Name: Vineet Chotaliya **SAP ID:** 60009210051

Q1. Explain the stages of Natural Language Processing

Natural Language Processing (NLP) is a field of artificial intelligence (AI) and computational linguistics that focuses on the interaction between computers and humans through natural language.It is the technology that is used by machines to understand, analyze, manipulate, and interpret human's languages.

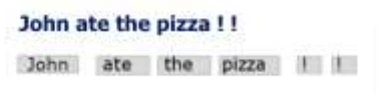
Stages in NLP:

1. Morphological Analysis (word level)

Morphological analysis is a crucial stage in NPL that focuses on the study and processing of the internal structure of words, particularly their morphemes—the smallest units of meaning in a language.

The following are the key components in morphological analysis:

a. **Tokenization:**
   - Morphological analysis typically begins with tokenization, where the input text is segmented into individual tokens, usually words or subwords.
   - Tokenization is essential for isolating units of analysis and facilitating subsequent morphological processing.



John ate the pizza ! !

John	ate	the	pizza	!	!

b. **Stop Word Removal:**
   - Removing the words that occur commonly across the documents, in general articles and pronouns are considered as stop words.

   Ex: “John ate the pizza !!” becomes “John ate pizza”

c. **Stemming:**
   - Stemming is a process in which words are reduced to their root or base form, known as the stem, by removing affixes such as prefixes and suffixes.

   John -> John
   Ate -> eat
   Pizza -> pizza

d. **Lemmatization:**
   - Lemmatization is similar to stemming but involves reducing words to their canonical or dictionary form, known as the lemma.
   - Unlike stemming, lemmatization considers the context and morphological features of words to generate accurate lemmas.

   playing,played, plays -> play

e. **Parts of Speech tagging (POS)**
   - Assigning grammatical categories or parts of speech (e.g., noun, verb, adjective) to individual words in a sentence.
   - POS tagging relies on morphological features such as word endings and prefixes to determine the grammatical role of words in context.

   John - noun, ate - verb, pizza - noun

f. **N-gram language model**

   Continuous sequence of N items/words from a given sample text.

2. Syntactic Analysis

- Syntactic analysis involves the parsing of sentences to identify the grammatical structure and relationships between words.
- Checking the grammatical syntax of a sentence to understand its meaning.
- It aims to determine how words combine to form phrases and sentences according to the rules of a language's grammar.

Ex: “John ate the apple” is correct
“Ate the john apple” is syntactically incorrect

3. Semantic Analysis

- Semantic analysis aims to derive the meaning of text by considering the context, relationships between words, and overall semantics.
- It goes beyond syntactic analysis (which focuses on grammar and sentence structure) to understand the deeper meaning conveyed by language.

“She drank some milk” is both syntactically and semantically correct
“She drank some books” is syntactically correct but semantically incorrect

4. Discourse Analysis

- Discourse analysis is the study of how language is used in communication beyond the level of individual sentences.
- It examines the structure, organization, and flow of text to understand how meaning is constructed and conveyed across larger units of discourse.
- Resolving the references in context of the sample text

a. “Monkeys eats banana when they wake up”
Who are they in this sentence ? Monkeys
b. “Monkeys eat bananas when they are ripe”
Who are they in this sentence ? Bananas

5. Pragmatic Analysis

- Pragmatic analysis in NLP involves interpreting language beyond its literal meaning and considering the pragmatic aspects of communication, such as speaker intentions, implied meaning, and contextual factors.
- It aims to understand how language is used in real-world situations to convey meaning effectively.

“Close the door !” - Order
“Can you please close the door ?” - Request

Q2. “Natural Language is ambiguous”. Justify with the help of suitable examples.

- Natural language is indeed ambiguous, and this ambiguity arises from various sources such as lexical, syntactic, semantic, discourse, and pragmatic ambiguities.
- An input is said to be ambiguous if there are multiple alternative linguistics that can be built for it. A word or sentence may have multiple meanings.

1. Lexical Ambiguity:

- Lexical ambiguity arises when a word has multiple meanings.
- Can be resolved using POS tagging.

Book - Noun - Textbook/Novel
Book - Verb - Booking tickets

2. Syntactic Ambiguity:

- Syntactic ambiguity occurs when the structure or arrangement of words in a sentence can lead to multiple interpretations.

"I saw the man with the telescope." This sentence could mean that the speaker saw a man who had a telescope or that the speaker used a telescope to see the man.

3. Semantic Ambiguity:

- Semantic ambiguity arises when the meaning of a word, phrase, or sentence is unclear due to multiple interpretations.
- A single sentence can have multiple meanings.

"The chicken is ready to eat." This could mean the chicken is prepared for consumption or that the chicken is prepared to consume something else.

4. Anaphoric Ambiguity

- Anaphoric ambiguity occurs when a pronoun or noun phrase refers back to a previously mentioned word or phrase, but it's unclear what exactly it is referring to.
- Anaphora - When we make use of pronouns instead of repeating nouns again and again in a sentence.

“Monkeys eats banana when they wake up”
Who are they in this sentence ? Monkeys
“Monkeys eat bananas when they are ripe”
Who are they in this sentence ? Bananas

5. Pragmatic Ambiguity:

- Pragmatic ambiguity concerns the intentions or implications behind the language used.

“You are late”. This sentence can be informative when you’re making someone aware that they are late or criticizing someone because they are late.

Thus from the above types of ambiguities and different examples it is very evident that the Natural language in which humans communicate is ambiguous.

Q3. You have a dataset containing user reviews for a product, but it includes a significant amount of irrelevant symbols and special characters. How would you approach cleaning the text data to ensure meaningful analysis and sentiment extraction?

- Remove special characters and symbols using regex, i.e. remove the characters that are not alphabets or numbers.
- Lowercase conversion to ensure consistency.
- Tokenization.
- Stop word removal.
- Lemmatization.
- Join the tokens back into a single string which marks the end of the cleaning pipeline.

