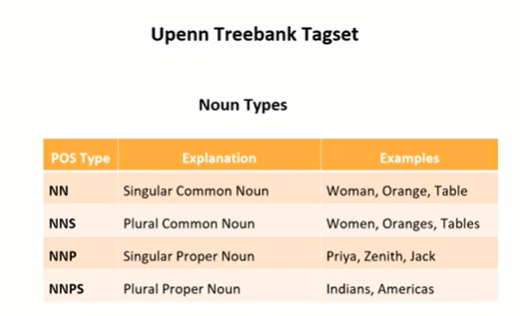
**Module 4**

1. **Write short notes on Penn Tree Bank?**

* A very popular part-of-speech tagset was developed by the University of Pennsylvania and is known as the Penn Treebank (PTB). This tagset contains 45 part-of-speech tags and has become fairly standardized for English. It is widely used in natural language processing (NLP) tools such as Stanford CoreNLP and spaCy.
* The POS tags in the PTB are small and compact, typically consisting of 2-4 capital characters.
* The Penn Treebank tagset, has been applied to the Brown corpus and a number of other corpora.
* Here is an example of a tagged sentence from the Penn Treebank version of the Brown corpus(in a flat ASCII file, tags are often represented after each word, following a slash, but tags can also be represented in various other ways).



A table of different types of english grammar

Description automatically generated

A table of different types of english language

Description automatically generated with medium confidence

A table of english prepositions

Description automatically generated

A table of english grammar

Description automatically generated with medium confidence

Write the table and explain the example

**Sentence 1: She plays tennis**.

* Word: She

POS Type: PRP (Pronoun)

Explanation: It is a pronoun.

* Word: plays

POS Type: VBZ (Verb, 3rd person singular present)

Explanation: It is a verb in the present tense for the 3rd person singular subject ("she").

* Word: tennis

POS Type: NN (Noun, singular or mass)

Explanation: It is a singular common noun.

**Sentence 2: They play football.**

* Word: They

POS Type: PRP (Pronoun)

Explanation: It is a pronoun.

* Word: play

POS Type: VBP (Verb, non-3rd person singular present)

Explanation: It is a verb in the present tense for a non-3rd person singular subject ("they").

* Word: football

POS Type: NN (Noun, singular or mass)

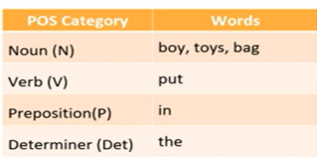
Explanation: It is a singular common noun.

1. **Explain POS tagging with example? Describe open class words and closed class words with examples?**

* Part-of-Speech (POS) tagging is the process of assigning a part-of-speech label to each word in a sentence. These labels indicate the grammatical category of the words, such as noun, verb, adjective, etc. POS tagging is a fundamental task in natural language processing (NLP) because it provides syntactic information that is crucial for understanding the structure and meaning of sentences.

**POS Tagging Problem:**

* Problem is to identify what is the actual category for each of the word.
* Given a text of English, identify the parts of speech of each word
* Text: **The boy put the toys in the bag.**

****

* POS tags tell us more information about the word and its overall role in the sentence
* POS tag of a word also tells some information about its neighbouring words.

e.g. Nouns are generally preceded by adjectives and/or determiners.

e.g. The girl, an ant, handsome boy

* Identifying POS tags is an important initial step for more complex downstream NLP tasks

-Parsing

-Information Extraction

-Sentiment Analysis

**-**Machine Translation

**POS tags can be broadly categorized into two categories:**

1. **Closed class:**

* Relatively fixed set of words (limited in number)
* Addition of new closed words is very rare.
* Mostly functional: to tie the concepts of a sentence together.

**e.g. Prepositions, Determiners, Pronoun, Connectives**

1. **Pronouns: Replace nouns to avoid repetition and manage referents.**

* Examples: "he," "she," "it," "they"
* Generally, no new additions: Pronouns are a fixed set.

1. **Prepositions: Show relationships between nouns or pronouns and other words in a sentence.**

* Examples: "in," "on," "at," "by"
* Generally, no new additions: The set of prepositions is quite stable.

1. **Conjunctions: Connect words, phrases, or clauses**.

* Examples: "and," "but," "or," "because"
* Generally, no new additions: Conjunctions are a fixed set.

1. **Determiners: Introduce nouns and provide context such as definiteness, quantity, and possession.**

* Examples: "the," "a," "some," "my"
* Generally, no new additions: Determiners are a fixed set.

1. **Open class:**

* Cannot associate a fixed set of words, new words can be frequently encountered with such POS tags
* Mostly content bearing: they refer to objects, actions and features in the world.
* Addition of new open words is very frequent.

**e.g. Nouns , Verbs, Adjectives, Adverbs**

1. **Nouns: Represent people, places, things, or ideas.**

* Examples: "dog," "city," "happiness," "innovation"
* New additions: "selfie," "blog," "cryptocurrency"

1. **Verbs: Describe actions, states, or occurrences.**

* Examples: "run," "think," "become," "innovate"
* New additions: "google" (as in to search on Google), "tweet" (as in to post on Twitter)

1. **Adjectives: Modify nouns by providing additional information.**

* Examples: "happy," "large," "innovative," "blue"
* New additions: "bingeable" (referring to content that can be watched in a single session), "glam" (short for glamorous)

1. **Adverbs: Modify verbs, adjectives, or other adverbs by providing additional information.**

* Examples: "quickly," "very," "well," "yesterday"
* New additions: "amazingly," "online" (used as an adverb)

