Times asked: 6 times
5 times
4 times
3 times
2 times
1 time
# indicates 5-mark question
% Questions from same topic merged

NLP Question bank

1. Introduction to NLP

1. Explain different stages of NLP.
2. Explain the preprocessing operations in NLP.
3. Differentiate between Syntactic ambiguity and Lexical ambiguity. #
4. Discuss the challenges in various stages of NLP.

2. Word Level Analysis

5. Explain the Porter Stemming algorithm in detail.
6. Explain the concept of N-gram (including Bi-gram) with formula. Describe its use in language modelling and explain how it is applied in spelling correction. %
7. What is the output of Morphological analysis for regular verb, irregular verb, singular noun, plural noun. #
8. Explain and compare inflectional and derivational morphology with an example. %
9. Application of Bigram Model for Word Prediction and Sentence Probability in NLP:

   For following corpus, apply Bi-gram model,

   Training Corpus:

   <s> I am Sam </s>

   <s> Sam I am </s>

   <s> Sam I like </s>

   <s> Sam I do like </s>

   <s> do I like Sam </s>

   i) What is the most probable next word predicted by the model for the following word sequences?

   (a) <s> Sam ... (b) <s> Sam I do ... (c) <s> Sam I am Sam ... (d) <s> do I like ...

   ii) Which of the following sentences is better, i.e., gets a higher probability with this model?

   (e) <s> Sam I do I like </s> (f) <s> Sam I am </s> (g) <s> I do like Sam I am </s>

3. Syntax Analysis

10. Apply Hidden Markov Model (HMM) for POS tagging, compute emission and transition probabilities, and decode using Viterbi algorithm (if asked).
    i. For given corpus, <S> indicates start of the statement and <E> indicates end of the statement. *(Asked twice)*

<S>	Martin	Justin	can	watch	Will	<E>
<S>	Spot	will	watch	Martin	<E>	
<S>	Will	Justin	spot	Martin	<E>	
<S>	Martin	will	pat	Spot	<E>	

N: Noun [Martin, Justin, Will, Spot, Pat]
M: Modal verb [can, will]
V: Verb [watch, spot, pat]
Create Transition Matrix & Emission Probability Matrix
Statement is: "Justin will spot Will"
Apply Hidden Markov Model and do POS tagging for given statements.

    ii. Consider the following corpus:
<s> a/DT dog/NN chases/V a/DT cat/NN </s>
<s> the/DT dog/NN barks/V loudly/RB </s>
<s> a/DT cat/NN runs/V fast/RB </s>
Compute the emission and transition probabilities for a bigram HMM. Also, decode the following sentence using the Viterbi algorithm:
"The cat chases the dog."

11. Explain various challenges in POS tagging.
12. For a given grammar using CYK algorithm parse the statement "The man read this book"
Rules:

S → NP VP	Det → *that* | *this* | *a* | *the*
S → Aux NP VP	Noun → *book* | *flight* | *meal* | *man*
S → VP	Verb → *book* | *include* | *read*
NP → Det NOM	Aux → *does*
NOM → Noun	
NOM → Noun NOM	
VP → Verb	
VP → Verb NP	

13. Explain how Conditional Random Field (CRF) is used for sequence labelling. %

14. Explain Maximum Entropy model for POS tagging and sequence labelling. %
15. What are the limitations of Hidden Markov Model (HMM) and MaxEnt Model for POS Tagging. #
16. Explain rule-based, stochastic and hybrid POS tagging. %

4. Semantic Analysis

17. What is Word Sense Disambiguation (WSD)? #
18. Explain dictionary-based approach (Lesk algorithm) for word sense disambiguation (WSD) with suitable example.
19. Explain Naïve Bayes supervised algorithm for Word sense disambiguation.
20. Explain the semi-supervised Yarowsky algorithm for word sense disambiguation (WSD). #
21. Explain semantic analysis and demonstrate lexical semantic analysis using an example.%
22. Explain with suitable example the following relationships between word meanings: Hyponymy, Hypernymy, Meronymy, Holonymy, Homonymy, Polysemy, Synonymy, Antonymy. %

5. Pragmatic and Discourse Processing

23. Explain reference resolution in detail.
24. Explain Anaphora Resolution using Hobbs algorithm.
25. Explain Anaphora Resolution using Centering algorithm.

6. Applications of NLP

26. Explain Machine translation approaches used in NLP. %
27. Explain the information retrieval system.
28. Explain Question Answering system (QAS) in detail.
29. Explain Text summarization in detail.