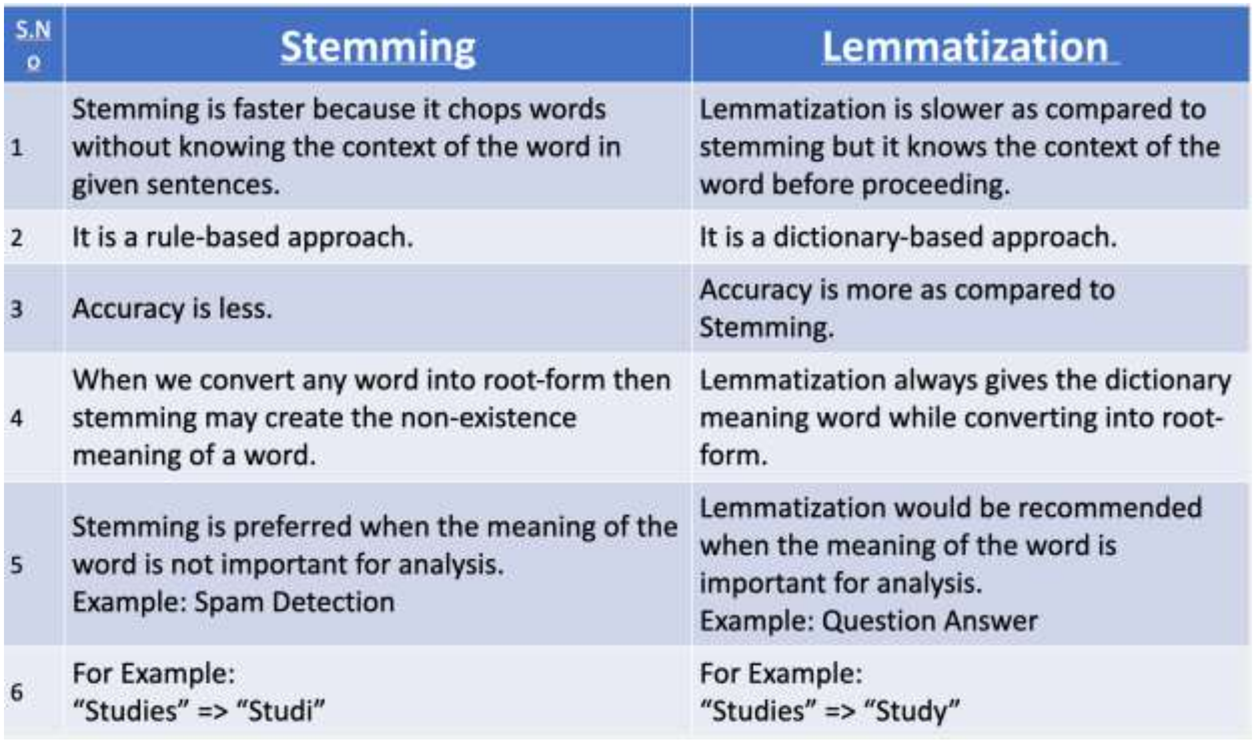
Q4. In a project involving customer support ticket analysis, you notice variations of words such as "help," "helping," and "helped." How would you decide between stemming and lemmatization to normalize the text data, and what considerations should be taken into account?

In the context of customer support ticket analysis, deciding between stemming and lemmatization to normalize text data requires considering several factors:

- **Linguistic Accuracy:**
    - Lemmatization tends to be more linguistically accurate compared to stemming.
    - Stemming may often result in chopping off the end of words to remove suffixes, which can sometimes lead to non-words or words that are not semantically related to the original word.
    - Lemmatization, on the other hand, aims to return the base or dictionary form of a word (the lemma), which is usually a proper word. In your case, if maintaining linguistic accuracy is crucial, lemmatization might be preferred.
- **Word Meaning Preservation:**
    - Stemming can sometimes lead to the loss of the original word's meaning since it operates on a heuristic algorithm to remove affixes.
    - Lemmatization, however, aims to return the base form of the word, preserving its meaning.
    - For customer support ticket analysis where understanding the context of the customer's query or issue is important, lemmatization might be more suitable.
- **Computational Complexity:**
    - Lemmatization is generally more computationally intensive compared to stemming because it requires access to a dictionary or vocabulary and understanding of the word's context to determine its lemma.
    - Stemming, being a simpler process, is usually faster and less resource-intensive.
    - Therefore, if computational resources are limited, stemming might be preferred.
- **Application-specific Requirements:**
    - Consider the specific requirements of your application. If the goal is to simplify text for tasks such as clustering or classification, stemming might be sufficient and more computationally efficient.
    - However, if the analysis requires a deeper understanding of the text and precise semantic relationships between words, lemmatization might be necessary.
- **Language Considerations:**
    - The effectiveness of stemming and lemmatization can vary depending on the language of the text.
    - Some languages have more complex morphology than others, making lemmatization more beneficial. Consider the languages present in your customer support tickets and evaluate which normalization technique works best for each language.

In summary, if linguistic accuracy and maintaining the original word's meaning are essential for your customer support ticket analysis project, and computational resources allow, lemmatization might be preferred. However, if you prioritize computational efficiency or if the analysis task does not require a deep understanding of word meanings, stemming could be a suitable choice. It's often a good practice to experiment with both techniques and evaluate their effectiveness based on your specific requirements and constraints.

Q5. Differentiate between the stemming and lemmatization.



S.No	Stemming	Lemmatization
1	Stemming is faster because it chops words without knowing the context of the word in given sentences.	Lemmatization is slower as compared to stemming but it knows the context of the word before proceeding.
2	It is a rule-based approach.	It is a dictionary-based approach.
3	Accuracy is less.	Accuracy is more as compared to Stemming.
4	When we convert any word into root-form then stemming may create the non-existence meaning of a word.	Lemmatization always gives the dictionary meaning word while converting into root-form.
5	Stemming is preferred when the meaning of the word is not important for analysis. Example: Spam Detection	Lemmatization would be recommended when the meaning of the word is important for analysis. Example: Question Answer
6	For Example: "Studies" => "Studi"	For Example: "Studies" => "Study"

Q6. Explain the various types of ambiguities in NLP.
Refer to Q2.

Q7.Identify the type of ambiguity present in the following sentences.
A. I saw the girl with the binocular. **Syntactic Ambiguity**
B. Rahul loves his cat and Dipesh does too. **Semantic Ambiguity**
C. Monkeys Eat Banana, when they Wake up. **Anaphoric Ambiguity**
D. Monkeys eat Banana, when they are ripe. **Anaphoric Ambiguity**
E. You are Late. **Pragmatic Ambiguity**

Q8. Explain Challenges in NLP.
1. Elongated Words:
- Some users elongate words for emphasis or stylistic reasons (e.g., "sooo" instead of "so" or "loooove" instead of "love").
- Handling such variations can be challenging as they may alter the perceived sentiment or intensity of the text.

2. Shortcuts and Abbreviations:

- Sentences often contain shortcuts, acronyms, and abbreviations (e.g., "lol," "btw," "IDK") in informal communication.
- Deciphering these shortcuts and mapping them to their full forms or meanings is essential for accurate understanding.