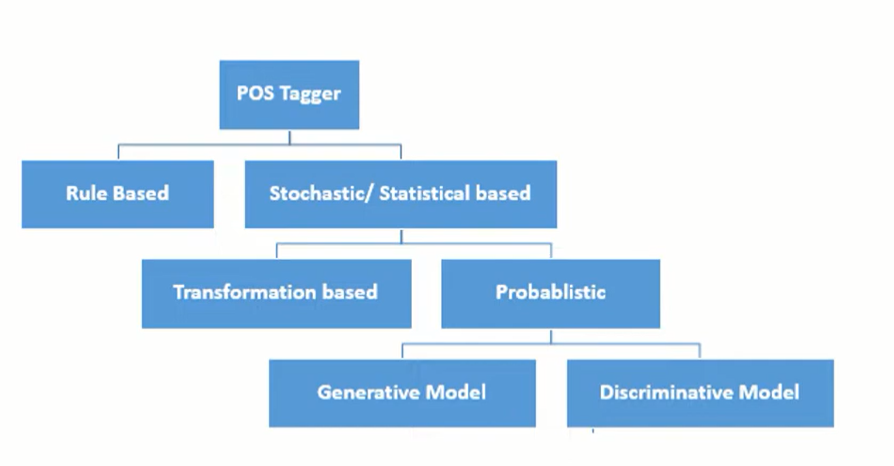
**Difficulties/Challenges**

* A word may have different POS tags. For e.g. to sort the numbers. Sort is verb. Bubble Sort: Sort is Noun.
* Closed class POS tags are not ambiguous, but their frequency is also limited in a statement.
* Generally, words with open class are more prone to multiple POS tags.
* Open class POS tags occur a lot more frequently and its tagging depends on its neighboring context that is previous and future words.

**Examples of ambiguities in POS tagging**

* The attack/NN was brutal.
* King was planning to attack/VB neighbouring states.
* Tigers usually attack/VBP their prey in a group.

**Classification of Taggers**



1. **Rule-based Approach**

* Rule-based part-of-speech tagging is the oldest approach that uses hand-written rules for tagging.
* Rule based taggers depends on dictionary or lexicon to get possible tags for each word to be tagged.
* Some language expert will sit together find out what are the symbol patterns.
* Hand-written rules are used to identify the correct tag when a word has more than one possible tag.
* Disambiguation is done by analyzing the linguistic features of the word, its preceding word, its following word and other aspects.

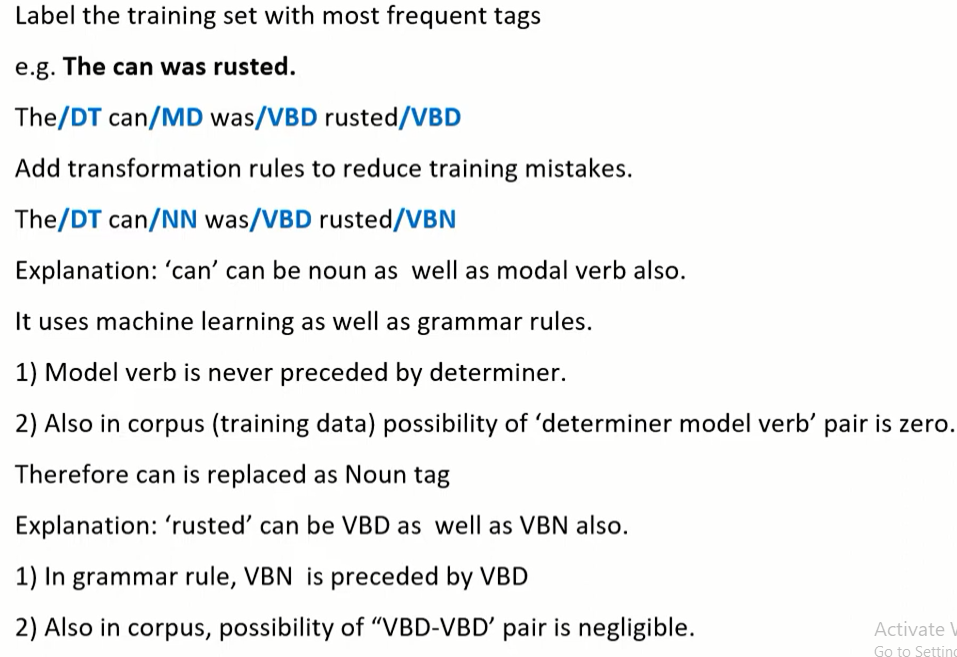
1. **Stochastic/ Statistical based Approach**

* Stochastic taggers generally resolve tagging ambiguities by using a training corpus to compute the probability of a given word having a given tag in a given context.
* It is based on concept of probability and machine learning.
* Training and Test corpus are used.
* Two approaches are:

a) Transformation based Tagger b) Probabilistic based Tagger

1. **Transformation based Tagger.**

* This tagger is based on concept of Transformation-Based Learning (TBL) approach.
* TBL uses supervised learning.
* There is assumption of pre-tagged training corpus.
* It combines idea of the rule-based and stochastic taggers.
* Like the rule based taggers, TBL is based on rules that specify what tags should be assigned to what words.
* But like the stochastic taggers, TBL is a machine learning technique, in which rules are automatically induced from the data.



1. **Probabilistic:**

* Approach is to "pick the most-likely tag for this word".
* Two approaches are there whether you generate the data from the class or the class from the data.
* Problem statement: Data available of the form [d, c] where d is observations and c is hidden classes.
* Two types of Model are there

1. Generative Model
2. Discriminative Model

A diagram of a model

Description automatically generated

A diagram of a person's body

Description automatically generated with medium confidence

A diagram of a data model

Description automatically generated

A screen shot of a diagram

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Rule-Based POS Tagging** | **Stochastic-Based POS Tagging** |
| Definition | Rules are manually crafted based on linguistic principles. | Probability theory is used to model uncertainty and make predictions. |
| Approach | Relies on predefined rules to assign POS tags. | Utilizes statistical models or machine learning algorithms. |
| Linguistic Knowledge | Requires linguistic expertise to design rules. | Relies less on linguistic expertise but more on annotated data. |
| Adaptability | May not adapt well to new domains or languages. | Can adapt to different domains and languages with sufficient training data. |
| Handling Ambiguity | May struggle with handling ambiguity and exceptions. | Can handle ambiguity by assigning probabilities to POS tags. |
| Performance | Generally has lower performance compared to statistical models. | Can achieve higher accuracy with large annotated datasets and advanced algorithms. |
| Examples | Regular expression-based taggers, handcrafted grammars. | Hidden Markov Models (HMMs), Maximum Entropy Markov Models (MEMMs), Conditional Random Fields (CRFs). |

