “Turn off the lights” is an order or request to switch off the lights.

Morphological Analysis (Morphological Parsing)

- The goal of morphological parsing is to find out what morphemes a given word is built from.
- The goal of morphological parsing is to find out what morphemes a given word is built from. For example, a morphological parser should be able to tell us that the word cats is the plural form of the noun stem cat, and that the word mice is the plural form of the noun stem mouse. So, given the string cats as input, a morphological parser should produce an output that looks similar to cat N PL. Here are some more examples:



mouse	==>	*mouse N SG*
mice	==>	*mouse N PL*
foxes	==>	*fox N PL*

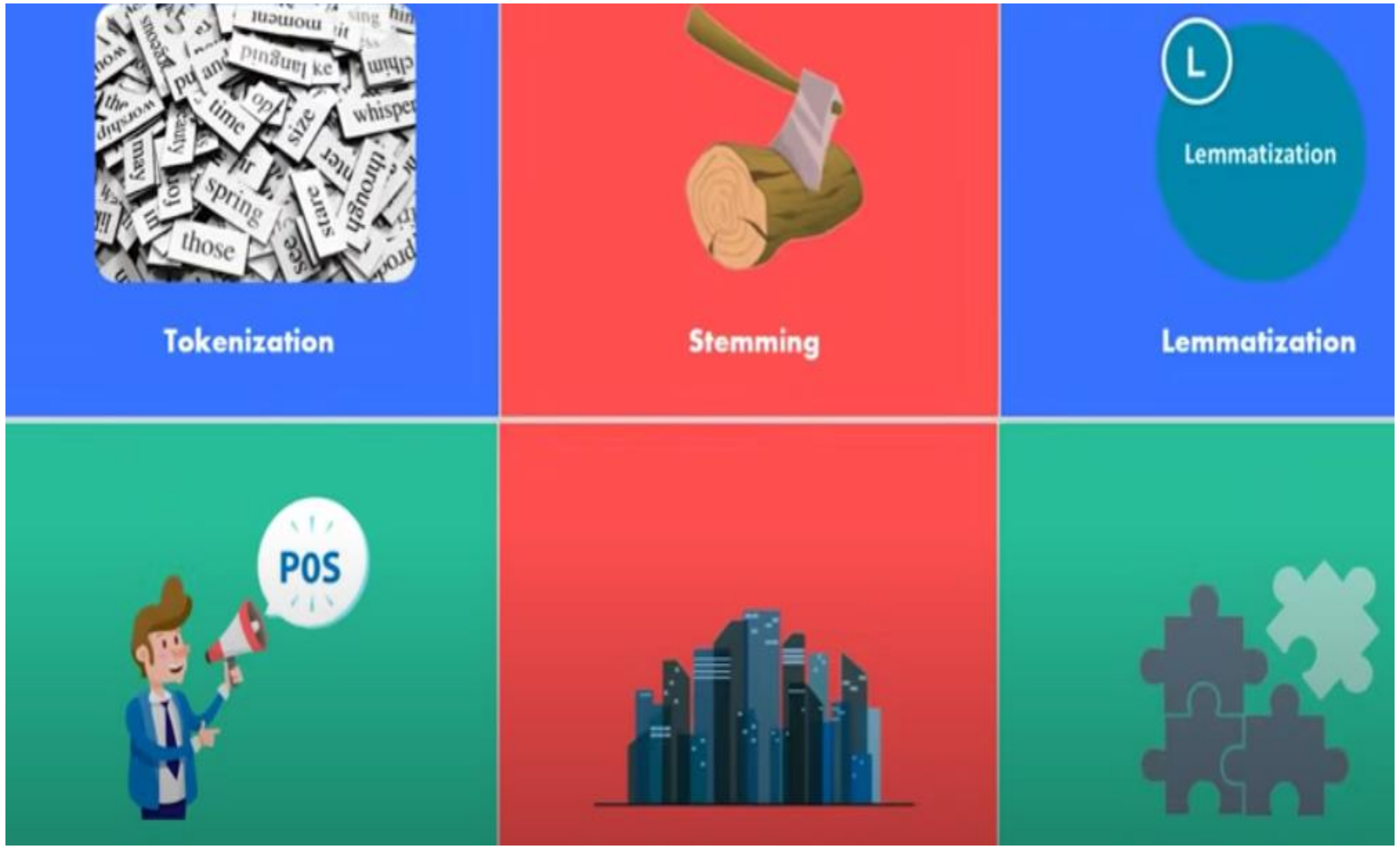
Morphological Analysis (Morphological Parsing)

- Morphological parsing yields information that is useful in many NLP applications. In parsing, e.g., it helps to know the agreement features of words. Similarly, grammar checkers need to know agreement information to detect such mistakes.

- But morphological information also helps spell checkers to decide whether something is a possible word or not, and in information retrieval it is used to search not only cats, if that's the user's input, but also for cat.

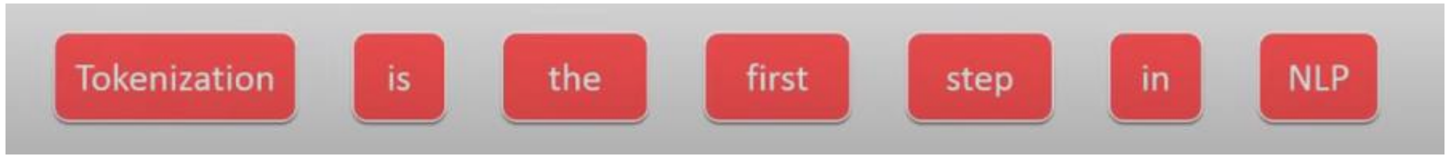
Pipeline of NLP in AI /Steps in NLP

1. Tokenization
2. Stemming
3. Lemmatization
4. POS tags
5. Named Entity Recognition
6. Chunking



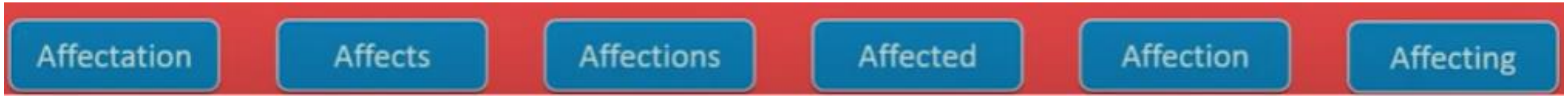
Tokenization

- Process the strings into tokens. Tokens is like a small structures or units.



Stemming

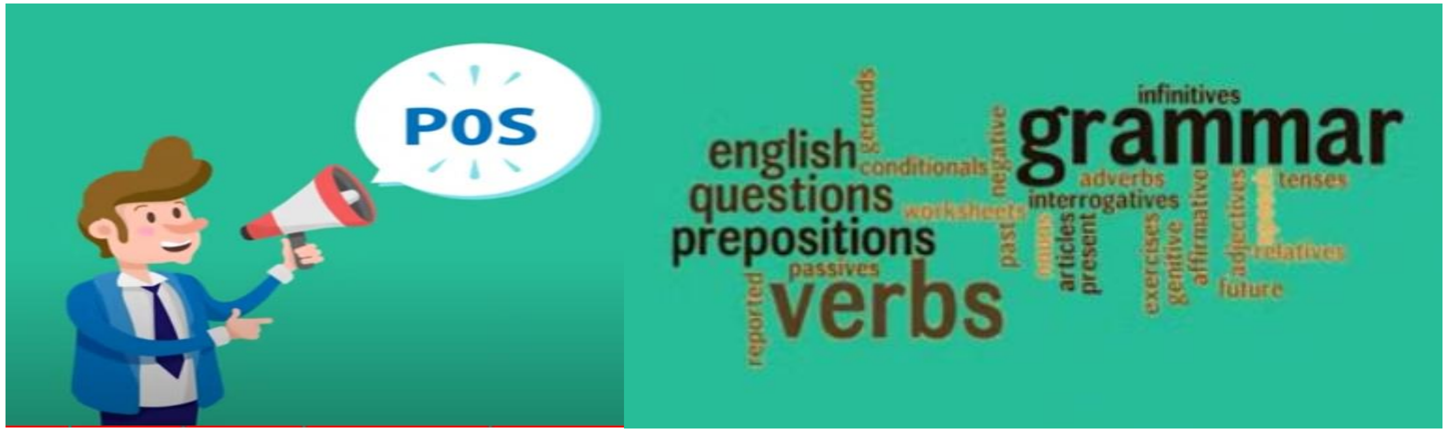
- Normalize words into its base or root form.
- stemming algorithm works by cutting off the end or the beginning of the word taking into account a list of common prefixes suffixes that can be found in an infected word.
- Here the **single root word is affect**.



Lemmatization

- Morphological analysis of the word to do so its necessary to have a detail dictionary and the algorithm can look through to link the original word or root word.
- Somehow similar to stemming, as it maps several words into one common root.
- Output of lemmatization is **proper word**.
- For example, Lemmatizer should map **gone,going and went into go**.