3. Emojis and Emoticons:

- Emojis and emoticons convey non-verbal cues such as emotions, tone, and intentions in text-based communication.
- However, interpreting emojis accurately can be challenging due to their subjective nature and the wide range of possible interpretations.

4. Mixed Use of Languages:

- Multilingual communication is common, with users frequently mixing multiple languages or dialects within a single sentence or conversation.
- Handling code-switching and understanding the context for each language switch presents a significant challenge for NLP systems.
- “I liked that movie a lot, Salman Khan ka acting was mast.”

5. Ellipsis:

- Ellipsis, or the omission of words or phrases that are understood from context, is common in informal communication (e.g., "Going to the store, want anything?").
- Understanding the intended meaning and filling in the missing information accurately is challenging for NLP models.

6.Punctuational Ambiguity:

- Punctuation can significantly alter the meaning of a sentence (e.g., "Let's eat, Grandma" vs. "Let's eat Grandma").
- Resolving punctuational ambiguity and interpreting the intended meaning correctly is crucial for accurate language understanding.

7. Irony and Sarcasm:

- Detecting and understanding irony, sarcasm, and other forms of figurative language requires recognizing subtle linguistic cues, tone, and context.
- Interpreting such nuances accurately is challenging but essential for understanding the intended meaning of text.

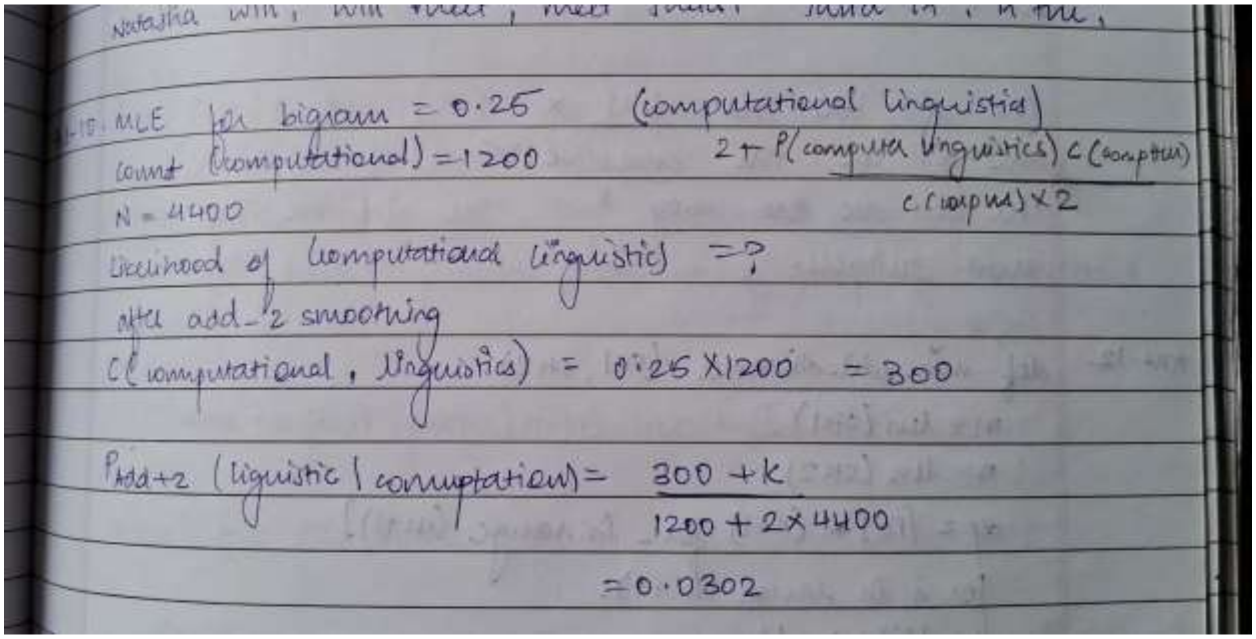
8. Slang and Informal Language:

- Slang terms, colloquialisms, and informal language are prevalent in online communication but may not be present in formal language resources or dictionaries.
- Understanding and interpreting slang terms accurately is challenging, as they often have context-specific meanings.

Q9. Consider the following corpus C1 of three sentences. What is the total count of unique bi-grams for which the likelihood will be estimated? Assume we do not perform any pre-processing. (Consider the beginning of sentence and end of sentence tokens, i.e., <s> and </s>. • Julia is visiting the museum • Julia, Grover and Natasha are friends • Zoe and Natasha will meet Julia in the museum

Total 23 bigrams possible out of which **19** unique bigrams.

Q10 Given a corpus C2, the Maximum Likelihood Estimation (MLE) for the bigram “computational linguistics” is 0.25 and the count of occurrence of the word “computational” is 1200. If the vocabulary size is 4400, what is the likelihood of “computational linguistics” after applying add-2 smoothing?



Natasha will, will meet, meet [illegible] [illegible] in, in the,

Q10. MLE for bigram = 0.25 (computational linguistics)

count (computational) = 1200

$$\frac{2 + P(\text{computa linguistics})\, c(\text{computa})}{c(\text{corpus}) \times 2}$$

N = 4400

Likelihood of (computational linguistics) = ?

after add-2 smoothing

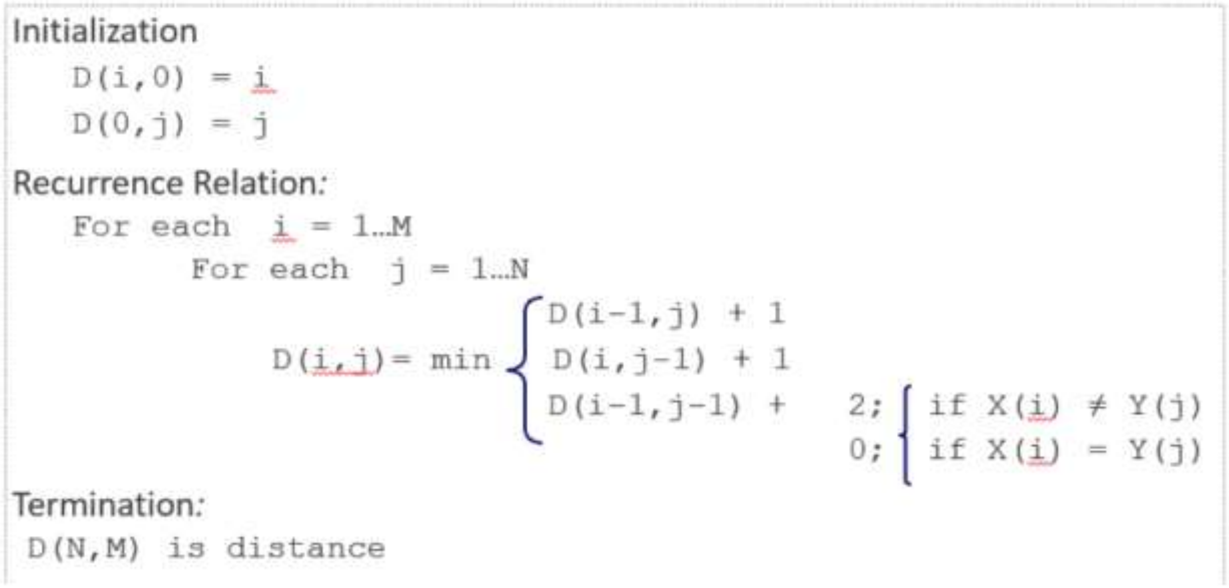
C(computational, linguistics) = 0.25 X 1200 = 300