1. **What is Markov process? How HMM is related with Markov Process?**

* A Markov process, also known as a Markov chain, is a stochastic process that exhibits the Markov property, named after the Russian mathematician Andrey Markov. The Markov property states that the future behaviour of the process depends only on its current state and not on the sequence of events that preceded it. In other words, given the present state, the future states are conditionally independent of the past states.
* A Markov process can be defined by a set of states and transition probabilities between those states. Mathematically, it can be represented as a directed graph, where each node represents a state, and the edges between nodes represent the transition probabilities between states. These transition probabilities are often arranged into a transition probability matrix, where each element represents the probability of transitioning from one state to another.
* Markov chain has 3 components
  1. Initial probability distribution
  2. One or more states
  3. Transition Probability distribution

**Example of a Markov Process:**

Consider a weather forecasting system with two states: "Sunny" and "Rainy". The system operates based on the following rules:

* + If it is sunny today, there is an 80% chance that it will be sunny tomorrow and a 20% chance of rain.
  + If it is rainy today, there is a 50% chance that it will be sunny tomorrow and a 50% chance of rain.
* This system can be represented as a Markov process:
* **Transition Probability Matrix:**

0.5

0.8

0.5

0.2

rainy

Sunny

* Each node represents a state: "Sunny" and "Rainy".
* The arrows represent the transition probabilities between states.
* The numbers next to the arrows represent the probabilities of transitioning from one state to another.

**For example:**

* There is an 80% chance of transitioning from "Sunny" to "Sunny", as indicated by the arrow labeled 0.8. There is a 20% chance of transitioning from "Sunny" to "Rainy", as indicated by the arrow labeled 0.2. Similarly, there is a 50% chance of transitioning from "Rainy" to "Sunny" and a 50% chance of transitioning from "Rainy" to "Rainy".
* Hidden Markov Models (HMMs) are extensions of Markov processes that incorporate hidden states. In an HMM, the states of the system are not directly observable, but rather, there are observations associated with each state. HMMs are used in various applications such as speech recognition, natural language processing, bioinformatics, and more.
* **Relation between HMMs and Markov Processes:**
* HMMs are related to Markov processes in that they are a special case of a more general Markov model. In an HMM, the underlying process generating the observed sequence is assumed to be a Markov process, where the hidden states transition according to Markovian dynamics, and each hidden state emits an observation with certain probabilities. HMMs provide a framework for modeling sequential data where the underlying states are not directly observable but can be inferred from the observed data using probabilistic inference algorithms like the Forward-Backward algorithm or the Viterbi algorithm.

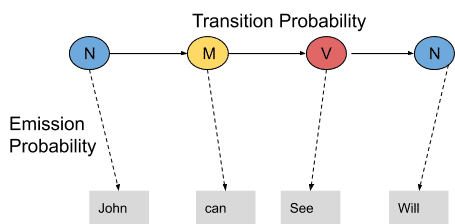
1. **What are the limitations of Hidden Markov Model?**
2. **Short Memory**: HMMs only consider the current state to predict the next state, ignoring long-term dependencies.
3. **Discrete States**: HMMs use a finite set of states, which can be limiting for modeling continuous or complex data.
4. **Independent Observations**: HMMs assume that observations are independent given the current state, which is not always true in real-world data.
5. **Scalability**: HMMs can be slow and computationally expensive to train, especially with large datasets and many states.
6. **State Duration**: HMMs have difficulty modeling the time spent in each state, often assuming it follows a geometric distribution, which might not fit all scenarios.
7. **Parameter Estimation**: Estimating the model parameters accurately can be challenging, especially with limited or noisy data.
8. **Fixed Structure**: The number of states must be determined beforehand, which can be tricky and may require expert knowledge.
9. **How HMM is used for POS tagging? Explain in detail? Describe the process of Parts of Speech Tagging using Hidden Markov Model? what do you mean by transmission probability matrix and emission probability matrix. Explain with one example?**

* **The hidden Markov Model** (HMM) is a [statistical model](https://www.geeksforgeeks.org/difference-between-statistical-model-and-machine-learning/) that is used to describe the probabilistic relationship between a sequence of observations and a sequence of hidden states. It is often used in situations where the underlying system or process that generates the observations is unknown or hidden, hence it has the name “Hidden Markov Model.”
* It is used to predict future observations or classify sequences, based on the underlying hidden process that generates the data.
* An HMM consists of two types of variables: hidden states and observations.
* The hidden states are the underlying variables that generate the observed data, but they are not directly observable.
* The observations are the variables that are measured and observed

.

* The relationship between the hidden states and the observations is modeled using a probability distribution. The Hidden Markov Model (HMM) is the relationship between the hidden states and the observations using two sets of probabilities: the transition probabilities and the emission probabilities.
  1. The transition probabilities describe the probability of transitioning from one hidden state to another.
  2. The emission probabilities describe the probability of observing an output given a hidden state.

**POS tagging with Hidden Markov Model**



In this example, we consider only 3 POS tags that are noun, model and verb. Let the sentence “ Ted will spot Will ” be tagged as noun, model, verb and a noun and to calculate the probability associated with this particular sequence of tags we require their Transition probability and Emission probability.

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.

Now, what is the probability that the word Ted is a noun, will is a model, spot is a verb and Will is a noun. These sets of probabilities are Emission probabilities and should be high for our tagging to be likely.

Let us calculate the above two probabilities for the set of sentences below

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

Note that Mary Jane, Spot, and Will are all names.

In the above sentences, the word Mary appears four times as a noun. To calculate the emission probabilities, let us create a counting table in a similar manner.

The transition probability matrix (also called the transmission probability matrix) defines the probabilities of moving from one hidden state to another. This matrix captures the dynamics of the hidden states, which are not directly observable.

The emission probability matrix (also called the observation probability matrix) defines the probabilities of observing each possible output from a given hidden state. This matrix relates the hidden states to the observed data.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

Now let us divide each column by the total number of their appearances for example, ‘noun’ appears nine times in the above sentences so divide each term by 9 in the noun column. We get the following table after this operation.