	1	2	3	4	5	6
2025 May	15	30	35	20	15	10
2024 Dec	15	30	35	15	15	15
2024 May	5	25	25	25	20	20
2023 Dec	10	25	45	15	15	15
2023 May	20	25	30	10	25	15
2022 Dec	15	40	25	20	10	20
Estimate	**15**	**30**	**35**	**20**	**15**	**15-20**
Total	80	175	195	105	100	95

Asked once:

1. Introduction to NLP

1. Explain generic NLP system. #
2. Explain the applications of NLP. #
3. Explain the ambiguities associated at each level with example for NLP.

2. Word Level Analysis

4. Explain the significance of regular expression in NLP.
5. Illustrate the concept of tokenization and stemming in NLP. #
6. Explain Good Turing Discounting. #
7. Define affixes. Explain the types of affixes. #
8. Describe open class words and closed class words in English with examples. #
9. Explain perplexity of any language model. #
10. Explain the role of FSA in morphological analysis.
11. Explain FSA for nouns and verbs. Also design a Finite State Automata (FSA) for the words of English numbers 1-99.
12. Explain role of FST in morphological parsing with an example. Design FST for regular and plural nouns.

3. Syntax Analysis

13. Explain Shift Reduce Parser in NLP with an example.
14. Explain Hidden Markov Model for POS based tagging.
15. Construct a parse tree for the following sentence using the given CFG rules:
    The tall girl sings.
    Rules:
    S → NP VP NP → Det Adj N | Det N VP → V | V NP Det → "the"
    Adj → "tall" N → "girl" V → "sings"
16. Explain the use of Probabilistic Context Free Grammar (PCFG) in NLP with example.
17. Compare top-down and bottom-up approach of parsing with example.

4. Semantic Analysis

18. Explain Unsupervised method (Hyperlex). #

5. Pragmatic and Discourse Processing

19. Explain following Syntactic and Semantic constraints on Coreference:
    1) Number agreement 2) Person & Case agreement. #
20. Compare and contrast Hobbs algorithm and Centering theory.
21. Illustrate the reference phenomena for solving the pronoun problem.
22. What are five types of referring expressions? Explain with the help of example.
23. Explain three types of referents that complicate the reference resolution problem. #

6. Applications of NLP

24. Explain information retrieval versus Information extraction systems.

NLP Answer bank

# indicates 5-mark question
% Questions from same topic merged

1. Introduction to NLP

1. Explain different stages of NLP.

1. Lexical or Morphological Analysis

- Breaks sentences into smaller units called tokens and finds their base or root forms known as morphemes.
- Includes operations like tokenization, stemming, and lemmatization.

Example: "playing" → "play".

2. Syntactic Analysis (Parsing)

- Checks if the sentence is grammatically correct according to language rules.
- Determines relationships between words and constructs a parse tree.

Example: "The dog chased the cat." follows the correct Subject–Verb–Object order.

3. Semantic Analysis

- Focuses on understanding the meaning of words and sentences in context.
- Resolves ambiguities when a word or phrase has more than one possible meaning.

Example: "bank" may refer to river bank or a financial institution.

4. Discourse Integration

- Ensures coherence and logical connection between multiple sentences in a text.
- Finds links between words like pronouns and the nouns they refer to.

Example: "Raj went home. He was tired." → "He" refers to "Raj".

5. Pragmatic Analysis

- Interprets the intended meaning based on context and speaker's intention.
- Considers how language is used in real-life situations to convey purpose or emotion.

Example: "Can you open the door?" means a request, not a question.

2. Explain the preprocessing operations in NLP.

Preprocessing in NLP is the initial and most important step that involves cleaning and transforming raw text into a structured form that can be easily understood by machines.

1. Tokenization

- Tokenization breaks the text into smaller units called tokens such as words, phrases, or sentences.
- It helps in analyzing each word separately instead of the entire text as one string.

Example: "I hate NLP" → ["I", "hate", "NLP"]

2. Stop Word Removal

- Removes commonly used words like "is", "the", "a" that do not add significant meaning.
- This step reduces data size and focuses only on important words.

Example: "the cat is sleeping" → "cat sleeping"

3. Stemming

- Reduces inflected or derived words to their root form by removing prefixes or suffixes.
- It is a rule-based process and may not always produce actual dictionary words.

Example: "playing", "played" → "play"

4. Lemmatization

- Converts words to their base or dictionary form (lemma) using vocabulary and grammar.
- Unlike stemming, it produces valid words and considers part of speech.