P_{Add+2} (linguistic | computation) = $\frac{300 + k}{1200 + 2 \times 4400}$

= 0.0302

Q11. Given two strings str1 and str2 and below operations that can be performed on str1. Find minimum number of edits (operations) required to convert ‘str1’ into ‘str2’. Insert - cost (1) Remove- cost (1) Replace -cost (2) Str1= Str2=

Q12.Write the algorithm for finding minimum edit distance using dynamic programming



Initialization

```
    D(i,0) = i
    D(0,j) = j
```

Recurrence Relation:

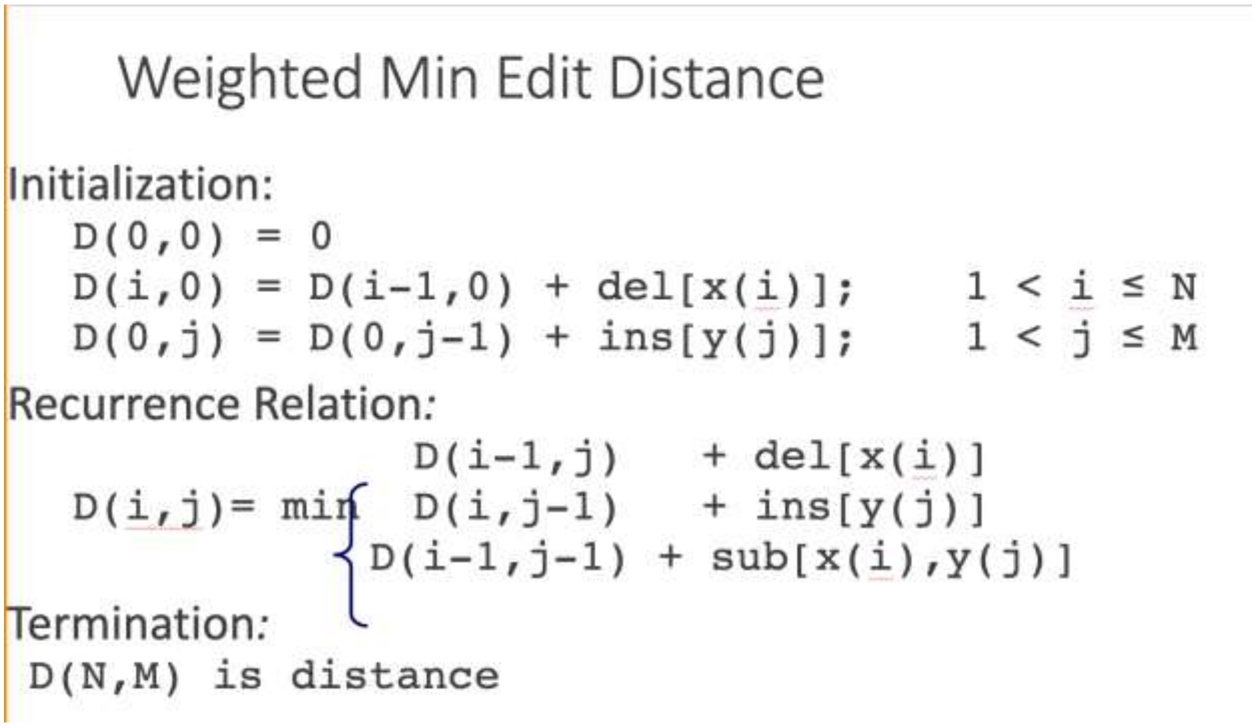
```
    For each  i = 1…M
          For each  j = 1…N
```

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases} \end{cases}$$

Termination:

```
 D(N,M) is distance
```

Q13.Write the algorithm for finding Weighted Minimum Edit Distance.



Weighted Min Edit Distance

Initialization:

```
D(0,0) = 0
D(i,0) = D(i-1,0) + del[x(i)];     1 < i ≤ N
D(0,j) = D(0,j-1) + ins[y(j)];     1 < j ≤ M
```

Recurrence Relation:

$$D(i,j) = \min \begin{cases} D(i-1,j) + \text{del}[x(i)] \\ D(i,j-1) + \text{ins}[y(j)] \\ D(i-1,j-1) + \text{sub}[x(i),y(j)] \end{cases}$$

Termination:

```
D(N,M) is distance
```

Q14. Explain Evaluating Language Models using perplexity

- The naive approach for evaluating two language models A and B is doing the manual tasks like checking the number of misspelled words corrected by the model, checking the total correct translations and then comparing manually. This extrinsic evaluation.
- The extrinsic evaluation is application dependent.
- Intrinsic evaluation is the one that evaluates the quality of a model independent of the application
- Divide the data into training and testing set, train the model on the train set and see how well it fits the test set
- Perplexity is a such common metric used to evaluate the performance of language models. It quantifies how well a language model predicts a sample of text.
- Perplexity measures how well a probability distribution or a language model predicts a given sample of text. It's defined as the geometric mean of the inverse probability of the test set, normalized by the number of words

- The **perpelexity PP** for a test set W=$w_1w_2 \ldots w_n$ is

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1w_2\ldots w_N)}} = \sqrt[N]{\prod_{i=1}^{N}\frac{1}{P(w_i|w_1\ldots w_{i-1})}} \qquad \text{by chain rule}$$

- The **perpelexity PP** for bigrams:

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

- Lower perplexity values on the test set indicate better generalization performance of the language model i.e. minimizing the perplexity yields maximum probability.
- Perplexity allows for the comparison of different language models or different configurations of the same model.
- While perplexity is a widely used metric, it has some limitations. It assumes that the test set and training set come from the same distribution, which may not always be the case.
- Additionally, perplexity may not always correlate perfectly with real-world performance in downstream tasks like machine translation or text generation.