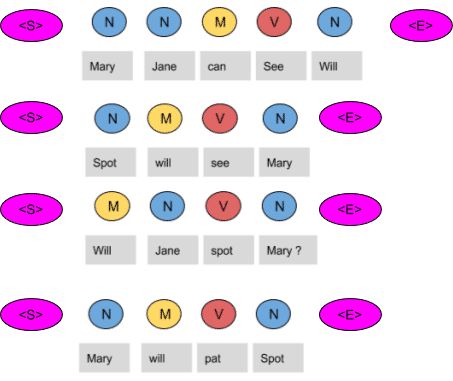
|  |  |  |  |
| --- | --- | --- | --- |
| Words | Noun | Model | Verb |
| Mary | 4/9 | 0 | 0 |
| Jane | 2/9 | 0 | 0 |
| Will | 1/9 | 3/4 | 0 |
| Spot | 2/9 | 0 | 1/4 |
| Can | 0 | 1/4 | 0 |
| See | 0 | 0 | 2/4 |
| pat | 0 | 0 | 1 |

From the above table, we infer that

* 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

In a similar manner, you can figure out the rest of the probabilities. These are the emission probabilities.

Next, we have to calculate the transition probabilities, so define two more tags <S> and <E>. <S> is placed at the beginning of each sentence and <E> at the end as shown in the figure below.

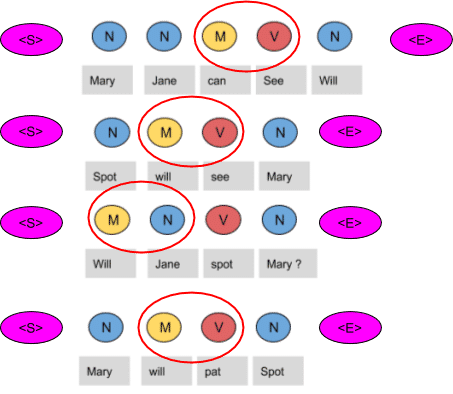


Let us again create a table and fill it with the co-occurrence counts of the tags.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | N | M | V | <E> |
| <S> | 3 | 1 | 0 | 0 |
| N | 1 | 3 | 1 | 4 |
| M | 1 | 0 | 3 | 0 |
| V | 4 | 0 | 0 | 0 |

In the above figure, we can see that the <S> tag is followed by the N tag three times, thus the first entry is 3.The model tag follows the <S> just once, thus the second entry is 1. In a similar manner, the rest of the table is filled.

Next, we divide each term in a row of the table by the total number of co-occurrences of the tag in consideration, for example, The Model tag is followed by any other tag four times as shown below, thus we divide each element in the third row by four.



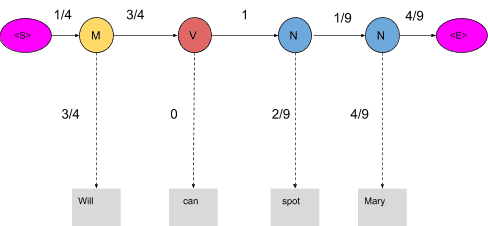
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | N | M | V | <E> |
| <S> | 3/4 | 1/4 | 0 | 0 |
| N | 1/9 | 3/9 | 1/9 | 4/9 |
| M | 1/4 | 0 | 3/4 | 0 |
| V | 4/4 | 0 | 0 | 0 |

These are the respective transition probabilities for the above four sentences. Now how does the HMM determine the appropriate sequence of tags for a particular sentence from the above tables? Let us find it out.

Take a new sentence and tag them with wrong tags. Let the sentence, ‘ Will can spot Mary’  be tagged as-

* Will as a  model
* Can as a verb
* Spot as a noun
* Mary as a noun

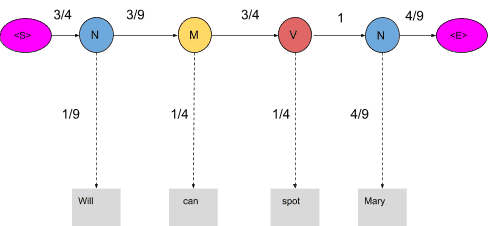
Now calculate the probability of this sequence being correct in the following manner.



The probability of the tag Model (M) comes after the tag <S> is ¼ as seen in the table. Also, the probability that the word Will is a Model is 3/4. In the same manner, we calculate each and every probability in the graph. Now the product of these probabilities is the likelihood that this sequence is right. Since the tags are not correct, the product is zero.

**1/4\*3/4\*3/4\*0\*1\*2/9\*1/9\*4/9\*4/9=0**

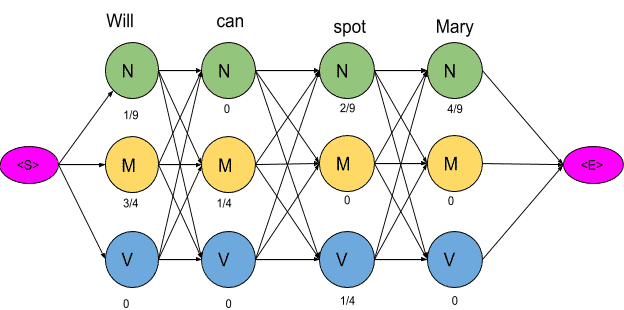
When these words are correctly tagged, we get a probability greater than zero as shown below



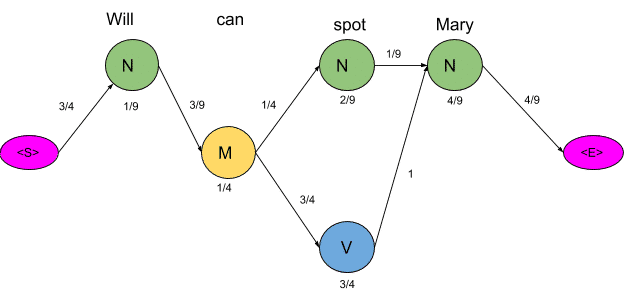
Calculating  the product of these terms we get,

**3/4\*1/9\*3/9\*1/4\*3/4\*1/4\*1\*4/9\*4/9=0.00025720164**

For our example, keeping into consideration just three POS tags we have mentioned, 81 different combinations of tags can be formed. In this case, calculating the probabilities of all 81 combinations seems achievable. But when the task is to tag a larger sentence and all the POS tags in the Penn Treebank project are taken into consideration, the number of possible combinations grows exponentially and this task seems impossible to achieve. Now let us visualize these 81 combinations as paths and using the transition and emission probability mark each vertex and edge as shown below.