Example: "better" → "good", "am" → "be"

5. Lowercasing

- Converts all characters in text to lowercase to maintain uniformity.
- Prevents treating "Apple" and "apple" as different words.

Example: "Natural Language Processing" → "natural language processing"

6. Removing Punctuation and Special Characters

- Removes unwanted characters such as punctuation marks, digits, or symbols that add noise.

Example: "Hello!!!" → "Hello"

7. Part-of-Speech (POS) Tagging

- Assigns grammatical tags like noun, verb, adjective to each token.
- Useful for syntactic and semantic analysis.

Example: "She sings beautifully." → [She–pronoun, sings–verb, beautifully–adverb]

3. Differentiate between Syntactic ambiguity and Lexical ambiguity. #

Basis	**Syntactic Ambiguity**	**Lexical Ambiguity**
Definition	Occurs when a sentence has more than one meaning due to grammatical structure.	Occurs when a single word has multiple possible meanings.
Type of Ambiguity	Structural ambiguity.	Word-level ambiguity.
Level	Appears at the sentence or phrase level.	Appears at the word level.
Focus	Related to syntax (grammar and arrangement of words).	Related to semantics (meaning of words).
Frequency	Less common and more complex to detect.	More frequent and easier to identify.
Cause	Caused by confusing or unclear sentence structure.	Caused by words that have multiple meanings (polysemy).
Resolution	Resolved using syntactic parsing or grammar rules.	Resolved using context or semantic analysis.
Example	"Visiting relatives can be annoying." → (Relatives who are visiting you / You are visiting your relatives)	"Bank" can mean river bank or a financial institution.

4. Discuss the challenges in various stages of NLP.

1. Lexical / Morphological Analysis

- **Ambiguity in Word Forms:** Many words have multiple morphological forms, making their root detection tough.
- **Unknown Words:** New or rare words may not exist in the dictionary.

2. Syntactic Analysis (Parsing)

- **Structural Ambiguity:** A sentence can have more than one correct structure.
- **Complex Grammar Rules:** Irregular rules and sentence patterns are hard to handle.

3. Semantic Analysis

- **Word Sense Ambiguity:** A single word can have several meanings depending on context (e.g., bank).
- **Context Understanding:** Hard to know the right meaning without full sentence context.

4. Discourse Integration

- **Reference Resolution:** Identifying which noun a pronoun like “he” or “it” refer to.
- **Maintaining Coherence:** Maintaining connection and flow between sentences.

5. Pragmatic Analysis

- **Hidden meaning:** Difficult to catch real intention behind words (like sarcasm or request).
- **Context change:** Meaning may differ with tone, place, or situation.

2. Word Level Analysis

5. Explain the Porter Stemming algorithm in detail.

The Porter Stemming Algorithm is a rule-based process used in Natural Language Processing to reduce words to their root form (called a *stem*).

- Example: “connection,” “connected,” “connecting” → connect
- It removes common suffixes from English words without doing full morphological analysis.

It was developed by Martin Porter in 1980 and is widely used in Information Retrieval tasks like search engines.

Why it is used:

- To treat related words as the same during text processing.
- Reduces vocabulary size for algorithms, improving efficiency.
- Helps match queries with relevant documents.

Working:

The algorithm works in five sequential steps; each applying a set of rules to remove or replace suffixes.

A suffix is removed only if certain conditions are met (like stem length, vowel presence etc)

Step 1: Plural and past tense endings

- Removes common plural (-s, -es) and past/present participle endings (-ed, -ing).
- Example:
  - *agreed* → **agree**
  - *playing* → **play**

Step 2: Replace common suffixes

- Changes longer suffixes like - ational → -ate, -ization → -ize, -izer → -ize.
- Example:
  - *relational* → **relate**
  - *organizer* → **organize**

Step 3: More suffix replacement

- Converts endings like -icate → -ic, -alize → -al, -ness → (remove).
- Example:
  - *electrical* → **electric**
  - *hopefulness* → **hopeful**

Step 4: Remove certain suffixes

- Remove endings like -ance, -ence, -ment if certain conditions are met.
- Example:
  - *persistence* → **persist**
  - *adjustment* → **adjust**

Step 5: Final adjustments

- Removes a final -e if appropriate.
- Example:
  - *probate* → **probat**
  - *rate* → **rate** (unchanged if removal changes meaning too much)

Example:

Word: *generalization*

1. Step 1: No change (doesn't match plural/past tense rules).
2. Step 2: -ization → -ize; rule applies → **generalize.**
3. Step 3: No further change.
4. Step 4: Remove the suffix **-ize** → **general**.
5. Step 5: No change.