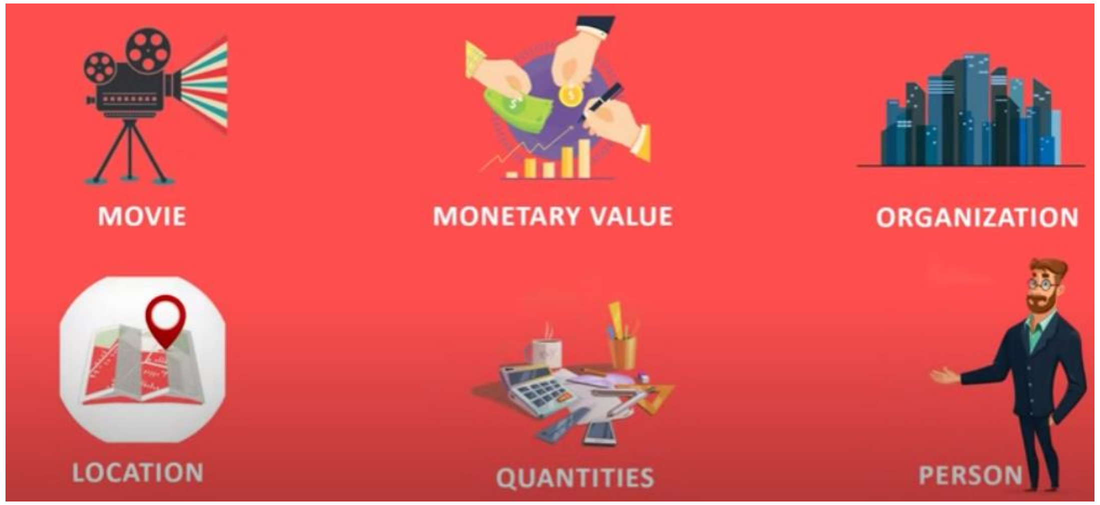
POS tags

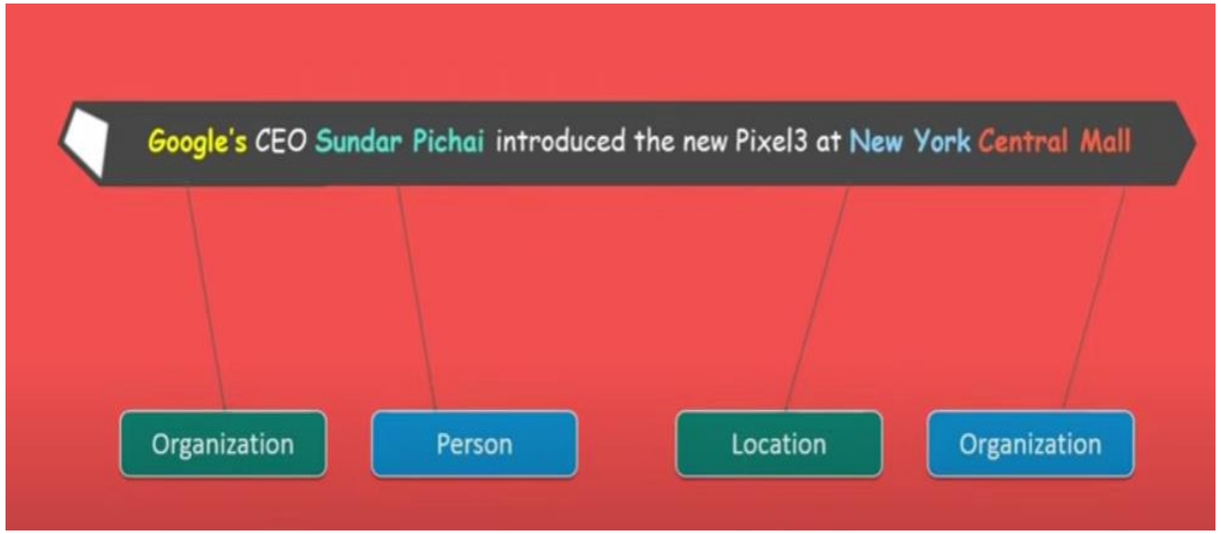


- Speaking the grammatical type of the word is referred to as POS tags or parts of speech.
- Nouns, pronouns, verbs, adverbs, adjectives, prepositions, Conjunctions and Interjections.

Named Entity Recognition

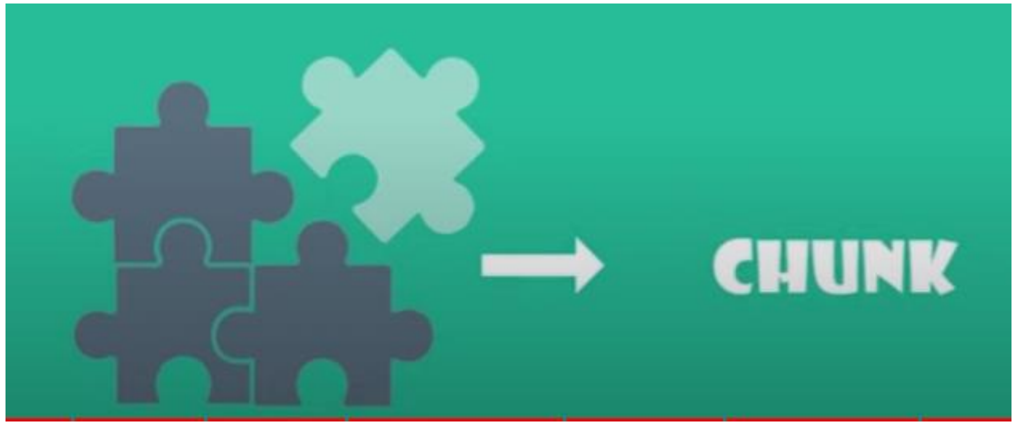
- Process of detecting the named entities such as the person name, company name and location that is pharse identification.

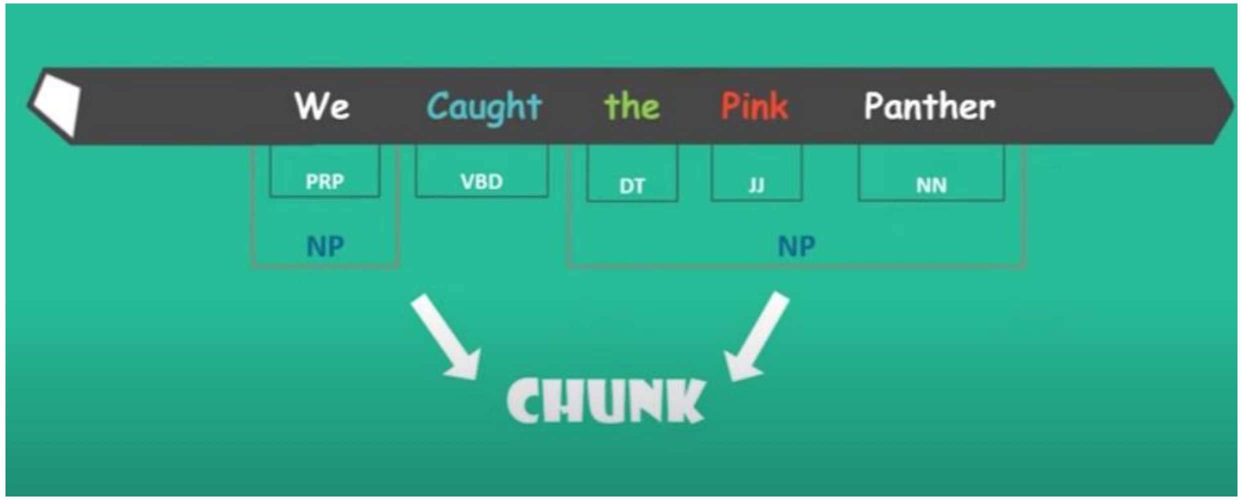



Google's CEO Sundar Pichai introduced the new Pixel3 at New York Central Mall
Organization
Person
Location
Organization

Chunking

- Picking up individual pieces of information and grouping them into bigger pieces.
- Grouping of words or tokens.





Why is NLP hard?

- ambiguity and variability of linguistic expression:

variability:	many forms can mean the same thing
ambiguity:	one form can mean many things

- Many different kinds of ambiguity
- Each NLP task has to address a distinct set of kinds

Steps – Morphology – Syntax – Semantics

- Language - smallest individual unit
- Phoneme - Single distinguishable sound
- Morphology - word formation

Morphology

- Morphology helps linguists understand the structure of words by putting together morphemes.
- A morpheme is the smallest grammatical, meaningful part of language.
- 2 types - Free morpheme and Bound morpheme.
- A **free morpheme** is a single meaningful unit of a word that can stand alone in the language. For example: cat, mat, trust, slow.
- A **bound morpheme** cannot stand alone, it has no real meaning if it is on its own. For example: walked, **(ed)** can not stand alone or unpleasant **(un)** is not a stand alone morpheme.Bound morphemes that are part of **prefixes** and **suffixes**.
- Bound morphemes can also be grouped into into a further two categories.

  1. Derivational 2. Inflectional

Derivational

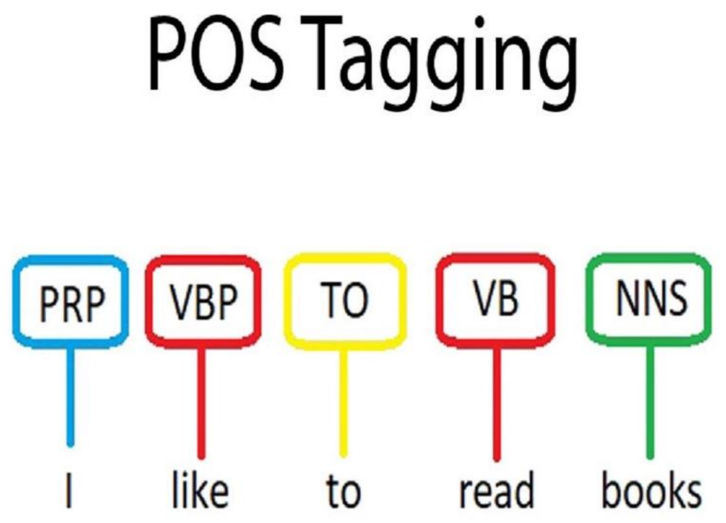
- Added to the base form of the word to create a new word.
- Look at the word **able** and let it become **ability**. In this instance the **adjective** becomes a **noun**.
- The word **send** as a **verb morpheme** becomes **sender** and a **noun with the addition of er**.
- While **stable** to **unstable** changes the meaning of the word to become the opposite meaning.
- In other words the meaning of the word is completely changed by adding a derivational morpheme to a base word.

Inflectional

- Additions to the base word that do not change the word, but rather serve as grammatical indicators. They show grammatical information. For example:

1. **Laugh** becomes the past tense by adding ed and changing the word to **laughed**.
2. **Dog to dogs** changes the word from **singular to plural**.
3. **Swim to swimming** changes the **verb into a progressive verb**.
4. All these examples show how morphology participates in the study of linguistics.

Parts of Speech Tagging



- Introduce the task of part-of-speech tagging, taking a sequence of words and assigning each word a part of speech like NOUN or VERB
- Task of named entity recognition (NER), assigning words or phrases tags like PERSON, LOCATION, or ORGANIZATION.

Introductions- Part-of-Speech Tagging

- Part-of-speech tagging is the process of assigning a part-of-speech to each word in text.
- The input is a sequence x1, x2, ..., xn of (tokenized) words and a tagset, and
- the output is a sequence y1, y2, ..., yn of tags,
- each output yi corresponding exactly to one input xi,

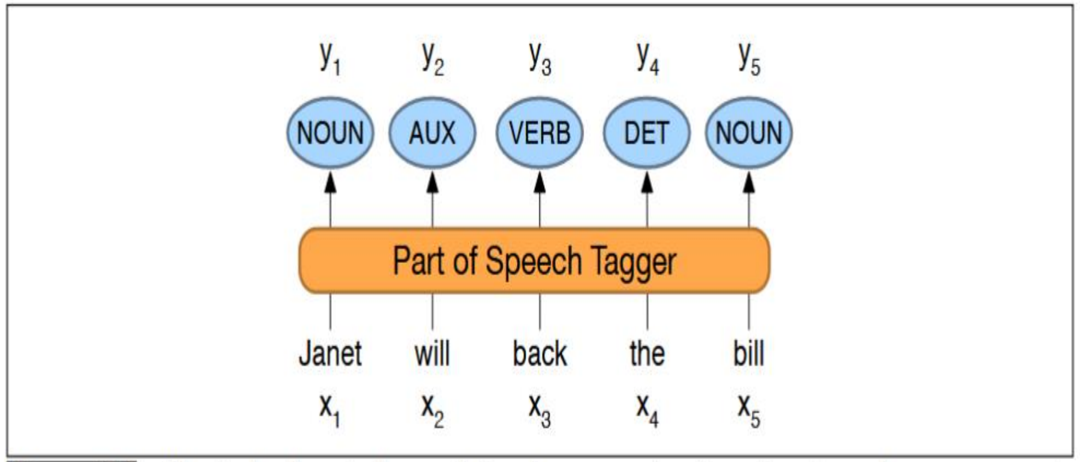


Figure 8.3 The task of part-of-speech tagging: mapping from input words $x_1, x_2, \ldots, x_n$ to output POS tags $y_1, y_2, \ldots, y_n$.