The next step is to delete all the vertices and edges with probability zero, also the vertices which do not lead to the endpoint are removed. Also, we will mention-



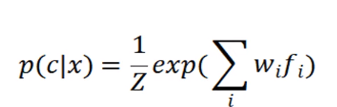
Now there are only two paths that lead to the end, let us calculate the probability associated with each path.

<S>→N→M→N→N→<E> =**3/4\*1/9\*3/9\*1/4\*1/4\*2/9\*1/9\*4/9\*4/9=0.00000846754**

<S>→N→M→N→V→<E>=**3/4\*1/9\*3/9\*1/4\*3/4\*1/4\*1\*4/9\*4/9=0.000257**

1. **Pos tagging Maximum entropy model**

* For POS disambiguation, HMM POS tagging uses restricted features.
* Both the tag and the observed word in HMM are dependent on the previous tag.
* To establish the POS tag for an observed word, further features must be considered.
* A discriminatory model is the Maximum Entropy Model.
* It does POS tagging for a given sentence by combining several heterogeneous features in a probabilistic framework.
* The MaxEnt is based on the Principle of Maximum Entropy, which states that of all models that match training data, the one with the highest entropy should be chosen.
* When the conditional independence of the features cannot be assumed, the Maximum Entropy classifier is utilised.
* This means that features are interdependent.
* This is especially evident in problems involving text categorization, because the characteristics are words and they are not independent.
* Maximum Entropy Modeling (MaxEnt) is a machine learning framework. Classification is the problem of taking a single observation, extracting some useful features describing the observation, and then classifying the observation into one of a set of discrete classes depending on these features.
* **Probabilistic classifier:** Indicates the likelihood that the observation belongs to that category.
* **Non-sequential categorization -** When it comes to text classification, we may need to evaluate whether or not an email should be categorised as spam. We must assess whether a statement or document represents a favourable or negative attitude in sentiment analysis. A period character (V) must be classified as a sentence boundary or not.
* MaxEnt belongs to the family of classifiers known as the exponential or log-linear classifiers.
* MaxEnt works by extracting some set of features from the input, combining them linearly (multiply each feature by a weight and then add them up), and then using this sum as an exponent.
* Example: tagging - A feature for tagging might be this word ends in ed or the previous word was adverb

****

**A white paper with black text

Description automatically generated**

**Module 5**

1. **Discuss various relations among the word senses. relation between synonym, antonym, hypernemy, hyponemy, polysemy**

* A sense (or **word sense**) is a discrete representation of one aspect of the meaning of a word. Loosely following lexicographic tradition, we represent each sense with a superscript : bank1 and bank2, mouse1 and mouse2. In context, it's easy to see the different meanings :
  1. mouse1 : .... a mouse controlling a computer system in 1968.
  2. mouse2 : .... a quiet animal like a mouse
  3. bank1 : ... a bank can hold the investments in a custodial account ...
  4. bank2 :... as agriculture burgeons on the east bank, the river ...
* The senses of a word might not have any particular relation between them; it may be almost coincidental that they share an orthographic form. For example, the financial institution and sloping mound senses of bank seem relatively unrelated.
* In such cases we say that the two senses are homonyms, and the relation between the senses is one of homonymy. Thus bank1 ("financial institution") and bank2 ("sloping mound") are homonyms, as are the sense of bat meaning 'club for hitting a ball' and the one meaning 'nocturnal flying animal'. We say that these two uses of bank are homographs, as are the two uses of bat, because they are written the same.
* Two words can be homonyms in a different way if they are spelled differently but pronounced the same, like write and right, or piece and peace. We call these homophones; they are one cause of real-word spelling errors.

A close-up of a text

Description automatically generated

A colorful circle with yellow text

Description automatically generated

A screenshot of a computer

Description automatically generated

A blue rectangle with yellow text

Description automatically generated

A diagram of food and fruits

Description automatically generated

A close-up of a text

Description automatically generated

A close-up of a white board

Description automatically generated

1. **Explain lexicon, lexeme and the different types of relations that hold between Lexemes**

* **Lexicon**
* Definition: The lexicon of a language is its vocabulary, including its words and expressions. It is essentially a mental repository or database of all the words (and their meanings) that a speaker of the language knows.
* Example: The English lexicon includes words like "cat," "run," "happiness," and idiomatic expressions like "kick the bucket."
* **Lexeme**
* Definition: A lexeme is the fundamental unit of lexical meaning, which can encompass various forms that a word might take based on inflection. A lexeme is represented by a base form or lemma.
* Example: The lexeme "run" includes forms like "run," "runs," "ran," and "running."
* **Types of Relations Between Lexemes:**

synonym, antonym, hypernemy, hyponemy, polysemy etc same define just replace word with lexemes.

1. **State the difference between hypernymy and hyponymy and give an example of each?**

**Hypernymy**

* Definition: A hypernym is a word with a broad meaning that encompasses the meanings of more specific words (hyponyms). It represents a general category or class.
* Example: "Animal" is a hypernym for "dog," "cat," and "elephant."

**Hyponymy**

* Definition: A hyponym is a word with a more specific meaning that falls under the broader category of a hypernym.
* Example: "Dog" is a hyponym of "animal."

**Example in Context**

* Hypernym: "Fruit" (general category)
* Hyponyms: "Apple," "banana," "cherry" (specific instances of the general category)

1. **State the difference between homonymy and polysemy and give an example of each?**

**Homonymy**

* Definition: Homonyms are words that are spelled the same and/or sound the same but have different, unrelated meanings. Homonyms can be further divided into homophones (same pronunciation) and homographs (same spelling).
* Example: "Bat" (the flying mammal) and "bat" (the equipment used in baseball) are homonyms.

**Polysemy**

* Definition: Polysemy occurs when a single word has multiple related meanings. The meanings share a common origin or are conceptually connected.
* Example: "Bank" (the side of a river) and "bank" (a financial institution) have related meanings derived from the concept of a place where something is stored or accumulated.

**Example in Context**

* Homonymy Example: "Lead" (to guide) and "lead" (a type of metal) are homonyms.
* Polysemy Example: "Book" can mean a physical object with pages (as in a novel) and the act of making a reservation (as in "to book a flight").