Final stem: general

Advantages:

1. **Simple and Efficient** – Uses a fixed set of rules, so it's quick to run.
2. **Widely Used** – Common in search engines, information retrieval, and NLP pipelines.
3. **Reduces Vocabulary Size** – Groups related words together, improving storage and processing.

Limitations:

1. **Over-Stemming** – Different words with different meanings may be reduced to the same stem (universe and university → univers).
2. **Not Context-Aware** – Ignores the meaning of words, only processes by pattern.
3. **Produces Non-Words** – Stems may not be valid English words (conditional → condit).

6. Explain the concept of N-gram (including Bi-gram) with formula. Describe its use in language modelling and explain how it is applied in spelling correction. %

N-gram:

- An N-gram is a sequence of N consecutive words used to predict the next word in a sentence.
- It is used to predict how likely a word is to appear after another in a sentence.
  Example: Unigram (1-word sequence): "I", "like", "apples"

Bigram:

- A Bigram model looks at two words at a time to predict the next one.
- It assumes that each word depends only on the previous word, not the entire sentence.
  Example: Bigram (2-word sequence): "I like", "like apples"

Formula for N-gram Probability:

To estimate the probability of a word sequence:

$$P(w_1, w_2, \ldots, w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-1}, \ldots, w_{i-N+1})$$

For a Bigram (N=2) model:

$$P(w_1, w_2, \ldots, w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-1})$$

Where. w_i is the current word and w_{i-1} is the previous word in the sequence.

Use in Language Modelling

- N-gram models are used to estimate the probability of word sequences and predict the next word in context.
- They help build statistical language models for tasks like speech recognition and machine translation.
- For instance, the model calculates that "I am" is more likely than "I apple," helping predict the next logical word.
- It is also used in text generation, autocomplete, and chatbots, where context-based prediction is required.

Application in Spelling Correction

- Used to detect and correct spelling errors by analyzing the probability of word sequences.
- If a user types "I have a pear of shoes," both "pear" and "pair" are words, but the model corrects it to "pair" since "pair of shoes" has higher probability.
- N-gram models also correct split or merged words, like "thetable" to "the table".
- Widely used in search engines, autocorrect systems, and grammar checkers to suggest the most contextually appropriate correction.

7. What is the output of Morphological analysis for regular verb, irregular verb, singular noun, plural noun. #

Regular Verb: Gives the root form by removing regular suffixes like -ed or -ing.
Example: played → play

Irregular Verb: Identifies the base form using a lexicon lookup instead of simple rules.
Example: went → go

Singular Noun: Gives the same word as it is already in base form.
Example: cat → cat

Plural Noun: Removes plural suffix -s or -es to find the singular base form.
Example: dogs → dog

8. Explain and compare inflectional and derivational morphology with an example. %

1. Inflectional Morphology

Inflectional morphology adds grammatical information to a word like tense, number, person, or degree of comparison — without changing its core meaning or its word class (noun stays a noun, verb stays a verb).
It simply adjusts the word so it "fits" grammatically in a sentence.

Examples:

- *cat* → *cats* (plural)
- *play* → *playing* (present participle verb)
- *big* → *bigger* (comparative adjective)

In all these, the word's role stays the same, only the form changes to match grammar.

2. Derivational Morphology

Derivational morphology creates new words from existing ones by adding prefixes or suffixes, often changing both meaning and sometimes the word class.
It's helps expand vocabulary and form new concepts.

Examples:

- *care* → *careless* (noun → adjective)
- *write* → *rewrite* (verb → verb, meaning changes)
- *govern* → *government* (verb → noun)

Here, both meaning and sometimes grammatical category are altered, giving a different word altogether.

Compare Inflectional and Derivational Morphology (asked once)

Aspect	**Inflectional Morphology**	**Derivational Morphology**
Purpose	Shows grammatical variations	Forms new words
Word Class	Does not change word class	Often changes word class
Meaning	Meaning remains similar	Meaning changes
Position in Word	Usually occurs after derivational morphemes	Occurs before inflectional morphemes
Number of Affixes	Limited and fixed set (e.g., -s, -ed, -ing)	Large and open set (e.g., -ness, -er, un-, re-)
Dependency	Required for grammar	Optional, used for vocabulary building
Example	*play → played*	*teach → teacher*

9. Application of Bigram Model for Word Prediction and Sentence Probability in NLP:

i. For following corpus, apply Bi-gram model,

Training Corpus:

<s> I am Sam </s>

<s> Sam I am </s>

<s> Sam I like </s>

<s> Sam I do like </s>

<s> do I like Sam </s>

i) What is the most probable next word predicted by the model for the following word sequences?