Introduction

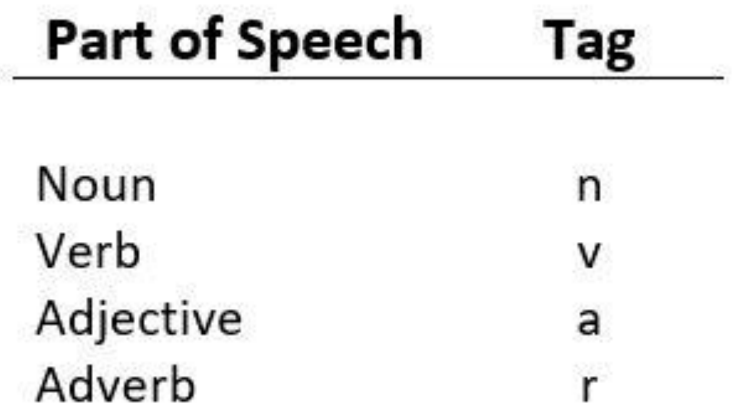
- Tagging is a **disambiguation** task; words are **ambiguous** —have more than one possible part-of-speech—and the goal is to find the correct tag for the situation.
- For example, book can be a verb (book that flight) or
- a noun (hand me that book).
- That can be a determiner (Does that flight serve dinner) or
- a complementizer (thought that your flight was earlier).
- The goal of POS-tagging is to resolve these ambiguity , choosing the proper tag for the context.

Introduction to POS Tagging

- Part-of-speech (POS) tagging is a process in natural language processing (NLP) where each word in a text is labeled with its corresponding part of speech.
- This can include nouns, verbs, adjectives, and other grammatical categories.
- POS tagging is useful for a variety of NLP tasks, such as **information extraction, named entity recognition, and machine translation.**
- It can also be used to identify the grammatical structure of a sentence and to disambiguate words that have multiple meanings.
- **example,**
- **Text:** “The cat sat on the mat.”
- **POS tags:**
- The: determiner , cat: noun , sat: verb , on: preposition , the: determiner ,mat: noun

What is Part-of-speech (POS) tagging ?

- It is a process of converting a sentence to forms – list of words, list of tuples (where each tuple is having a form *(word, tag)*).
- The tag in case of is a part-of-speech tag, and signifies whether the word is a noun, adjective, verb, and so on



Part of Speech	Tag
Noun	n
Verb	v
Adjective	a
Adverb	r

Part Of Speech	**Tag**
Noun (Singular)	NN
Noun (Plural)	NNS
Verb	VB
Determiner	DT
Adjective	JJ
Adverb	RB

Universal Part-of Speech Tagset

Universal Part-of-Speech Tagset

Tag	Meaning	English Examples
ADJ	adjective	new, good, high, special, big, local
ADP	adposition	on, of, at, with, by, into, under
ADV	adverb	really, already, still, early, now
CONJ	conjunction	and, or, but, if, while, although
DET	determiner, article	the, a, some, most, every, no, which
NOUN	noun	year, home, costs, time, Africa
NUM	numeral	twenty-four, fourth, 1991, 14:24
PRT	particle	at, on, out, over per, that, up, with
PRON	pronoun	he, their, her, its, my, I, us
VERB	verb	is, say, told, given, playing, would
.	punctuation marks	. , ; !
X	other	ersatz, esprit, dunno, gr8, univeristy

Use of Parts of Speech Tagging in NLP

1. **To understand the grammatical structure of a sentence:** By labeling each word with its POS, we can better understand the syntax and structure of a sentence.
2. **To disambiguate words with multiple meanings:** Some words, such as "bank," can have multiple meanings depending on the context in which they are used.
3. **To improve the accuracy of NLP tasks:** POS tagging can help improve the performance of various NLP tasks, such as named entity recognition and text classification.
4. **To facilitate research in linguistics:** POS tagging can also be used to study the patterns and characteristics of language use and to gain insights into the structure and function of different parts of speech.

Steps Involved in the POS tagging

- **Collect a dataset of annotated text:** This dataset will be used to train and test the POS tagger. The text should be annotated with the correct POS tags for each word.
- **Preprocess the text:** This may include tasks such as **tokenization (splitting the text into individual words), lowercasing, and removing punctuation.**
- **Divide the dataset into training and testing sets:** The training set will be used to train the POS tagger, and the testing set will be used to evaluate its performance.
- **Train the POS tagger:** This may involve building a statistical model, such as **a hidden Markov model (HMM), or defining a set of rules for a rule-based or transformation-based tagger.**
- The model or rules will be trained on the annotated text in the training set.
- **Test the POS tagger:** Use the trained model or rules to predict the POS tags of the words in the testing set. Compare the predicted tags to the true tags and calculate metrics such as precision and recall to evaluate the performance of the tagger.
- **Fine-tune the POS tagger:** If the performance of the tagger is not satisfactory, adjust the model or rules and repeat the training and testing process until the desired level of accuracy is achieved.
- **Use the POS tagger:** Once the tagger is trained and tested, it can be used to perform POS tagging on new, unseen text.

Application of POS Tagging

- **Information extraction:**
  - POS tagging can be used to identify specific types of information in a text, such as names, locations, and organizations.
  - T his is useful for tasks such as extracting data f rom news articles or building knowledge bases for artificial intelligence systems.
- **Named entity recognition**:
  - POS tagging can be used to identify and classify named entities in a text, such as people, places, and organizations.
  - This is useful f or tasks such as building customer profiles or identifying key figures in a news story.
- **Text classification:**
  - POS tagging can be used to help classify texts into different categories, such as spam emails or sentiment analysis.
  - By analyzing the POS tags of the words in a text, algorithms can better understand the content and tone of the text.

Application of POS Tagging

- **Machine translation:**
  - POS tagging can be used to help translate texts from one language to another by identifying the grammatical structure and relationships between words in the source language and mapping them to the target language.
- **Natural language generation:**
  - POS tagging can be used to generate natural-sounding text by selecting appropriate words and constructing grammatically correct sentences.
  - This is useful f or tasks such as chatbots and virtual assistants.

Different POS Tagging Techniques

1.Rule-Based POS Tag

- This is one of the **oldest** approaches to POS tagging.
- It involves using a **dictionary** consisting of all the possible POS tags for a given word.
- If any of the words have more than one tag, **hand-written rules** are used to assign the correct tag based on the tags of surrounding words.
- For example, if the preceding of a word an article, then the word has to be a noun.
- Consider the words: *A Book*
- **Get all the possible POS tags for individual words:** A – Article; Book – Noun or Verb
- **Use the rules to assign the correct POS tag:** As per the possible tags, "A" is an *Article* and we can assign it directly. But, a book can either be a Noun or a Verb. However, if we consider "A Book", A is an article and following our rule above, Book has to be a *Noun*. Thus, we assign the tag of Noun to book.
- **POS Tag:** [("A", "Article"), ("Book", "Noun")]

Rule-based POS Tagging

These rules may be either –

- **Context-pattern rules**
- **Regular expression** compiled into finite-state automata, intersected with lexically ambiguous sentence representation.

Rule-based POS tagging by its two-stage architecture –

- **First stage** – In the first stage, it uses a dictionary to assign each word a list of potential parts-of-speech.
- **Second stage** – In the second stage, it uses large lists of hand-written disambiguation rules to sort down the list to a single part-of-speech for each word.

Properties of Rule-Based POS Tagging

- These taggers are knowledge-driven taggers.
- The rules in Rule-based POS tagging are built manually.
- The information is coded in the form of rules.
- We have some limited number of rules approximately around 1000.
- Smoothing and language modeling is defined explicitly in rule-based taggers.

Rule-based POS tagger Example

1. Define a set of rules for assigning POS tags to words. For example:

- If the word ends in “-tion,” assign the tag “noun.”
- If the word ends in “-ment,” assign the tag “noun.”
- If the word is all uppercase, assign the tag “proper noun.”
- If the word is a verb ending in “-ing,” assign the tag “verb.”

2. Iterate through the words in the text and apply the rules to each word in turn. For example:

- “**Nation”** would be tagged as “noun” based on the first rule.
- “**Investment”** would be tagged as “noun” based on the second rule.
- “**UNITED”** would be tagged as “proper noun” based on the third rule.
- “**Running**” would be tagged as “verb” based on the fourth rule.

3. Output the POS tags for each word in the text.

Stochastic POS Tagging

Another technique of tagging is Stochastic POS Tagging.
The model that includes **frequency or probability** (statistics) can be called stochastic.
Any number of different approaches to the problem of part-of-speech tagging can be referred to as stochastic tagger.
The simplest stochastic tagger applies the following approaches for POS tagging −

Word Frequency Approach

- In this approach, the stochastic taggers disambiguate the words based on the probability that a word occurs with a particular tag.
- We can also say that the tag encountered most frequently with the word in the training set is the one assigned to an ambiguous instance of that word.
- The main issue with this approach is that it may yield inadmissible sequence of tags.

Tag Sequence Probabilities

- It is another approach of stochastic tagging, where the tagger calculates the probability of a given sequence of tags occurring.
- It is also called n-gram approach.
- It is called so because the best tag for a given word is determined by the probability at which it occurs with the n previous tags.

Statistical POS Tagging

- **Statistical part-of-speech (POS)** tagging is a method of labeling words with their corresponding parts of speech using statistical techniques.
- This is in contrast to rule-based POS tagging, which relies on pre-defined rules, and to unsupervised learning-based POS tagging, which does not use any annotated training data.
- In statistical POS tagging, a model is trained on a large annotated corpus of text to learn the patterns and characteristics of different parts of speech.
- The model uses this training data to predict the POS tag of a given word based on the context in which it appears and the probability of different POS tags occurring in that context.
- Statistical POS taggers can be more accurate and efficient than rule-based taggers, especially for tasks with large or complex datasets.