1. **What do you mean by word sense disambiguation (WSD)? Discuss dictionary based approach for WSD.**

What does this word mean?

* This plant needs to be watered each day.

= living plant

* This plant manufactures 1000 widgets each day.

= factory

* We say that these words all have various word senses and that some of the senses are synonymous with one another. The process of choosing the right sense in context is called word sense disambiguation (or WSD).
* WSD algorithms take as input a word in context and a fixed inventory of potential word senses and outputs the correct word sense in context.
* Word Sense Disambiguation basically solves the ambiguity that arises in determining the meaning of the same word used in different situations.

**Approaches and Methods to Word Sense Disambiguation (WSD)**

1. **Dictionary-based or Knowledge-based Methods**

* As the name suggests, for disambiguation, these methods primarily rely on dictionaries, treasures and lexical knowledge base. They do not use corpora evidences for disambiguation. The Lesk method is the seminal dictionary-based method introduced by Michael Lesk in 1986. The Lesk definition, on which the Lesk algorithm is based is “measure overlap between sense definitions for all words in context”. However, in 2000, Kilgarriff and Rosensweig gave the simplified Lesk definition as “measure overlap between sense definitions of word and current context”, which further means identify the correct sense for one word at a time. Here the current context is the set of words in surrounding sentence or paragraph.

1. **Supervised Methods**

* For disambiguation, machine learning methods make use of sense-annotated corpora to train. These methods assume that the context can provide enough evidence on its own to disambiguate the sense. In these methods, the words knowledge and reasoning are deemed unnecessary. The context is represented as a set of “features” of the words. It includes the information about the surrounding words also. Support vector machine and memory-based learning are the most successful supervised learning approaches to WSD. These methods rely on substantial amount of manually sense-tagged corpora, which is very expensive to create.

1. **Semi-supervised Methods**

* Due to the lack of training corpus, most of the word sense disambiguation algorithms use semi-supervised learning methods. It is because semi-supervised methods use both labelled as well as unlabeled data. These methods require very small amount of annotated text and large amount of plain unannotated text. The technique that is used by semisupervised methods is bootstrapping from seed data.

1. **Unsupervised Methods**

* These methods assume that similar senses occur in similar context. That is why the senses can be induced from text by clustering word occurrences by using some measure of similarity of the context. This task is called word sense induction or discrimination. Unsupervised methods have great potential to overcome the knowledge acquisition bottleneck due to non-dependency on manual efforts.

Application

Machine Translation

Information Retrieval

Text Mining

Lexicography

1. **Word net**

* WordNet is a lexical database of the English language.
* It groups words into sets of synonyms called synsets, provides short definitions, and records the various semantic relationships between these synonym sets.
* It's a resource commonly used in natural language processing and computational linguistics tasks, as well as in educational and research contexts.
* Developed at Princeton University in the Cognitive Science Laboratory starting in 1985, WordNet has become a fundamental resource in natural language processing (NLP), computational linguistics, and related fields.

**Key components and features of WordNet**

* **Synsets:** A synset, or synonym set, is a collection of words that are synonymous or closely related in meaning. Each synset represents a distinct concept or idea. For example, the synset for the word "dog" might include synonyms like "canine," "pooch," and "hound."
* **Words and Lemmas:** WordNet includes individual words, or lemmas, which are organized into synsets. Lemmas represent the base or dictionary form of a word. For example, the lemma "running" might be included in synsets related to activities or physical exercise.
* **Definitions:** Each synset in WordNet is accompanied by a short definition that describes the meaning of the concept represented by that synset. These definitions help users understand the nuances of each word's usage and meaning
* **Part-of-Speech (POS) Tags:** WordNet assigns part-of-speech tags to each word or lemma, indicating whether it functions as a noun, verb, adjective, or adverb.

This information is crucial for disambiguating word meanings and identifying appropriate synonyms in different contexts.

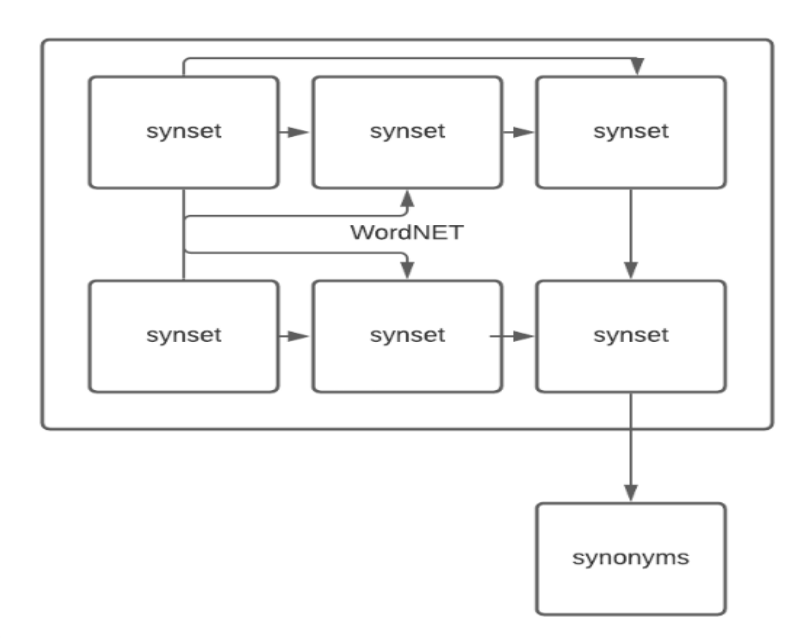
* **Semantic Relationships:** WordNet records various semantic relationships between synsets, providing valuable information about the connections between words and concepts. Some common semantic relationships in WordNet include:

• Hypernyms and Hyponyms: Hypernyms represent more general concepts, while hyponyms are more specific. For example, "animal" is a hypernym of "dog," and "dog" is a hyponym of "animal."

• Meronyms and Holonyms: Meronyms are parts or components of a whole, while holonyms are the wholes that contain those parts. For example, "wheel" is a meronym of "car," and "car" is a holonym of "wheel."