(a) <s> Sam ... (b) <s> Sam I do ... (c) <s> Sam I am Sam ... (d) <s> do I like ...

ii) Which of the following sentences is better, i.e., gets a higher probability with this model?

(e) <s> Sam I do I like </s> (f) <s> Sam I am </s> (g) <s> I do like Sam I am </s>

https://www.youtube.com/watch?v=f0xlSjpIS80

Step 1: Bigram Counts (current word → next word : count)

From <s>: { Sam: 3, I: 1, do: 1 } ; total = 5
From I: { am: 2, like: 2, do: 1 } ; total = 5
From Sam: { I: 3, </s>: 2 } ; total = 5
From do: { I: 1, like: 1 } ; total = 2
From like: { </s>: 2, Sam: 1 } ; total = 3
From am: { Sam: 1, </s>: 1 } ; total = 2

Step 2: Calculate Bigram Probabilities:

$$P(w_i \mid w_{i-1}) = \frac{\text{Count}(w_{i-1}, w_i)}{\text{Count}(w_{i-1})}$$

.....Count(previous, next) ÷ Count(previous)

From <s>:
P(Sam | <s>) = C(<s>, Sam) / C(<s>) = 3/5
P(I | <s>) = C(<s>, I) / C(<s>) = 1/5
P(do | <s>) = C(<s>, do) / C(<s>) = 1/5

From I:
P(am | I) = C(I, am) / C(I) = 2/5
P(like | I) = C(I, like) / C(I) = 2/5
P(do | I) = C(I, do) / C(I) = 1/5

From Sam:
P(I | Sam) = C(Sam, I) / C(Sam) = 3/5
P(</s> | Sam) = C(Sam, </s>) / C(Sam) = 2/5

From do:
P(I | do) = C(do, I) / C(do) = 1/2
P(like | do) = C(do, like) / C(do) = 1/2

From like:
P(</s> | like) = C(like, </s>) / C(like) = 2/3
P(Sam | like) = C(like, Sam) / C(like) = 1/3

From am:
P(Sam | am) = C(am, Sam) / C(am) = 1/2
P(</s> | am) = C(am, </s>) / C(am) = 1/2

i) Most probable next word

(a) <s> Sam … → **I** (P(I | Sam) = 3/5 > 2/5)
(b) <s> Sam I do … → **tie: I or like** (both 1/2)
(c) <s> Sam I am Sam … → **I** (P(I | Sam)= 3/5 > 2/5)
(d) <s> do I like … → **</s> (end)** (P(</s> | like)= 2/3 > 1/3)

ii) Sentence probabilities (product of bigram probabilities)

$P(\text{sentence}) = \prod P(w_i \mid w_{i-1})$ .....(current word | previous word)

(e) <s> Sam I do I like </s>
= P(Sam | <s>) · P(I | Sam) · P(do | I) · P(I | do) · P(like | I) · P(</s> | like)
= (3/5) · (3/5) · (1/5) · (1/2) · (2/5) · (2/3)
= **0.0096**

(f) <s> Sam I am </s>
= P(Sam | <s>) · P(I | Sam) · P(am | I) · P(</s> | am)
= (3/5) · (3/5) · (2/5) · (1/2)
= **0.072 // highest**

(g) <s> I do like Sam I am </s>
= P(I | <s>) · P(do | I) · P(like | do) · P(Sam | like) · P(I | Sam) · P(am | I) · P(</s> | am)
= (1/5) · (1/5) · (1/2) · (1/3) · (3/5) · (2/5) · (1/2)
= **0.0008**

Answer: (f) has the highest probability.

ii. Consider the following corpus:

<s> I tell you to sleep and rest </s>

<s> I would like to sleep for an hour </s>

<s> Sleep helps one to relax </s>

List all possible bigrams. Compute conditional probabilities and predict the next word for the word "to". #

All possible Bigrams from the given corpus

Sentence 1 <s> I tell you to sleep and rest </s>	**Sentence 2 <s> I would like to sleep for an hour </s>**	**Sentence 3 <s> Sleep helps one to relax </s>**
<s> → I	<s> → I	<s> → Sleep
I → tell	I → would	Sleep → helps
tell → you	would → like	helps → one
you → to	like → to	one → to
to → sleep	to → sleep	to → relax
sleep → and	sleep → for	relax → </s>
and → rest	for → an	
rest → </s>	an → hour	
	hour → </s>	