Transformation-based tagging (TBT)

- **Transformation-based tagging (TBT)** is a method of part-of-speech (POS) tagging that uses a series of rules to transform the tags of words in a text.
- This is in contrast to rule-based POS tagging, which assigns tags to words based on pre-defined rules, and to statistical POS tagging, which relies on a **trained model to predict tags based on probability.**

Working Principles

- Here is an example of how a TBT system might work:

1. Define a set of rules for transforming the tags of words in the text. For example:

- If the word is a verb and appears after a determiner, change the tag to “noun.”
- If the word is a noun and appears after an adjective, change the tag to “adjective.”

2. Iterate through the words in the text and apply the rules in a specific order. For example:

- In the sentence “The cat sat on the mat,” the word “sat” would be changed from a verb to a noun based on the first rule.
- In the sentence “The red cat sat on the mat,” the word “red” would be changed from an adjective to a noun based on the second rule.

3. Output the transformed tags for each word in the text.

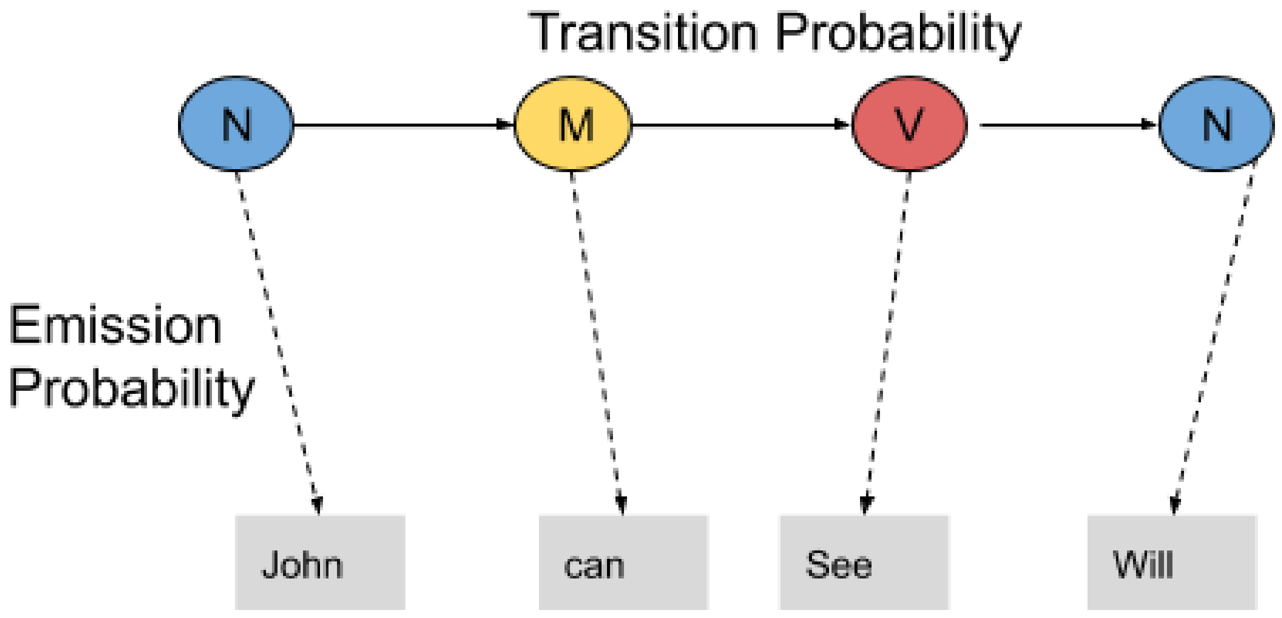
Hidden Markov Model POS tagging

- Hidden Markov models (HMMs) are a type of statistical model that can be used for part-of-speech (POS) tagging in natural language processing (NLP).
- In an HMM-based POS tagger, a model is trained on a large annotated corpus of text to learn the patterns and characteristics of different parts of speech.
- The model uses this training data to predict the POS tag of a given word based on the probability of different tags occurring in the context of the word.

HMM (Hidden Markov Model)

- HMM (Hidden Markov Model) is a Stochastic technique for POS tagging.
- Hidden Markov models are known for their applications to reinforcement learning and temporal pattern recognition
- such as
  - Speech,
  - Handwriting,
  - Gesture recognition,
  - Musical score following,
  - Partial discharges, and
  - Bioinformatics.

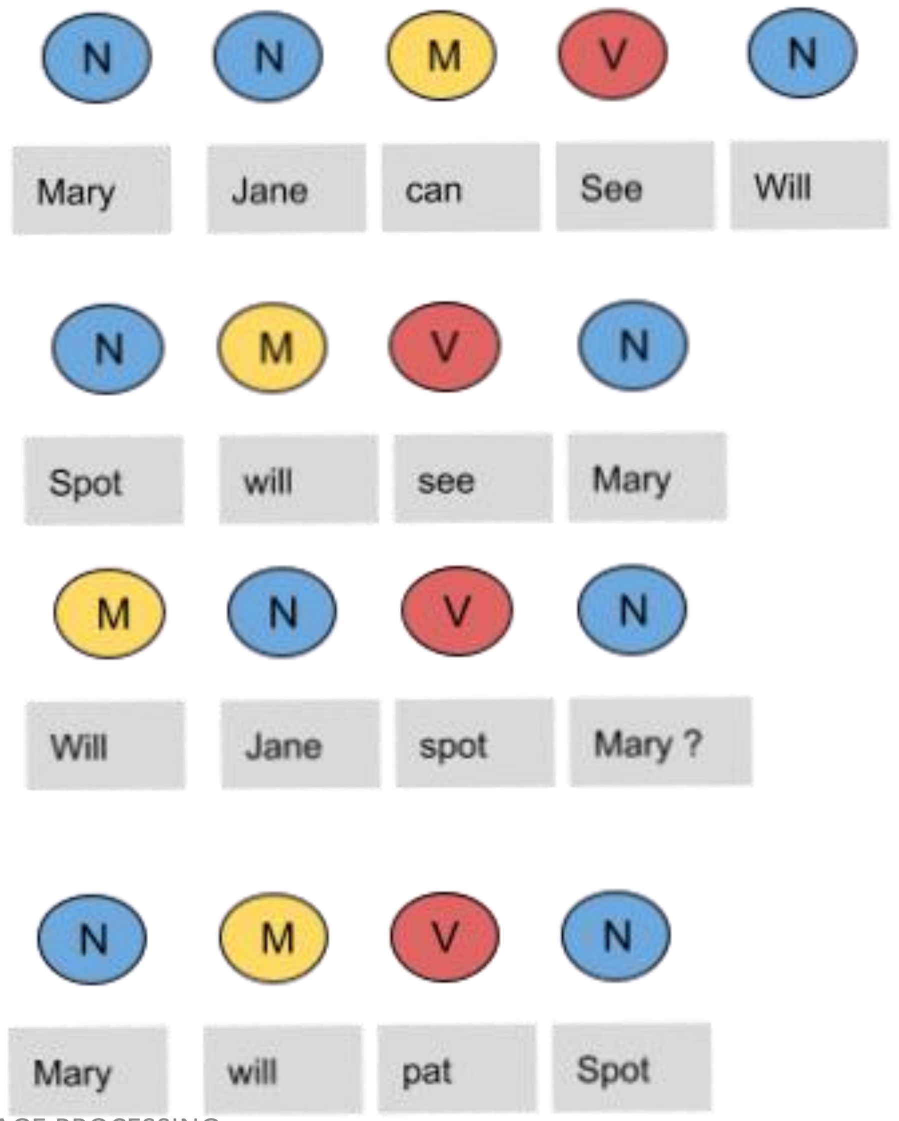
Example



The transition probability is the likelihood of a particular sequence for example, how likely is that a noun is followed by a model and a model by a verb and a verb by a noun.
This probability is known as Transition probability.
It should be high for a particular sequence to be correct.

Example

- Mary Jane can see Will
- Spot will see Mary
- Will Jane spot Mary?
- Mary will pat Spot



Counting Table – Emission Probability

Words	Noun	Model	Verb
Mary	4	0	0
Jane	2	0	0
Will	1	3	0
Spot	2	0	1
Can	0	1	0
See	0	0	2
pat	0	0	1

Words	Noun	Model	Verb
Mary	4/9	0	0
Jane	2/9	0	0
Will	1/9	3/4	0
Spot	2/9	0	1/4
Can	0	1/4	0
See	0	0	2/4
pat	0	0	1

Emission probabilities

- The probability that Mary is Noun = 4/9
- The probability that Mary is Model = 0
- The probability that Will is Noun = 1/9
- The probability that Will is Model = 3/4

Challenges in POS Tagging

- **Ambiguity:** Some words can have multiple POS tags depending on the context in which they appear, making it difficult to determine their correct tag. For example, the word “bass” can be a noun (a type of fish) or an adjective (having a low frequency or pitch).

- **Out-of-vocabulary (OOV) words:** Words that are not present in the training data of a POS tagger can be difficult to tag accurately, especially if they are rare or specific to a particular domain.

- **Complex grammatical structures:** Languages with complex grammatical structures, such as languages with many inflections or free word order, can be more challenging to tag accurately.

- **Lack of annotated training data:** Some languages or domains may have limited annotated training data, making it difficult to train a high-performing POS tagger.

- **Inconsistencies in annotated data:** Annotated data can sometimes contain errors or inconsistencies, which can negatively impact the performance of a POS tagger.

STUDENT EVALUATION

1) What is the main challenges of NLP?

A. Handling Tokenization B. Handling POS-Tagging

C. Handling Ambiguity of Sentences D. None of the above

2) All of the following are challenges associated with natural language processing except

A. **dividing up a text into individual words in English.** B. understanding the context in which something is said.

C. recognizing typographical or grammatical errors in texts D. distinguishing between words that have more than one meaning

3) In linguistic morphology, _____________ is the process for reducing inflected words to their root form.

A. Stemming B. Rooting

C. Text-Proofing D.Both A and B

4) Morphological Segmentation

A. Is an extension of propositional logic

B. Does Discourse Analysis

C. **Separate words into individual morphemes and identify the class of the morphemes**

D. None of the mentioned

Language modelling in NLP

- Language modeling is a fundamental task in Natural Language Processing (NLP) that involves building a statistical model to predict the probability distribution of words in a given language.
- The language model learns the patterns and relationships between words in a corpus of text
- Can be used to generate new text, evaluate the likelihood of a sentence, perform **speech recognition, machine translation, Spam filtering**, etc.
- In language modeling, the primary goal is to estimate the probability of a sequence of words (a sentence or a phrase) using the conditional probability of each word given its preceding context.
- The model learns from large amounts of text data to predict the likelihood of a particular word given the previous words in a sentence.

Language modelling in NLP

There are different types of language models, but two prominent approaches are

- **N-gram Language Models**
- **Neural Language Models**

N-gram Language Models: N-gram language models are simple and widely used in early NLP tasks. An N-gram model predicts the probability of a word based on the previous (N-1) words in a sentence.