• Synonyms and Antonyms: WordNet identifies synonyms (words with similar meanings) and antonyms (words with opposite meanings) within and between synsets

* **Hierarchical Structure:** WordNet's synsets and semantic relationships are organized in a hierarchical structure, allowing users to navigate through different levels of abstraction and specificity. This hierarchical organization aids in understanding the relationships between concepts and facilitates various NLP tasks such as word sense disambiguation and semantic similarity measurement.
* **Applications:** WordNet is widely used in natural language processing tasks such as text analysis, information retrieval, machine translation, sentiment analysis, and more. Its rich collection of words, synsets, and semantic relationships provides a valuable resource for understanding and processing natural language text.



1. **How do you find the Cosine distance between the documents?**

* Text Similarity has to determine how the two text documents close to each other in terms of their context or meaning. There are various text similarity metric exist such as Cosine similarity, Euclidean distance and Jaccard Similarity. All these metrics have their own specification to measure the similarity between two queries.
* Cosine similarity is one of the metric to measure the text-similarity between two documents irrespective of their size in Natural language Processing. A word is represented into a vector form. The text documents are represented in n-dimensional vector space.
* Mathematically, Cosine similarity metric measures the cosine of the angle between two n-dimensional vectors projected in a multi-dimensional space. The Cosine similarity of two documents will range from 0 to 1. If the Cosine similarity score is 1, it means two vectors have the same orientation. The value closer to 0 indicates that the two documents have less similarity.
* Cosine similarity is the cosine of the angle between two vectors and it is used as a distance evaluation metric between two points in the plane. The cosine similarity measure operates entirely on the cosine principles where with the increase in distance the similarity of data points reduces.
* In Data Minning, similarity measure refers to distance with dimensions representing features of the data object, in a dataset. If this distance is less, there will be a high degree of similarity, but when the distance is large, there will be a low degree of similarity.

The cosine similarity is given as

where:

𝐴⋅𝐵 is the dot product of vectors 𝐴 and 𝐵 = ,

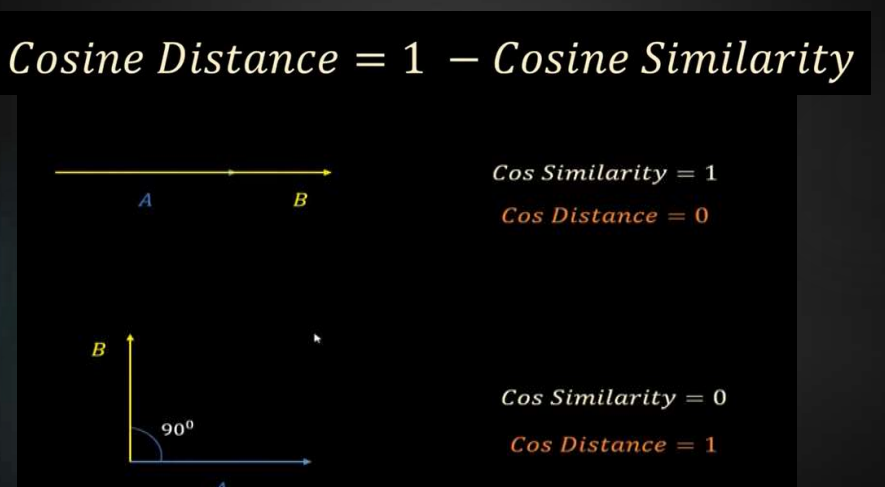
∥𝐴∥ is the magnitude (or norm) of vector 𝐴 = ,

∥𝐵∥ is the magnitude (or norm) of vector 𝐵 = .

A black background with text and numbers

Description automatically generated with medium confidence

* The similarity can take values between -1 and +1. Smaller angles between vectors produce larger cosine values, indicating greater cosine similarity.
* For example:
  + When two vectors have the same orientation, the angle between them is 0, and the cosine similarity is 1.
  + Perpendicular vectors have a 90-degree angle between them and a cosine similarity of 0.
  + Opposite vectors have an angle of 180 degrees between them and a cosine similarity of -1.



A paper with black text and numbers

Description automatically generated

A close up of a paper

Description automatically generated

**Finding Cosine Distance Between Documents**

To find the cosine distance between two documents, follow these steps:

1. **Vector Representation:**

Represent each document as a vector in a multi-dimensional space. Typically, this involves converting text documents into numerical vectors using methods like term frequency (TF), term frequency-inverse document frequency (TF-IDF), or word embeddings.

1. **Compute Dot Product:**

Calculate the dot product of the two vectors.

1. **Compute Magnitudes:**

Calculate the magnitude (norm) of each vector.

1. **Calculate Cosine Similarity:**

Use the dot product and magnitudes to find the cosine similarity.

1. **Calculate Cosine Distance:**

Subtract the cosine similarity from 1 to get the cosine distance.

**Module 6**

* **Sentiment analysis** is a classification task in the area of natural language processing. Sometimes called 'opinion mining,' sentiment analysis models transform the opinions found in written language or speech data into actionable insights.
* Sentiment analysis involves determining whether the author or speaker's feelings are positive, neutral, or negative about a given topic. For instance, you would like to gain a deeper insight into customer sentiment, so you begin looking at customer feedback under purchased products or comments under your company's post on any social media platform. You would like to know if the customer is pleased with your services, neutral, or if he/she has any complaints, meaning whether the customer has a neutral, positive or negative sentiment regarding your products, services or actions. Figuring this out is called sentiment analysis.

**NLP methods for sentiment analysis**

* In classical methods, we define features and models that can then be identified by the sentiment analysis system. This can be done by :

1. Using a dictionary of manually defined keywords,
2. Creating a 'bag of words',
3. Using the TF-IDF strategy.

* Using a dictionary of manually defined keywords is based on the assumption that we know what words are typically associated with positive and negative emotions. For example, if we are going to classify movie reviews, we expect to find words such as "great", "super", and "love" in positive comments and words like "hate", "bad", and "awful" in negative comments.
* We can count the number of occurrences of every selected word to define feature vectors. Then, we can train a sentiment analysis classifier on each comment.
* This approach restricts you to manually defined words, and it is unlikely that every possible word for each sentiment will be thought of and added to the dictionary.
* This is where a **bag of words** comes in. Instead of calculating only words selected by domain experts, we can calculate the occurrences of every word that we have in our language (or every word that occurs at least once in all of our data). This will cause our vectors to be much longer, but we can be sure that we will not miss any word that is important for prediction of sentiment.
* We can think about **TF-IDF** as a modified version of the bag of words. Instead of treating every word equally, we normalize the number of occurrences of specific words by the number of its occurrences in our whole data set and the number of words in our document (comments, reviews, etc.). This means that our model will be less sensitive to occurrences of common words like "and", "or", "the", "opinion" etc., and focus on the words that are valuable for analysis.