Q15. Write a short note on Add-1 Smoothing.

- Add-1 smoothing, also known as Laplace smoothing, is a simple technique used to address the problem of zero probabilities in probabilistic models, particularly in the context of language modeling.
- In language modeling, when calculating the probability of a word given its context (e.g., preceding words), it's common to encounter unseen word-context pairs.
- These unseen pairs result in zero probabilities, which can cause issues during calculation and may lead to poor model performance.
- Add-1 smoothing addresses this problem by adding a small constant value (typically 1) to the count of each word-context pair. This additional count ensures that no probability estimate becomes zero, and it redistributes probability mass from seen to unseen events.
- $P(W_i , W_{i-1})$ = [count(W_{i-1},W_i) + k] / [count(W_{i-1}) + kV] becomes
  $P(W_i , W_{i-1})$ = [count(W_{i-1},W_i) + 1] / [count(W_{i-1}) + V] because k=1 for add-1 smoothing
  Where V is the vocabulary size
- Add-1 smoothing has the effect of slightly inflating the probabilities of seen events (word-context pairs that occur in the training data) while ensuring that unseen events are assigned non-zero probabilities.
- While Add-1 smoothing is straightforward and helps mitigate the problem of zero probabilities, it may not be the most sophisticated smoothing technique.
- It can lead to over-smoothing, especially in the presence of a large number of unseen events.
- The choice of the smoothing constant (1 in this case) can impact the performance of the smoothed model and may require tuning.

Q16. Explain Steps in Noisy Channel Model for non-word error.

- The Noisy Channel Model is a widely used framework in natural language processing and information theory, particularly in the context of spell checking and error correction.
- It operates under the assumption that communication involves a noisy channel through which information is transmitted, and errors can occur during this transmission.
- Once we seen an observation 'x' for a misspelled word, the main aim of the model is to find W (which is a correct word from a dictionary having highest probability of being close to x)

$$\begin{aligned}\hat{w} &= \underset{w \in V}{\operatorname{argmax}}\, P(w \mid x) \\ &= \underset{w \in V}{\operatorname{argmax}}\, \frac{P(x \mid w)P(w)}{P(x)} \\ &= \underset{w \in V}{\operatorname{argmax}}\, P(x \mid w)P(w)\end{aligned}$$

Here P(w) is the language model that tells us the probability of the word being w.

P(x | w) is the noisy channel model that tells us the probability of the original text being w given x.

Steps:

1. **Selection for Language Model: I**n this step, a language model is selected or built. A language model estimates the likelihood of different sequences of words occurring in a given language.
2. **Channel Model Probability: T**he channel model defines the probability of observing the input x given the true underlying input w. It captures the likelihood of different types of noise or errors that could occur during transmission. This model helps to estimate the probability of particular errors given the true input.
3. **Computing Error Probability:** Based on the channel model, the probability of different types of errors occurring during transmission is computed.
4. **Calculation of P(x|w):** Using the language model and the error probabilities calculated in the previous steps, the likelihood of observing the input x given the true underlying input w is computed. This involves considering both the prior probability of the input w according to the language model and the likelihood of errors given w according to the channel model.
5. **Calculation for w ′** : Finally, the most probable corrected output w ′ is computed based on the observed input x.
   This is typically done by searching for the input w that maximizes the probability P(w|x), which is proportional to P(x|w)×P(w) according to Bayes' theorem.

Q17. Explain Substitutions and Capture Group in Regular expression with the of example.

A regular expression, often abbreviated as regex or regexp, is a sequence of characters that define a search pattern.

Substitution in Regular Expression:

- Substitutions in regular expressions refer to the process of replacing matched patterns within a string with a specified replacement string.
- This is commonly used in text processing tasks such as find-and-replace operations.

Ex:

Replacing Strings: The substitution operation **s/regex_exp/string/** used in Python and Unix commands like vim allows a string to be characterized (found) by a regular expression and then replace it by another string.

- s/colour/color
  - colour is the regular expression pattern to be matched.
  - color is the replacement string that will replace the matched pattern.

Removing Extra Spaces: Use the sub
stitution operation to remove extra spaces or whitespace characters from a string.

- **s/\s+/ /g**
  - This replaces one or more whitespace characters with a single space.
  - The \s+ pattern matches one or more whitespace characters, and
  - the g flag ensures that the substitution is applied globally throughout the string.

Capture Group in Regular Expression:

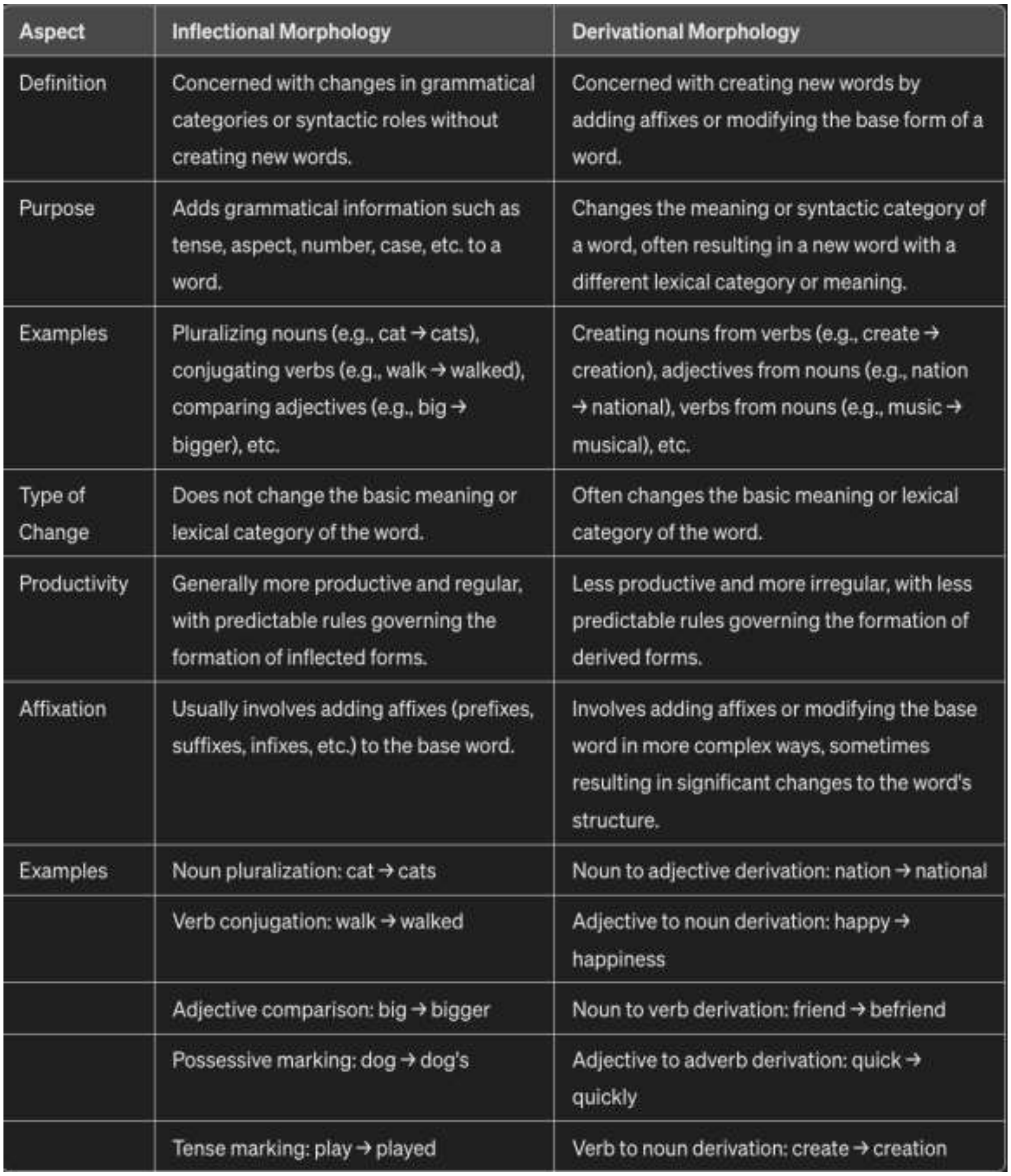
- Capture groups in regular expressions allow you to extract specific parts of a matched pattern.
- They are defined by enclosing the part of the pattern you want to extract within parentheses ().
- When a regular expression pattern is matched against a string, capture groups capture and remember the text matched by the enclosed part of the pattern.