For example, a trigram model (3-gram) predicts the probability of a word given the two preceding words. The model estimates the probabilities based on the frequency of word sequences observed in the training data.

Neural Language Models: Neural language models, such as recurrent neural networks (RNNs) and transformer-based models, have gained significant popularity in recent years due to their ability to capture long-range dependencies and contextual information

These models learn complex patterns in the language and can generate more coherent and contextually relevant text.

Language modelling in NLP

- **Recurrent Neural Networks (RNNs)**: RNNs are a class of neural networks designed for sequential data processing.
- They process input sequences step by step, maintaining a hidden state that captures information from previous steps.
- This hidden state acts as the context for the current word prediction.
- However, RNNs have challenges with capturing long-range dependencies and can suffer from vanishing or exploding gradients.
- **Transformer-based Models**: Transformer-based models, like the famous BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) series, have revolutionized language modeling.

What Are N-Grams?

- N-grams are continuous sequences of words or symbols, or tokens in a document.
- In technical terms, they can be defined as the neighboring sequences of items in a document.
- They come into play when we deal with text data in NLP (Natural Language Processing) tasks.
- An N-gram means a sequence of N words. So for example, “Medium blog” is a 2-gram (a bigram), “A Medium blog post” is a 4-gram, and “Write on Medium” is a 3-gram (trigram).
- They have a wide range of applications, like language models,
  - Semantic features,
  - Spelling correction,
  - Machine translation,
  - Text mining, etc.

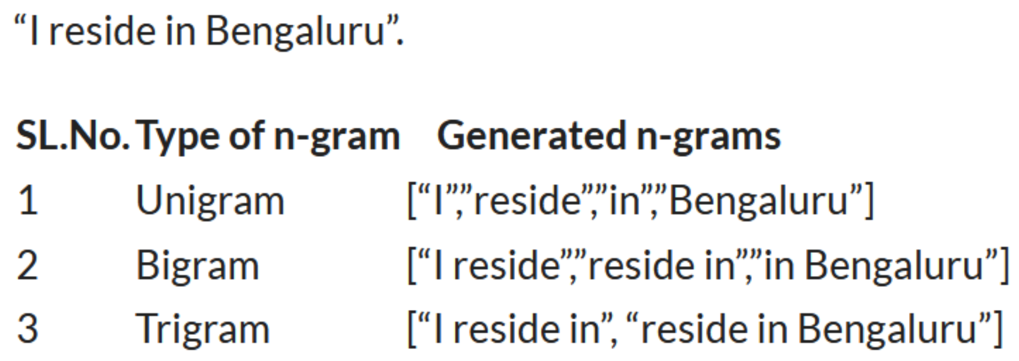
N-gram Language Model:

- N-gram can be defined as the contiguous sequence of n items from a given sample of text or speech.
- The items can be letters, words, or base pairs according to the application.
- The N-grams typically are collected from a text or speech corpus (A long text dataset).
- An N-gram language model predicts the probability of a given N-gram within any sequence of words in the language.
- A good N-gram model can predict the next word in the sentence i.e the value of p(w|h)

N-Gram Types

- Example of N-gram such as unigram (“This”, “article”, “is”, “on”, “NLP”) or bi-gram (‘This article’, ‘article is’, ‘is on’,’on NLP’).
- Now, we will establish a relation on how to find the next word in the sentence using
- We need to calculate p(w|h), where is the candidate for the next word.
- Types
  - Unigram
  - Bi gram
  - Trigram
  - N-Gram

Example of N-Grams



“I reside in Bengaluru”.

SL.No.	Type of n-gram	Generated n-grams
1	Unigram	[“I”,”reside”,”in”,”Bengaluru”]
2	Bigram	[“I reside”,”reside in”,”in Bengaluru”]
3	Trigram	[“I reside in”, “reside in Bengaluru”]

$p(NLP|this\, article\, is\, on)$

After generalizing the above equation can be calculated as:

$p(w_5|w_1, w_2, w_3, w_4)\; or\; P(W)$

$= p(w_n|w_1, w_2 ... w_n)$

But how do we calculate it? The answer lies in the chain rule of probability:

$$P(A|B) = \frac{P(A,B)}{P(B)}$$
$$P(A,B) = P(A|B)P(B)$$

Now generalize the above equation:

$$P(X_1, X_2, ..., X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)....P(X_n|X_1, X_2, ...X_n)$$
$$P(w_1 w_2 w_3 ... w_n) = \prod_i P(w_i|w_1 w_2 ... w_n)$$

Simplifying the above formula using Markov assumptions:

$$P(w_i|w_1, w_2, ...w_{i-1}) \approx P(w_i|w_{i-k}, ...w_{i-1})$$

- For unigram:

$$P(w_1 w_2, \ldots w_n) \approx \prod_i P(w_i)$$

- For Bigram:

$$P(w_i | w_1 w_2, .. w_{i-1}) \approx P(w_i | w_{i-1})$$

N-GRAM QUESTION

- The following sentences as the training corpus:
  - Thank you so much for your help.
  - I really appreciate your help.
  - Excuse me, do you know what time it is?
  - I’m really sorry for not inviting you.
  - I really like your watch.
- Suppose we’re calculating the probability of word “w1” occurring after the word “w2,” then the
- formula for this is as follows:
- count(w2 w1) / count(w2)
- which is the number of times the words occurs in the required sequence, divided by the number
- of the times the word before the expected word occurs in the corpus.

ANSWER

From our example sentences, let’s calculate the probability of the word “like” occurring after the

word “really”:

count(really like) / count(really)

= 1 / 3

= 0.33

Similarly, for the other two possibilities:

count(really appreciate) / count(really)

= 1 / 3

= 0.33

count(really sorry) / count(really)

= 1 / 3

= 0.33

P(w|h), the probability of a word w given some history h.

Metrics for Language Modelings

- **Entropy**: Entropy, as a measure of the amount of information conveyed by Claude Shannon. Below is the formula for representing entropy

$$H(p) = \sum_x p(x) \cdot (-log(p(x)))$$

H(p) is always greater than equal to 0.

- **Cross-Entropy**: It measures the ability of the trained model to represent test data(W_1^{i-1}).

$$H(p) = \sum_{i=1}^{x} \frac{1}{n}(-log_2(p(w_i|w_1^{i-1})))$$

The cross-entropy is always greater than or equal to Entropy i.e the model uncertainty can be no less than the true uncertainty.

Metrics for Language Modelings

- **Perplexity**: Perplexity is a measure of how good a probability distribution predicts a sample. It can be understood as a measure of uncertainty. The perplexity can be calculated by cross-entropy to the exponent of 2.

$$2^{Cross-Entropy}$$

Following is the formula for the calculation of Probability of the test set assigned by the language model, normalized by the number of words:

$$PP(W) = \sqrt[n]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

- **For Example:**
- Let’s take an example of the sentence: ***‘Natural Language Processing’.*** For predicting the first word, let’s say the word has the following probabilities:

word	**P(word | <start>)**
The	0.4
Processing	0.3
Natural	0.12
Language	0.18

- Now, we know the probability of getting the first word as natural. But, what’s the probability of getting the next word after getting the word ‘*Language*‘ after the word ‘*Natural*‘.

word	**P(word | ‘Natural’)**
The	0.05
Processing	0.3
Natural	0.15
Language	0.5

- After getting the probability of generating words 'Natural Language', what's the probability of getting '*Processing*'.

word	P(word | 'Language')
The	0.1
Processing	0.7
Natural	0.1
Language	0.1

- Now, the perplexity can be calculated as:

$$PP(W) = \sqrt[n]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}} = \sqrt[3]{\frac{1}{0.12*0.5*0.7}} \approx 2.876$$

- From that we can also calculate entropy:

$$Entropy = log_2(2.876) = 1.524$$

What are MWEs?

- Multiword expressions (MWEs) are expressions which are made up of at least 2 words and which can be syntactically and/or semantically idiosyncratic in nature.
- Sequence of words that has lexical, orthographic, phonological, morphological, syntactic, semantic, pragmatic or translational properties not predictable from the individual components or their normal mode of combination.
- Multi-word Expressions (MWEs) are word combinations with linguistic properties that cannot be
- predicted from the properties of the individual words or the way they have been combined.
- MWEs occur frequently and are usually highly domain-dependent
- A proper treatment of MWEs is essential for the success of NLP-systems.

MWE

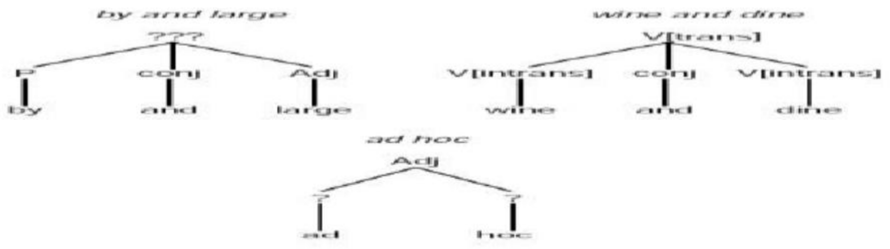
- A sequence, continuous or discontinuous, of words or other elements, which is or appears to be

  prefabricated: that is stored and retrieved whole from memory at the time from use, rather than being subject to generation or analysis by language grammar.
- A language word - lexical unit in the language that stands for a concept.
- e.g. train, water, ability
- However, that may not be true.
- e.g. Prime Minister
- Due to institutionalized usage, we tend to think of ‘Prime Minister’ as a single concept.
- • Here the concept crosses word boundaries.
- Simply put, a multiword expression (MWE):
- a. crosses word boundaries
- b. is lexically, syntactically, semantically, pragmatically and/or statistically idiosyncratic

E.g. traffic signal, Real Madrid, green card, fall asleep, leave a mark, ate up, figured out, kick the bucket, spill the beans, ad hoc.

Idiosyncrasies

- **Statistical idiosyncracies**
- Usage of the multiword has been conventionalized, though it is still semantically decomposable E.g. traffic signal, good morning
- **Lexical idiosyncrasies**
- Lexical items generally not seen in the language, probably borrowed from other languages
- E.g. ad hoc, ad hominem
- **Syntactic idiosyncrasy**
- Conventional grammar rules don’t hold, these multiword exhibit peculiar syntactic behaviour



Idiosyncrasies

Semantic Idiosyncrasy

- The meaning of the multi word is not completely composable from those of its constituents .
- This arises from figurative or metaphorical usage .
- The degree of compositionality varies

E.g. blow hot and cold – keep changing opinions
spill the beans – reveal secret
run for office – contest for an official post.