A diagram of a diagram

Description automatically generated

* **Machine Translation** or MT or robotized interpretation is simply a procedure when computer software translates text from one language to another without human contribution. At its fundamental level, machine translation performs a straightforward replacement of atomic words in a single characteristic language for words in another.
* In simple language, we can say that machine translation works by using computer software to translate the text from one source language to another target language.

**Types of machine translation**

There are different types of machine translation.

1. **Statistical Machine Translation or SMT**

* It works by alluding to statistical models that depend on the investigation of huge volumes of bilingual content. It expects to decide the correspondence between a word from the source language and a word from the objective language. A genuine illustration of this is Google Translate.
* Presently, SMT is extraordinary for basic translation, however its most noteworthy disadvantage is that it doesn't factor in context, which implies translation can regularly be wrong or you can say, don't expect great quality translation.
* There are several types of statistical-based machine translation models which are: Hierarchical phrase-based translation, Syntax-based translation, Phrase-based translation, Word-based translation.

1. **Rule-based Machine Translation or RBMT**

* RBMT basically translates the basics of grammatical rules. It directs a grammatical examination of the source language and the objective language to create the translated sentence. But, RBMT requires broad editing, and its substantial reliance on dictionaries implies that proficiency is accomplished after a significant period.

1. **Hybrid Machine Translation or HMT**

* HMT, as the term demonstrates, is a mix of RBMT and SMT. It uses a translation memory, making it unquestionably more successful regarding quality. Nevertheless, even HMT has a lot of downsides, the biggest of which is the requirement for enormous editing, and human translators will also be needed. There are several approaches to HMT like multi-engine, statistical rule generation, multi-pass, and confidence-based.

1. **Neural Machine Translation or NMT**

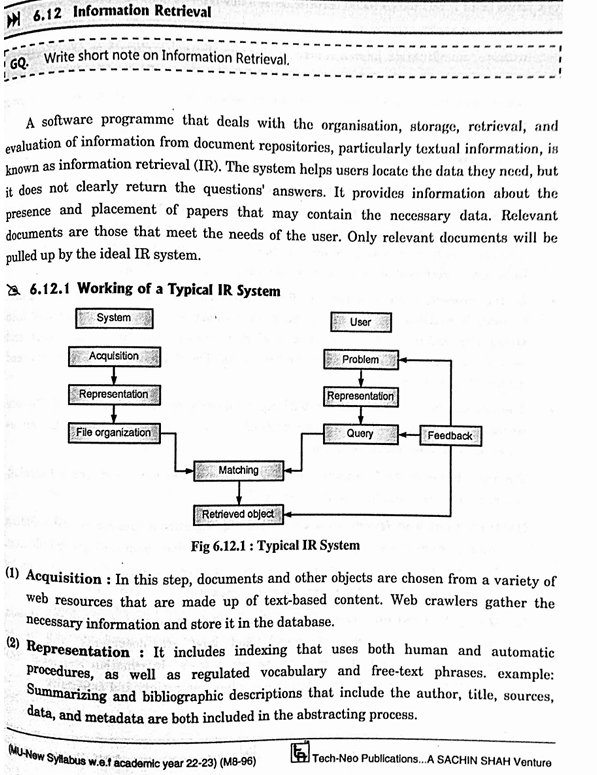
* NMT is a type of machine translation that relies upon neural network models (based on the human brain) to build statistical models with the end goal of translation. The essential advantage of NMT is that it gives a solitary system that can be prepared to unravel the source and target text. Subsequently, it doesn't rely upon specific systems that are regular to other machine translation systems, particularly SMT.
* **Explain text categorization in NLP**

A diagram of a process

Description automatically generated

Text categorization, also known as text classification, is a fundamental task in Natural Language Processing (NLP) that involves assigning predefined categories or labels to textual documents based on their content. Here's an overview of text categorization in NLP:

1. **Text Preprocessing**: Before performing text categorization, textual data undergoes preprocessing steps such as tokenization (splitting text into words or tokens), lowercasing, removing punctuation, and possibly stemming or lemmatization to normalize the text.
2. **Feature Extraction**: Once the text is preprocessed, features are extracted from it to represent the content in a numerical format that machine learning algorithms can understand. Common feature extraction techniques include:
   * Bag-of-Words (BoW): Representing each document as a vector of word counts or frequencies.
   * TF-IDF (Term Frequency-Inverse Document Frequency): Assigning weights to words based on their frequency in the document and across the entire corpus.
   * Word Embeddings: Representing words as dense, low-dimensional vectors learned from large text corpora using techniques like Word2Vec, GloVe, or FastText.
3. **Model Training**: Text categorization models are trained on a labeled dataset, where each document is associated with one or more predefined categories. Common machine learning algorithms used for text categorization include:
   * Naive Bayes Classifier: Based on Bayes' theorem, assumes independence between features.
   * Support Vector Machines (SVM): Learns a hyperplane that separates documents into different categories.
   * Logistic Regression: Predicts the probability of each category for a given document.
   * Neural Networks: Deep learning models such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) can also be used for text categorization.
4. **Model Evaluation**: Once the model is trained, it is evaluated on a separate test dataset to assess its performance. Common evaluation metrics for text categorization include accuracy, precision, recall, F1-score, and area under the ROC curve (AUC).
5. **Hyperparameter Tuning and Validation**: The performance of the text categorization model can be further improved by tuning its hyperparameters, such as regularization strength, learning rate, or the number of hidden layers in a neural network. This tuning is typically done using techniques like cross-validation on the training dataset.
6. **Inference**: After training and validation, the text categorization model can be used to classify new, unseen documents into predefined categories.
7. **Model Deployment**: Once the model is trained and evaluated, it can be deployed in real-world applications where it automatically categorizes incoming textual data into relevant categories.

****

**A close-up of a paper

Description automatically generated**