Ex:

Capturing Dates: Suppose we have dates in a document in the format MM/DD/YYYY and we want to extract such dates, we can make use of capture groups in regex.

- **(\d{2})/(\d{2})/(\d{4})**
  - (\d{2}) captures two digits representing the month within the first capture group ().
  - / matches the forward slash between the month and day.
  - (\d{2}) captures two digits representing the day within the second capture group ().
  - / matches the forward slash between the day and year.
  - (\d{4}) captures four digits representing the year within the third capture group ().

Q18. Differentiate between Inflectional morphology and derivational morphology.



Aspect	Inflectional Morphology	Derivational Morphology
Definition	Concerned with changes in grammatical categories or syntactic roles without creating new words.	Concerned with creating new words by adding affixes or modifying the base form of a word.
Purpose	Adds grammatical information such as tense, aspect, number, case, etc. to a word.	Changes the meaning or syntactic category of a word, often resulting in a new word with a different lexical category or meaning.
Examples	Pluralizing nouns (e.g., cat → cats), conjugating verbs (e.g., walk → walked), comparing adjectives (e.g., big → bigger), etc.	Creating nouns from verbs (e.g., create → creation), adjectives from nouns (e.g., nation → national), verbs from nouns (e.g., music → musical), etc.
Type of Change	Does not change the basic meaning or lexical category of the word.	Often changes the basic meaning or lexical category of the word.
Productivity	Generally more productive and regular, with predictable rules governing the formation of inflected forms.	Less productive and more irregular, with less predictable rules governing the formation of derived forms.
Affixation	Usually involves adding affixes (prefixes, suffixes, infixes, etc.) to the base word.	Involves adding affixes or modifying the base word in more complex ways, sometimes resulting in significant changes to the word's structure.
Examples	Noun pluralization: cat → cats	Noun to adjective derivation: nation → national
	Verb conjugation: walk → walked	Adjective to noun derivation: happy → happiness
	Adjective comparison: big → bigger	Noun to verb derivation: friend → befriend
	Possessive marking: dog → dog's	Adjective to adverb derivation: quick → quickly
	Tense marking: play → played	Verb to noun derivation: create → creation

Q19. In a text document, there are references to dates in the format MM/DD/YYYY, but you need to convert them to DD-MM-YYYY. How would you utilize capture groups in regular expressions to transform the date format effectively?

Step1 - Capture the dates in the MM/DD/YYYY format:

- **(\d{2})/(\d{2})/(\d{4})**
    - (\d{2}) captures two digits representing the month within the first capture group ().
    - / matches the forward slash between the month and day.

    - (\d{2}) captures two digits representing the day within the second capture group ().
    - / matches the forward slash between the day and year.
    - (\d{4}) captures four digits representing the year within the third capture group ().

Once the regular expression pattern is matched against the text document, the capture groups will extract the month, day, and year components of each date.

Step2 - Modify the date format:

- **\2-\1-\3**
    - \2: This refers to the second captured group, which represents the day.
    - -: This inserts the hyphen separator.
    - \1: This refers to the first captured group, which represents the month.
    - -: This inserts the hyphen separator.
    - \3: This refers to the third captured group, which represents the year.

Python:
```
pattern = r'(\d{2})/(\d{2})/(\d{4})'
transformed_text = re.sub(pattern, r'\2-\1-\3', text)
```

Q20. Write the Regular Expressions for the following
I. Write the regular expression that matches a string that has an a followed by zero or more b's
ab*

II. Write the regular expression for Regular expression of strings begin and end with 110
^110.*110$

III. Write the regular expression for the language L = {a, aa,aaa, ...}
a+

IV. Write the regular expression for the language accepting all the string which are starting with 1 and ending with 0, over ∑ = {0, 1}.
1(0+1)*0

Q21. How to Calculate the value m in porter stemmer algorithm? Find the value of m for the following words. Tree Troubles Recommendation Highlighting Calling

- The Porter Stemmer algorithm is a widely used algorithm for stemming words in the English language.
- Stemming is the process of reducing a word to its base or root form.
- The Porter Stemmer algorithm consists of a series of rules, organized into five phases, each of which addresses different suffix stripping rules.These rules are applied sequentially to a word until no further changes can be made.
- In the Porter Stemmer algorithm, the variable m represents the measure of the "stem" of a word. It measures the number of consonant sequences before the first vowel sequence in the word.

- To calculate the value of m, we need to calculate the number of vowel-consonant sequences in the given word.
- Steps:
    - Identify the consonant sequences (C) and vowel sequences (V) in the word.
    - Measure the number of vowel-consonant sequences and increase count for each successful occurrence..
    - Assign this count to m.