MWE Characteristics

- **Basis for MWE extraction**

o **Non-Compositionality**
  - Non-decomposable – e.g. blow hot and cold
  - Partially decomposable – e.g. spill the beans

o **Syntactic Flexibility**
  - Can undergo inflections, insertions, passivizations
  - e.g. promise(d/s) him the moon
  - The more non-compositional the phrase, the less syntactically flexible it is

o **Substitutability**
  - MWEs resist substitution of their constituents by similar words
  - E.g. ‘many thanks’ cannot be expressed as ‘several thanks’ or ‘many gratitudes’

o Institutionalization
  - Results in statistical significance of collocations

o **Paraphrasability**
  - Sometimes it is possible to replace the MWE by a single word
  - E.g. leave out replaced by omit

- **Based on syntactic forms and compositionality**

o **Institutionalized Noun collocations -** E.g. traffic signal, George Bush, green card

o Phrasal Verbs (Verb-Particle constructions) - E.g. call up, eat up ,

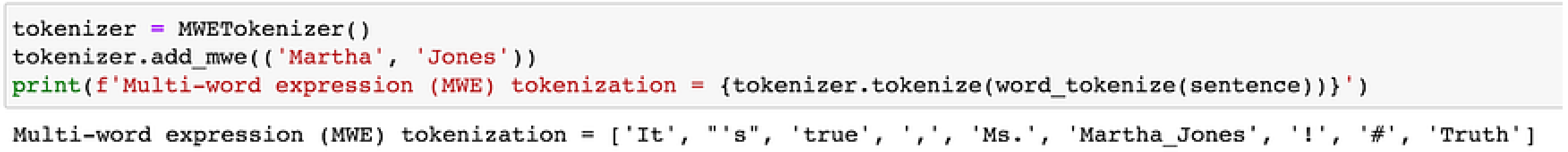
o Light verb constructions (V-N collocations) - E.g. fall asleep, give a demo

o Verb Phrase Idioms - E.g. sweep under the rug

MWETokenizer

- The multi-word expression tokenizer is a rule-based, “add-on” tokenizer offered by NLTK.
- Once the text has been tokenized by a tokenizer of choice, some tokens can be re-grouped into multi-word expressions.
- For example, the name Martha Jones is combined into a single token instead of being broken into two tokens.
- This tokenizer is very flexible since it is agnostic of the base tokenizer that was used to generate the tokens.

A MWETokenizer takes a string which has already been divided into tokens and retokenizes it, merging multi-word expressions into single tokens, using a lexicon of MWEs



```
tokenizer = MWETokenizer()
tokenizer.add_mwe(('Martha', 'Jones'))
print(f'Multi-word expression (MWE) tokenization = {tokenizer.tokenize(word_tokenize(sentence))}')
```

```
Multi-word expression (MWE) tokenization = ['It', "'s", 'true', ',', 'Ms.', 'Martha_Jones', '!', '#', 'Truth']
```

Student Evaluation

1.N-grams are defined as the combination of N keywords together. How many bi-grams can be generated from the given sentence: ***The Father of our nation is Mahatma Gandhiji***

A.8	B.9
C.7	D.4

2. It is the development of probabilistic models that are able to predict the next word in the sequence given the words that precede.

A. **Statistical Language Modelling**	B. Probabilistic Langualge Modeling
C. Neural Language Modelling	D. Natural Language Understanding

3. It is a measure of how good a probability distribution predicts a sample

A. Entropy	B. **Perplexity**
C. Cross-Entropy	D. Information Gain

4.What are the python libraries used in NLP?

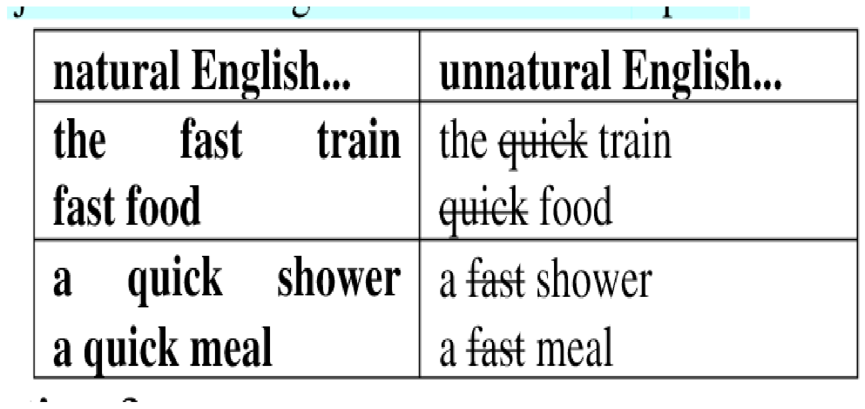
A. Pandas	B. NLTK
C. Spacy	D. **All the mentioned above**

COLLOCATIONS (ASSOCIATION MEASURES, COEFFICIENTS AND CONTEXT MEASURES)

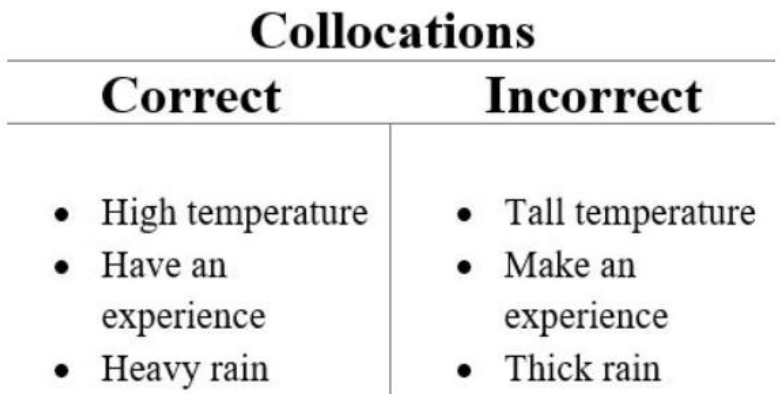
- Collocations are pairs or groups of words that frequently appear together in natural language.
- They often have a strong association or tendency to co-occur due to their semantic or syntactic relationship.
- **Association measures and coefficients** are statistical methods used to quantify the strength of the association between words in a collocation.
- These measures help identify **meaningful word combinations** and can be useful in various natural language processing tasks,
- such **as information retrieval, text mining, and machine translation.**

COLLOCATIONS (ASSOCIATION MEASURES, COEFFICIENTS AND CONTEXT MEASURES)

- Collocations are **phrases or expressions** containing multiple words, which are highly likely to co-occur.
- For example — ‘social media’, ‘school holiday’, ‘machine learning’, ‘Universal Studios Singapore’, etc.
- A collocation is two or more words that often go together.
- These combinations just sound "right" to native English speakers, who use them all the time.
- On the other hand, other combinations may be unnatural and just sound "wrong".
-



natural English...	unnatural English...
the fast train fast food	the ~~quick~~ train ~~quick~~ food
a quick shower a quick meal	a ~~fast~~ shower a ~~fast~~ meal



Collocations

Correct	Incorrect
• High temperature	• Tall temperature
• Have an experience	• Make an experience
• Heavy rain	• Thick rain

Why learn collocations?

- Your language will be more natural and more easily understood.
- You will have alternative and richer ways of expressing yourself.
- It is easier for our brains to remember and use language in chunks or blocks rather than as single words.

How to Learn collocations?

- Be aware of collocations, and try to recognize them when you see or hear them.
- Treat collocations as single blocks of language. Think of them as individual blocks or chunks, and learn strongly support, not strongly + support.
- When you learn a new word, write down other words that collocate with it (remember rightly, remember distinctly, remember vaguely, remember vividly).
- Read as much as possible. Reading is an excellent way to learn vocabulary and collocations in context and naturally.
- Revise what you learn regularly. Practise using new collocations in context as soon as possible after learning them.
- Learn collocations in groups that work for you. You could learn them by topic (time, number, weather, money, family) or by a particular word (take action, take a chance, take an exam).
- You can find information on collocations in any good learner's dictionary. And you can also find specialized dictionaries of collocations.

Collocations

Some commonly used association measures and coefficients for collocations

Pointwise Mutual Information (PMI): PMI measures the **degree of association** between two words, considering their **probability of co-occurrence** versus the **probability of their individual occurrences**. A higher PMI value indicates a stronger association.

Log-Likelihood Ratio (LLR): The LLR compares the likelihood of observing a collocation in a corpus to the likelihood of the individual words occurring independently. It helps identify statistically significant collocations.

Dice Coefficient: The Dice coefficient calculates the ratio of the number of times two words occur together to the sum of their individual occurrences. It ranges from 0 to 1, with 1 indicating a perfect collocation.

Mutual Information (MI): MI measures the reduction in uncertainty of one word's occurrence when the other word is known. It captures both positive and negative associations.

Frequency-based measures: Simple measures like raw frequency, conditional probability, and relative frequency can also be used to identify collocations.

Example

Suppose we have a large dataset of movie reviews, and we want to find collocations that frequently appear together in positive reviews. We are particularly interested in identifying collocations related to the theme of "amazing special effects" in movies.

Step 1: Preprocess the Data First, we preprocess the movie reviews by tokenizing them into words and removing any stop words, punctuation, and numbers.

Step 2: Calculate Association Measures Next, we calculate the association measures for different word pairs. Let's say we want to consider the Dice coefficient as our association measure.

- For each word pair (A, B), we calculate the Dice coefficient as follows:
- Dice(A, B) = (Number of times A and B co-occur) / (Number of times A occurs + Number of times B occurs)
- Step 3: Identify Significant Collocations Now, we look for collocations with high Dice coefficients, indicating strong associations. Let's say we find the following collocations with their corresponding Dice coefficients:

1. "amazing special" - Dice coefficient: 0.85
2. "special effects" - Dice coefficient: 0.80
3. "stunning visuals" - Dice coefficient: 0.75
4. "spectacular CGI" - Dice coefficient: 0.70
5. "mind-blowing action" - Dice coefficient: 0.65

Example

Step 4: Interpretation Based on the Dice coefficients, we can see that the word pairs "amazing special" and "special effects" have the highest associations in positive movie reviews.

This suggests that reviewers often mention "amazing special" and "special effects" together when praising movies with exceptional visual effects.

- In this real-world example, association measures like the Dice coefficient helped us identify significant collocations related to "amazing special effects" in movie reviews.
- These collocations can be useful for sentiment analysis, recommending movies to users who appreciate stunning visuals, or improving the understanding of what aspects of movies are highly praised by reviewers.

Types of collocation

- adverb + adjective: completely satisfied (NOT downright satisfied)
- adjective + noun: excruciating pain (NOT excruciating joy)
- noun + noun: a surge of anger (NOT a rush of anger)
- noun + verb: lions roar (NOT lions shout)
- verb + noun: commit suicide (NOT undertake suicide)
- verb + expression with preposition: burst into tears (NOT blow up in tears)
- verb + adverb: wave frantically (NOT wave feverishly)

Vector representation of words in NLP

- In Natural Language Processing (NLP), vector representation of words is a crucial concept used to convert words into numerical vectors.
- These vector representations are also known as **word embeddings.**
- Word embeddings are essential because they capture semantic and syntactic relationships between words and enable machine learning models to process and understand natural language text.

There are several methods to create word embeddings, and some of the commonly used techniques include:

One-Hot Encoding: In one-hot encoding, each word in the vocabulary is represented as a binary vector where all elements are zero except for the index corresponding to the word's position in the vocabulary, which is set to 1.

This method creates sparse and high-dimensional vectors that lack meaningful semantic relationships between words.

Techniques used

- **Word2Vec**: Word2Vec is a popular word embedding technique that learns continuous word representations from large amounts of text data.
- It offers two algorithms: **Continuous Bag of Words (CBOW) and Skip-gram.** These models generate dense word vectors that capture semantic similarities between words based on their context.
- **GloVe (Global Vectors for Word Representation)**: GloVe is another widely used method for learning word embeddings. It combines the global co-occurrence statistics of words in a corpus to create word vectors. GloVe embeddings capture both semantic and syntactic relationships between words.
- **Fast Text**: Fast Text is an extension of Word2Vec that represents each word as a bag of character n-grams. It can generate word embeddings for out-of-vocabulary words based on their character-level information, making it useful for handling misspellings and rare words.

Word 2 Vector Techniques

- **BERT (Bidirectional Encoder Representations from Transformers)**: BERT is a transformer-based model that generates contextual word embeddings. Unlike traditional methods that generate static embeddings, BERT considers the context of the word within the sentence, producing highly contextualized word representations.
- **ELMo (Embeddings from Language Models)**: ELMo is another contextual word embedding model that uses a bi-directional language model. It generates word embeddings based on the entire context of the sentence, capturing the polysemy (multiple meanings) of words.
- **ULMFiT (Universal Language Model Fine-tuning)**: ULMFiT is a transfer learning approach for NLP that utilizes pre-trained language models to fine-tune embeddings for specific downstream tasks. It enables efficient training on smaller datasets.
- These word embeddings can be used as input features for various NLP tasks, such as sentiment analysis, machine translation, named entity recognition, and more.
- They help improve the performance of NLP models by providing a more compact and meaningful representation of words in numerical form.

Word 2 Vector Example

- Let's demonstrate a simple example of word embeddings using Word2Vec, one of the popular techniques for learning word representations. For this example, we will use a small dataset of movie reviews and create word embeddings using the Word2Vec algorithm.
- Step 1: Preprocess the Data Suppose we have the following movie reviews:
  1. "The movie was fantastic, with amazing special effects."
  2. "The plot was engaging and kept me hooked till the end."
  3. "The acting was superb, especially by the lead actor."
  4. "The film had stunning visuals and great cinematography."
- We need to preprocess the data by tokenizing the sentences and converting the text to lowercase
- Step 2: Train Word2Vec Model Next, we train a Word2Vec model using the tokenized reviews
- Step 3: Retrieve Word Embeddings Now, we can access the word embeddings for specific words using the trained Word2Vec model
- Step 4: Similar Words We can also find words similar to a given word based on their embeddings

Word 2 Vector Example

- The resulting word embeddings and similarity scores will depend on the specific corpus and the number of training iterations, but they should capture the semantic relationships between words based on their context in the reviews.
- For instance, "fantastic" and "amazing" are likely to have a high similarity score, as they both frequently appear together in positive contexts in the dataset.
- Similarly, "plot" and "visuals" might also have a reasonable similarity score if they co-occur in sentences discussing movie elements.

REFERENCE

1) These are slides by stanford for all topics –

- https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html

2) This is e book which can be followed

- https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf

3) This is the channel by dan jurafsky and manning where they teach each topic from zero level.

- https://www.youtube.com/watch?v=808M7q8QX0E&list=PLaZQkZp6WhWyvdiP49JG-rjyTPck_hvEu

4) https://www.shiksha.com/online-courses/articles/pos-tagging-in-nlp/

5) https://web.stanford.edu/~jurafsky/slp3/

6) https://www.studocu.com/in/document/srm-institute-of-science-and-technology/natural-language-processing/nlp-notes-unit-1/39506511?origin=home-recent-1

21CSE356T: Natural Language Processing

Unit 1

Enabling Computers to Understand Human Language

Prepared by: Dr. Pritam Khan

Introduction to NLP

- Definition of NLP
- - Need for NLP
- - Examples: Text translation, sentiment analysis.

Applications of NLP

- Key Applications:
- - Sentiment Analysis
- - Chatbots
- - Spam Filtering
- Figures: Examples of NLP in action.

Levels of NLP

- Overview of Levels:
- - Morphology
- - Syntax
- - Semantics
- - Pragmatics.

Regular Expressions

- Use in NLP:
- - Pattern matching
- - Examples: Matching email patterns, token extraction.

Morphological Analysis

- Techniques:
- - Stemming
- - Lemmatization
- Example: 'running' → 'run'.

Tokenization

- Definition:
- - Breaking text into words or sentences.
- Examples:
- - Word tokenization: 'The cat sat.' → ['The', 'cat', 'sat'].

Feature Extraction: TF-IDF

TF-IDF: a statistical measure used in natural language processing (NLP) to evaluate the importance of a word in a document relative to a collection of documents (also called a corpus).

- - Term Frequency (TF)

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

Eg: For a document containing 100 words, if the word "NLP" appears 5 times, TF is 5/100 = 0.05

- - Inverse Document Frequency (IDF)

$$IDF(t) = \log \frac{\text{Total number of documents}}{\text{Number of documents containing the term } t}$$

Eg: If a corpus has 10,000 documents and the word "NLP" appears in 1,000 of them, then IDF = log (10,000/1,000) = 1

- Modeling using TF-IDF: product of Term Frequency and Inverse Document Frequency

TF-IDF (contd..)

Why TF-IDF is Important?
TF-IDF helps in:
•Identifying important words in a document while ignoring common words like "the" or "and."
•Building text-based models like search engines, document clustering, and topic modeling.

Modeling Using TF-IDF:
In machine learning, TF-IDF is often used as a feature representation for text data. Each word in the document becomes a feature, and its TF-IDF value is the corresponding weight. This feature matrix is fed into machine learning models like:
•**Classification**: For sentiment analysis, spam detection, etc.
•**Clustering**: For grouping similar documents together.
•**Similarity Measures**: To calculate how closely two documents are related.

Example:
If a document contains the terms ["cat", "dog", "fish"], and their TF-IDF scores are [0.5, 0.7, 0.2], the document is represented as a feature vector: [0.5,0.7,0.2][0.5, 0.7, 0.2][0.5,0.7,0.2]
This vector is used as input for further analysis or training models.

Parts of Speech Tagging

- **Definition**: Assigning parts of speech to words.
- **Examples**: Tagging in 'The dog runs fast.' → ['The/DT', 'dog/NN', 'runs/VB', 'fast/RB'].
- In the example provided, "DT," "NN," "VB," and "RB" are part-of-speech (POS) tags based on the Penn Treebank POS Tag Set. Here's what each of them represents:
- **DT**: Determiner (e.g., "the," "a," "this")
- **NN**: Noun, singular or mass (e.g., "dog," "cat," "water")
- **VB**: Verb, base form (e.g., "run," "eat," "go")
- **RB**: Adverb (e.g., "fast," "quickly," "easily")

Major POS Categories

Noun (NN, NNS, NNP, NNPS) – Names of people, places, things, or concepts.

1. **NN** (Singular Noun) – dog, table
2. **NNS** (Plural Noun) – dogs, tables
3. **NNP** (Proper Noun, Singular) – India, Google
4. **NNPS** (Proper Noun, Plural) – Americans, Indians

Pronoun (PRP, PRP$) – Replaces nouns.

1. **PRP** – he, she, it, they
2. **PRP$** (Possessive Pronoun) – his, her, their

Verb (VB, VBD, VBG, VBN, VBP, VBZ) – Action words.

1. **VB** (Base form) – run, eat
2. **VBD** (Past tense) – ran, ate
3. **VBG** (Gerund/Present Participle) – running, eating
4. **VBN** (Past Participle) – run, eaten
5. **VBP** (Present, non-3rd person singular) – run, eat
6. **VBZ** (Present, 3rd person singular) – runs, eats

Adjective (JJ, JJR, JJS) – Describes nouns.

1. **JJ** – big, beautiful
2. **JJR** (Comparative) – bigger, better
3. **JJS** (Superlative) – biggest, best

Adverb (RB, RBR, RBS) – Modifies verbs, adjectives, or other adverbs.

1. **RB** – quickly, silently
2. **RBR** (Comparative) – faster, sooner
3. **RBS** (Superlative) – fastest, soonest

Preposition (IN) – Shows relationships between words.

1. in, on, at, by, with

Conjunction (CC, IN) – Connects words or sentences.

1. **CC** (Coordinating Conjunction) – and, but, or
2. **IN** (Subordinating Conjunction) – because, although

Determiner (DT, PDT, WDT) – Introduces nouns.

1. **DT** – the, a, an
2. **PDT** (Predeterminer) – all, both
3. **WDT** (Wh-Determiner) – which, that

Numerals (CD) – Numbers and quantifiers.

1. one, two, 100

Interjection (UH) – Expresses emotions.

1. oh, wow, hey

Other POS Tags in Penn Treebank (Used in NLP Models)

Modal Verb (MD) – Expresses ability, possibility, or necessity.

1. can, could, will, should

Possessive Ending (POS) – Shows possession.

1. 's (as in "John's book")

Particle (RP) – Small function words used with verbs.

1. up (as in "give up"), off (as in "take off")

Wh-Pronoun (WP, WP$) – Question words.

1. **WP** – who, what, which
2. **WP$** (Possessive Wh-Pronoun) – whose

Existential 'there' (EX) – Indicates existence.

1. There is a problem.

Foreign Words (FW) – Words from other languages.

1. déjà vu, bona fide

List Item Marker (LS) – Used in lists.

1. 1., a), (i)

Symbol (SYM) – Mathematical or scientific symbols.

1. %, $, +, @

To (TO) – The word "to" when used before a verb.