Tree -> CCVV -> Value of m = 0
Troubles -> CCVVCCVC -> Value of m = 2
Recommendation -> CVCVCCVCCVCVVC -> Value of m = 5
Highlighting -> CVCCCVCCCVCC -> Value of m = 3
Calling -> CVCCVCC -> Value of m = 2

Q22. Justify "POS Tagging is disambiguation task "with the help of suitable example.

- Part-of-speech (POS) tagging assigns a grammatical category (e.g., noun, verb) to each word in a sentence, facilitating precise understanding and analysis of text.
- Part-of-speech (POS) tagging is indeed a disambiguation task because it involves determining the correct part of speech (e.g., noun, verb, adjective) for each word in a given text, which can often be ambiguous based solely on the word itself.
- In natural language the words are ambiguous almost all the time i.e. a single word can have multiple parts of speech associated with it.
- The goal of POS tagging is to assign appropriate 'Parts of Speech' to the words/tokens in the input sentence.

Here's how POS tagging helps disambiguate words:
Noun vs Verb:

- "Bank" can be a noun referring to a financial institution ("I went to the bank").
- It can also be a verb indicating an action ("The boat began to bank").
- POS tagging determines whether "bank" is functioning as a noun or a verb in the given context.

In this example, POS tagging helps disambiguate the word "bank" by identifying its part of speech as a noun, providing clarity about its function in the sentence. Without POS tagging, it might be unclear whether "bank" refers to a financial institution or the action of tilting. This demonstrates how POS tagging is essential for resolving ambiguity in natural language processing tasks.

Q23. Explain Rule Based POS Tagging with the help of an example.

- Rule-based POS tagging is a method used in natural language processing to assign parts-of-speech (POS) tags to words in a text based on predefined rules.
- These rules are typically crafted by linguistic experts and are designed to identify patterns in language usage to determine the correct POS tag for each word.
- Make use of a dictionary or lexicon for getting all the possible tags in order to tag it with the given token/word/sub word.
- If any word has more than one tag associated with it then it makes use of default handwritten rules to resolve the conflicts.

- Disambiguation is performed by analyzing the linguistic features and also taking into consideration the previous and the next word in the sentence.
- Rule based tagging has a 2 stage architecture:

Stage1: Assigning POS tags to the words.

- Starts with the dictionary and assigns all the possible POS to each of the words in the sentence.

Stage2: Resolving Conflicts.

- For the cases where a single word has more than one POS tag associated with it we make use of the rule based approach to resolve the conflicts.
- Stop the process when the word receives one unique tag corresponding to it.
- The rules can either be in context pattern or regex based expressions.
- One such example for a rule is, if the preceding word is an article then the current word must always be a noun.

TAGGIT was the very first rule based tagger which contained around 3300 disambiguation rules.

Ex: I want to read a book

Stage1: I -> Pronoun; want -> Verb; to -> Preposition; read -> Verb ; a -> article;
book -> Noun, Verb

Stage2: Remove ambiguity using rule based approach

Here the word ‘book’ is ambiguous as it has 2 POS tags associated with it.

The word preceding ‘book’ is ‘a’ which is an article thus ‘book’ can never be a verb in this context.

Therefore, book -> Noun

Limitations:

- Hardcoding of the rules is required.
- The manual rules have to be updated based on the changes happening in the language.
- Need to check the lexicon/dictionary everytime which is not an efficient approach.
- The team to build the rules need to be language experts for efficient rules.

Q24. Explain Components of Hidden Markov Models

- Hidden Markov Models (HMMs) are probabilistic sequence models used to describe sequences of observable events, where the underlying system that generates these events is assumed to be a Markov process with unobserved (hidden) states.
- Given a sequence of units/lexemes/words/tokens it calculates the probability distribution over the possible sequences and then chooses the best possible sequence.

Assumptions of HMM:

- An assumption that HMM’s follow the Markov property is made. The Markov property states that the probability of the (n+1)th step is only and only dependent on nth step and not on the complete sequence that came before the nth step. This is also referred to as the property of memorylessness.

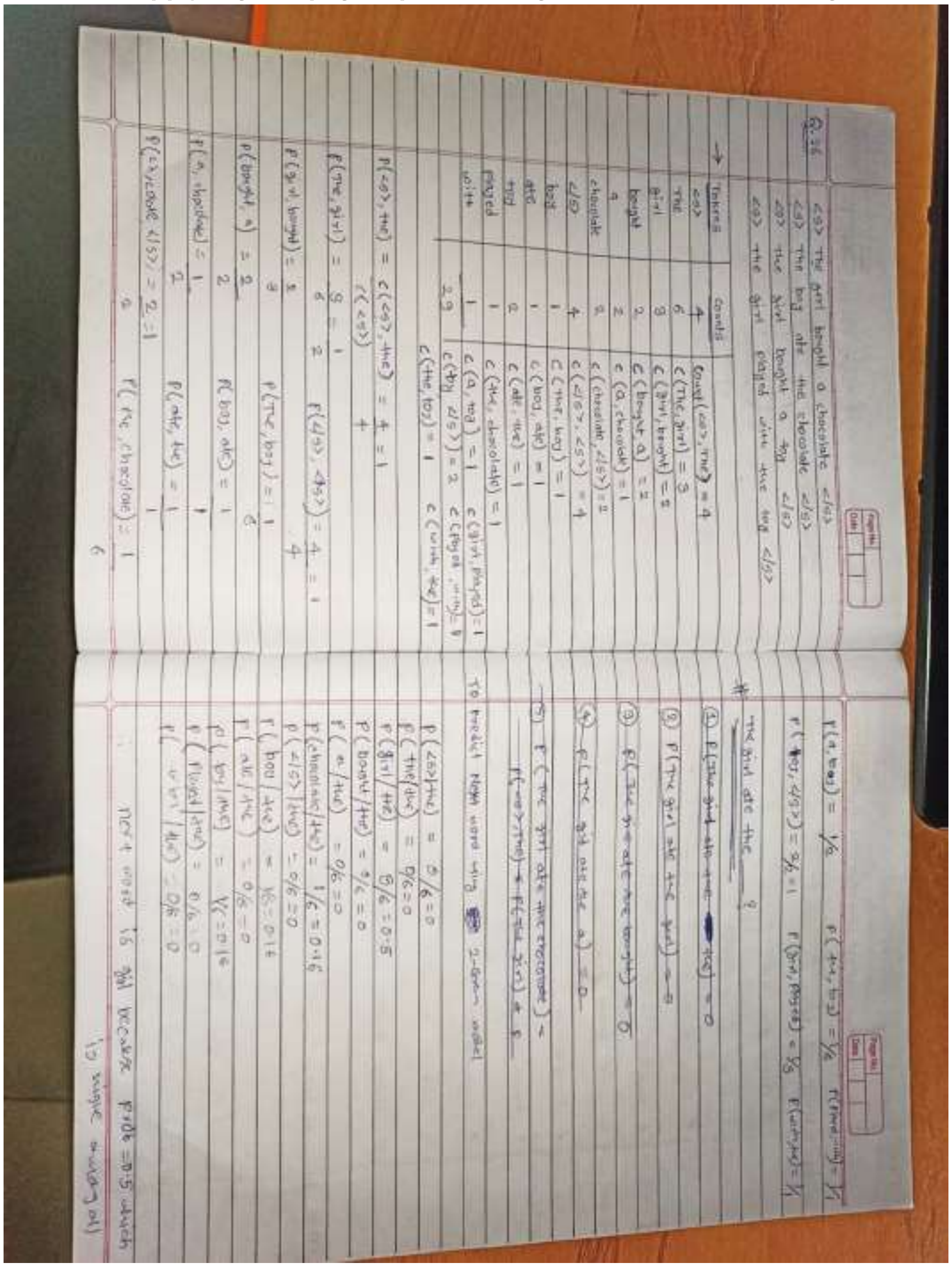
Components of HMM:

1. **Set of States:** Contains a set of all the N possible states Q= q1,q2,q3,....qn which are present in the model.
2. **State Space**
   Hidden States (S): These are the unobservable states that the system moves through over time. Each state represents a particular situation or condition of the system
   Observable States (O): These are the states that emit observable symbols or measurements. Observable states are directly accessible or measurable.
3. **State Transition Matrix:**This matrix defines the probabilities of transitioning from one hidden state to another.
   A = a11,a12,......aij,.....ann.
   Where each a_{ij} represented the probability of moving from state 'i' to state 'j'.
   Also the SUM(all a_{ij}) = 1
4. **Observation Sequence:** These are the observations that are emitted by the Hidden states over time. O = o1o2….oT. Where each o_i is drawn from the Vocabulary V = v1,v2,...vn
5. **Emission Matrix/ Observation Likelihoods:** This matrix defines the probabilities of emitting each observable symbol from each hidden state. B = $b_i o_t$.
   Represents the probability of an observation o_t being generated from a state q_i
6. **Initial Probability Distribution (π):** This vector represents the probability distribution of the system being in each hidden state at the initial time step.
   Here π_i is the probability that the Markov chain will start in state i.
   When a state say 'j' has $\pi_j = 0$ it means that state j cannot be the initial state.

Q25. Write Algorithm for Viterbi HMM.

Initialization: Initialize the trellis (Viterbi matrix) with dimensions (number of states) x (number of observations). Set the initial probabilities for each state based on the start probabilities of the HMM. Set the initial entry in the trellis as the logarithm of the initial probabilities. Recursion: For each observation in the sequence: For each state: Calculate the score for transitioning to the current state from every possible previous state and multiplying it by the emission probability of the observation given the current state. Select the maximum score among these possibilities. Store the maximum score in the corresponding entry in the trellis. Also, store the index of the state from which the maximum score was achieved (backpointer). Termination: After processing all observations, find the state with the maximum final score in the last column of the trellis. This state represents the most likely last state of the sequence. Backtrack through the trellis using backpointers to find the most likely sequence of states that led to the final state. Output: Return the most likely sequence of hidden states obtained from backtracking.

Q26. Consider the following Corpus <s>The girl bought a chocolate </s> <s> The boy ate the chocolate </s> <s> The girl bought a toy </s> <s> The girl played with the toy</s>. Choose the most correct word, which comes after "The girl ate the ___" in below sentence. Apply 2-gram (bigram) for solving it. Sentence: <s> The girl ate the ___?__



Q.26 <s> The girl bought a chocolate </s>
<s> The boy ate the chocolate </s>
<s> the girl bought a toy </s>
<s> the girl played with the toy </s>

→

Tokens	Counts	
<s>	4	count(<s>, The) = 4
The	6	c(The, girl) = 3
girl	3	c(girl, bought) = 2
bought	2	c(bought, a) = 2
a	2	c(a, chocolate) = 1
chocolate	2	c(chocolate, </s>) = 2
</s>	4	c(</s>, <s>) = 4
boy	1	c(The, boy) = 1
ate	1	c(boy, ate) = 1
toy	2	c(ate, the) = 1
played	1	c(the, chocolate) = 1
with	1	c(a, toy) = 1 c(girl, played) = 1
	29	c(toy </s>) = 2 c(played, with) = 1
		c(the, toy) = 1 c(with, the) = 1

$P(<s>, the) = \frac{c(<s>, the)}{c(<s>)} = \frac{4}{4} = 1$

$P(The, girl) = \frac{3}{6} = \frac{1}{2}$ $P(</s>, <s>) = \frac{4}{4} = 1$

$P(girl, bought) = \frac{2}{3}$ $P(The, boy) = \frac{1}{6}$

$P(bought, a) = \frac{2}{2}$ $P(boy, ate) = \frac{1}{1}$

$P(a, chocolate) = \frac{1}{2}$ $P(ate, the) = \frac{1}{1}$

$P(chocolate </s>) = \frac{2}{2} = 1$ $P(the, chocolate) = \frac{1}{6}$

$P(a, toy) = 1/2$ $P(the, toy) = 1/6$ $P(played, with) = 1/1$

$P(toy, </s>) = 2/2 = 1$ $P(girl, played) = 1/3$ $P(with, the) = 1/1$

The girl ate the ___ ?

① ~~P(The girl ate the [illegible] toy) = 0~~

② ~~P(The girl ate the girl) = 0~~

③ ~~P(The girl ate the bought) = 0~~

④ ~~P(The girl ate the a) = 0~~

~~⑤ P(The girl ate the chocolate) = P(<s>, The) * P(The girl) * P~~

To predict next word using 2-gram model

$P(<s> | the) = 0/6 = 0$
$P(the/the) = 0/6 = 0$
$P(girl/the) = 3/6 = 0.5$
$P(bought/the) = 0/6 = 0$
$P(a/the) = 0/6 = 0$
$P(chocolate/the) = 1/6 = 0.16$
$P(</s> | the) = 0/6 = 0$
$P(boy/the) = 1/6 = 0.16$
$P(ate/the) = 0/6 = 0$
$P(toy/the) = 1/6 = 0.16$
$P(played/the) = 0/6 = 0$
$P(with/the) = 0/6 = 0$

next word is girl because prob = 0.5 which is highest among all