1. to go, to run

Example Sentence with POS Tags

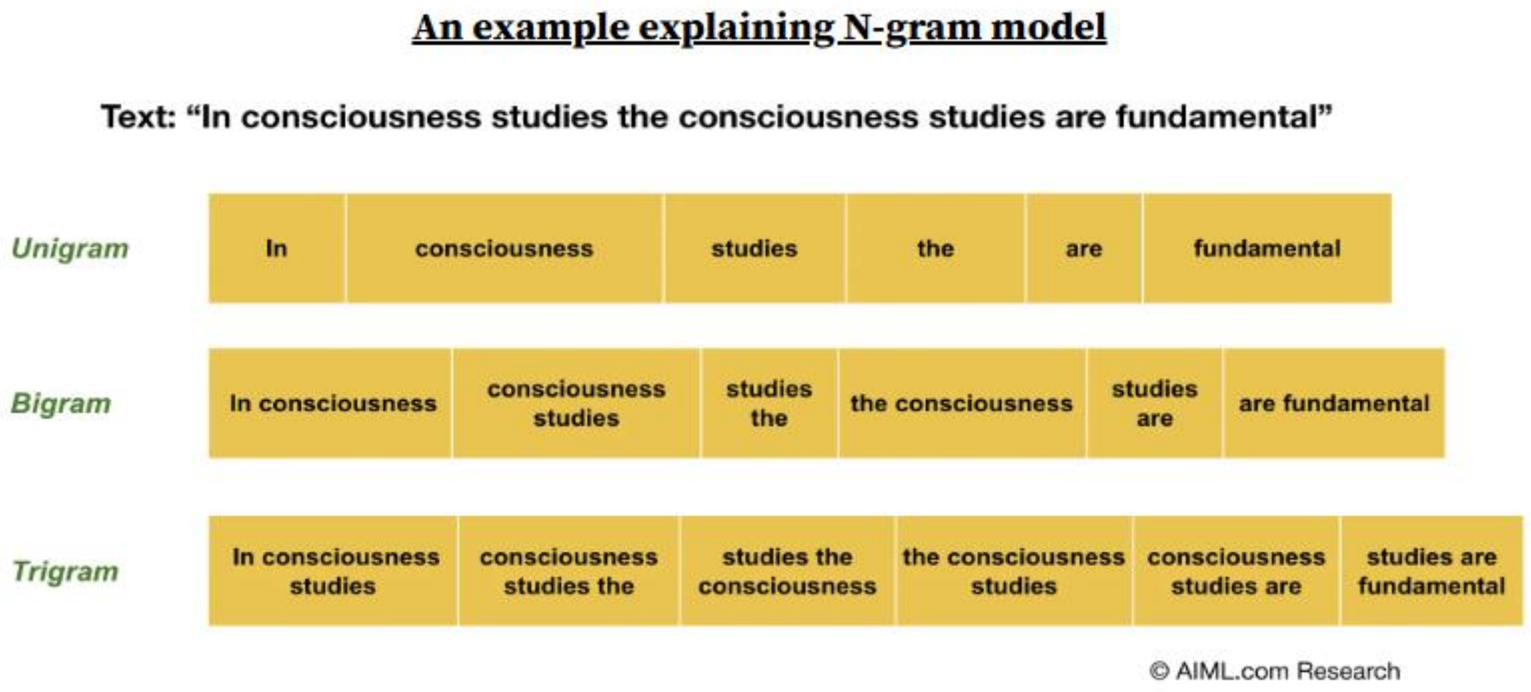
- *"The quick brown fox jumps over the lazy dog."*
- **DT (The)**
- **JJ (quick, brown)**
- **NN (fox, dog)**
- **VBZ (jumps)**
- **IN (over)**
- **DT (the)**
- **JJ (lazy)**

Named Entity Recognition

- Definition:
- - Identifying entities like names, places, etc.
- Examples:
- - 'John lives in New York.' → ['John/Person', 'New York/Location'].

N-grams and Smoothing

N-grams: Unigram, Bigram, Trigram



An example explaining N-gram model

Text: "In consciousness studies the consciousness studies are fundamental"

Unigram	In	consciousness	studies	the	are	fundamental
Bigram	In consciousness	consciousness studies	studies the	the consciousness	studies are	are fundamental
Trigram	In consciousness studies	consciousness studies the	studies the consciousness	the consciousness studies	consciousness studies are	studies are fundamental

© AIML.com Research

An example for n-gram
Source: AIML.com Research

Training an N-gram Model (example)

Unigram Model:

Set of Unigrams: {"in", "consciousness", "studies", "the", "are", "fundamental"}

The probability of a unigram *w*, can be estimated by taking the counts of how many times word *w* appears in the corpus divided by the total size of the corpus *M*. In the given example, *M* = 8

P ("consciousness") = Count("consciousness") / M = 2/8

P("are") = Count("are") / M = 1 / 8, and so on.

In general, the probability of a unigram is given by:

$$P(w) = \frac{C(w)}{m}$$

Training an N-gram Model (example)

Bigrams Model:

Set of Bigrams: {'in consciousness', 'consciousness studies', 'studies the', 'the consciousness', 'studies are', 'are fundamental'}

Using the example corpus above, the probability of a bigram "consciousness studies" would be the number of times the phrase "consciousness studies" occur in the corpus, divided by the count of the unigram "consciousness". One can also think of this as the conditional probability of "studies" given that "consciousness" appeared immediately before, and is represented as follows:

P("studies" | "consciousness") = Count("consciousness studies") / Count("consciousness") = 2/ 2

P("are" | "studies") = Count("studies are") / Count("studies") = 1/ 2

So, in general, the probability of a bigram is:

$$P(y|x) = \frac{C(x\ y)}{\sum_w C(x\ w)} = \frac{C(x\ y)}{C(x)}$$

Training an N-gram Model (example)

Trigrams Model:

Set of Trigrams: {'in consciousness studies', 'consciousness studies the', 'studies the consciousness', 'the consciousness studies', 'consciousness studies are', 'studies are fundamental'}

In mathematical terms this will be represented as the conditional probability of the third word, given that the previous two words occurred in the text as follows:

P("the" | "conscious studies") = Count("consciousness studies the") / Count ("consciousness studies") = 1/2

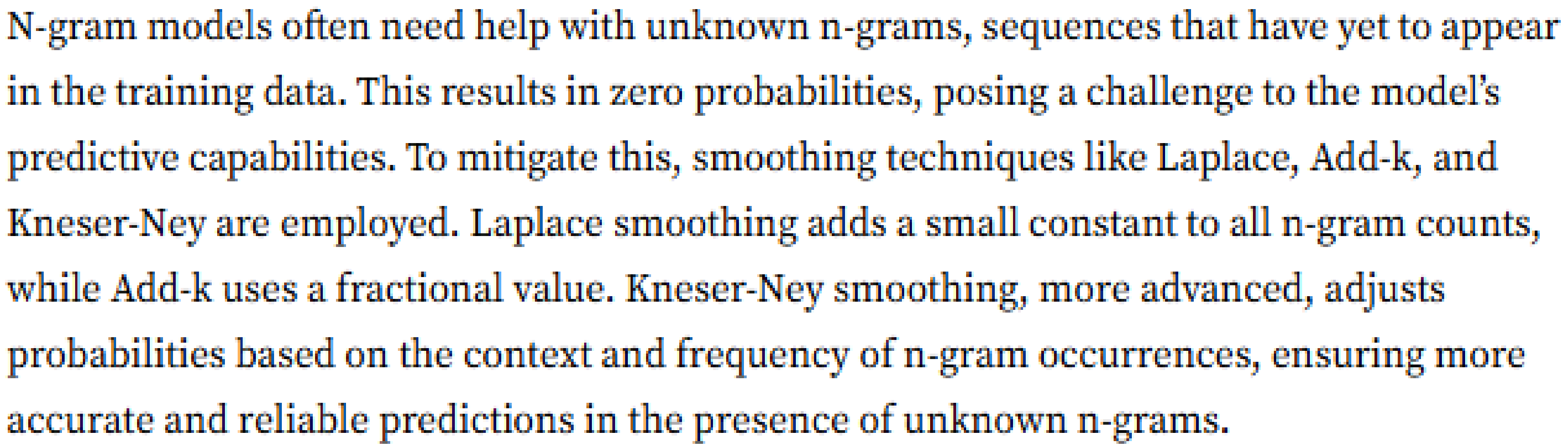
In general, the probability of a trigram is:

$$P\left(w_3 \mid w_1^2\right) = \frac{C\left(w_1^2 w_3\right)}{C\left(w_1^2\right)}$$

where,

$$C\left(w_1^2 w_3\right) = C\left(w_1 w_2 w_3\right) = C\left(w_1^3\right)$$

Smoothing: sequences with zero occurrence in training data



N-gram models often need help with unknown n-grams, sequences that have yet to appear in the training data. This results in zero probabilities, posing a challenge to the model's predictive capabilities. To mitigate this, smoothing techniques like Laplace, Add-k, and Kneser-Ney are employed. Laplace smoothing adds a small constant to all n-gram counts, while Add-k uses a fractional value. Kneser-Ney smoothing, more advanced, adjusts probabilities based on the context and frequency of n-gram occurrences, ensuring more accurate and reliable predictions in the presence of unknown n-grams.

Why is Smoothing Needed?

1. **Data Sparsity:** Language is vast, and many word combinations in a corpus will not appear in training data.
2. **Zero Probability Problem:** If a new n-gram or word appears in testing data but not in training, models like **n-gram language models** or **HMM POS taggers** assign **zero probability**, which is problematic.
3. **Better Generalization:** Helps models handle unseen data more effectively.

Types of Smoothing Techniques in NLP

1. Additive Smoothing (Laplace / Lidstone Smoothing)

Formula:

$$P(w) = \frac{C(w) + \alpha}{N + \alpha V}$$

where:

- $C(w)$ = count of word w
- N = total word count
- V = vocabulary size
- α = small constant (1 for **Laplace**, between 0 and 1 for **Lidstone**)

Example: P("cat | the") = C("the cat") / C("the")

2. Add-k Smoothing (Generalization of Laplace)

- Instead of adding **1**, we add a small fraction *k*.
- Works like Laplace smoothing but with more flexibility.

3. Good-Turing Smoothing

- Adjusts probabilities for **unseen events** by redistributing probability mass from low-frequency events.

Formula:

$$P(w) = \frac{(C(w)+1)N_{C(w)+1}}{N_{C(w)}}$$

where N_C is the count of words appearing C times.

- **Used in: n-gram language models.**
- **Example:** If a word appears once, Good-Turing reduces its probability and assigns some mass to unseen words.

4. Kneser-Ney Smoothing

- **Improves on Good-Turing** by adjusting probabilities **based on word contexts** rather than just frequency.
- **Uses a backoff mechanism:** If an n-gram is unseen, it "backs off" to lower n-grams.
- **Used in: Advanced language models like Google’s N-gram model.**
- **Example:** "San Francisco" is common, so "Francisco" is **more probable** after "San", even if "Francisco" alone is rare.

5. Backoff and Interpolation Smoothing

- **Backoff:** If a higher-order n-gram has **zero probability**, the model uses a **lower-order n-gram**.
- **Interpolation:** Combines **higher** and **lower** n-grams using a weighted sum.

Formula for Interpolation:

$$P(w_n|w_{n-1}) = \lambda_1 P(w_n|w_{n-1}) + \lambda_2 P(w_n)$$

where λ1 and λ2 are weights that sum to 1.

- **Used in: Speech recognition, predictive text.**