Counts involving "to"

Bigrams starting with "to":

- (to, sleep) = 2
- (to, relax) = 1

Total occurrences of to as previous word:

$$\text{Count(to)} = 2 + 1 = 3$$

Conditional probabilities for words following to

$$P(\text{sleep} \mid \text{to}) = \frac{2}{3} \approx 0.6667$$

$$P(\text{relax} \mid \text{to}) = \frac{1}{3} \approx 0.3333$$

Most probable next word after "to" = sleep (probability 2/3 ≈ 0.667)

3. Syntax Analysis

10. Apply Hidden Markov Model (HMM) for POS tagging, compute emission and transition probabilities, and decode using Viterbi algorithm.

ii. Consider the following corpus:
<s> a/DT dog/NN chases/V a/DT cat/NN </s>
<s> the/DT dog/NN barks/V loudly/RB </s>
<s> a/DT cat/NN runs/V fast/RB </s>
Compute the emission and transition probabilities for a bigram HMM. Also, decode the following sentence using the Viterbi algorithm:
"The cat chases the dog."

1) Counts (from the corpus)

Tag counts (total occurrences):

- <s> = 3
- DT = 4
- NN = 4
- V = 3
- RB = 2
- </s>= 3

Transition counts (observed adjacent tag pairs):

- <s> → DT : 3
- DT → NN : 4
- NN → V : 3
- NN → </s>: 1
- V → DT : 1
- V → RB : 2
- RB → </s>: 2

Emission counts (word | tag):

- DT → a : 3
- DT → the: 1
- NN → dog: 2
- NN → cat: 2
- V → chases: 1
- V → barks : 1
- V → runs : 1
- RB → loudly: 1
- RB → fast : 1

2) Transition probabilities (bigram HMM)

Use $P(t_{curr} \mid t_{prev}) = \dfrac{\text{count}(t_{prev} \to t_{curr})}{\text{count}(t_{prev})}$.

Transition	Fraction	Decimal
P(DT | ⟨s⟩)	3/3 = 1	1.0000
P(NN | DT)	4/4 = 1	1.0000
P(V | NN)	3/4	0.7500
P(⟨/s⟩ | NN)	1/4	0.2500
P(DT | V)	1/3	0.3333
P(RB | V)	2/3	0.6667
P(⟨/s⟩ | RB)	2/2 = 1	1.0000

3) Emission probabilities P(word | tag)

Use $P(w \mid t) = \dfrac{\text{count}(t, w)}{\text{count}(t)}$.

Tag	Word	Fraction	Decimal
DT	a	3/4	0.75
DT	the	1/4	0.25
NN	dog	2/4 = 1/2	0.50
NN	cat	2/4 = 1/2	0.50
V	chases	1/3	0.3333
V	barks	1/3	0.3333
V	runs	1/3	0.3333
RB	loudly	1/2	0.50
RB	fast	1/2	0.50

4) Viterbi decoding

Thanks to Binish

1. We start at <s> (start state)

In our training corpus, <s> always goes to DT (determiner) with probability 1.

Then, we multiply by how likely it is for DT to emit the word "the". That's P(the | DT)=0.25.

→ Score so far: 0.25

2. Next word = "cat"

From DT, the only next tag is NN (noun) with probability 1.

Then we multiply by P(cat | NN)=0.5.

We also multiply by the score from the previous step (0.25).

→ Score: 0.25 × 1 × 0.5 = 0.125

3. Next word = "chases"

From NN, we can go to V (verb) with probability 0.75.

Then we multiply by P(chases | V)=1/3.

We also multiply by the score from step 2 (0.125).

→ Score: 0.125 × 0.75 × 0.333 ≈ 0.03125

4. Next word = "the"

From V, we can go to DT with probability 1/3.

Then we multiply by P(the | DT)=0.25.

We also multiply by the score from step 3 (0.03125).

→ Score: 0.03125 × 0.333×0.25 ≈ 0.002604

5. Next word = "dog"

From DT, we must go to NN with probability 1.

Then we multiply by P(dog | NN)=0.5.

We also multiply by the score from step 4 (0.002604).

→ Score: 0.002604 × 1 × 0.5 ≈ 0.001302

Final Tagged Sentence: The/DT cat/NN chases/V the/DT dog/NN

11. Explain various challenges in POS tagging.

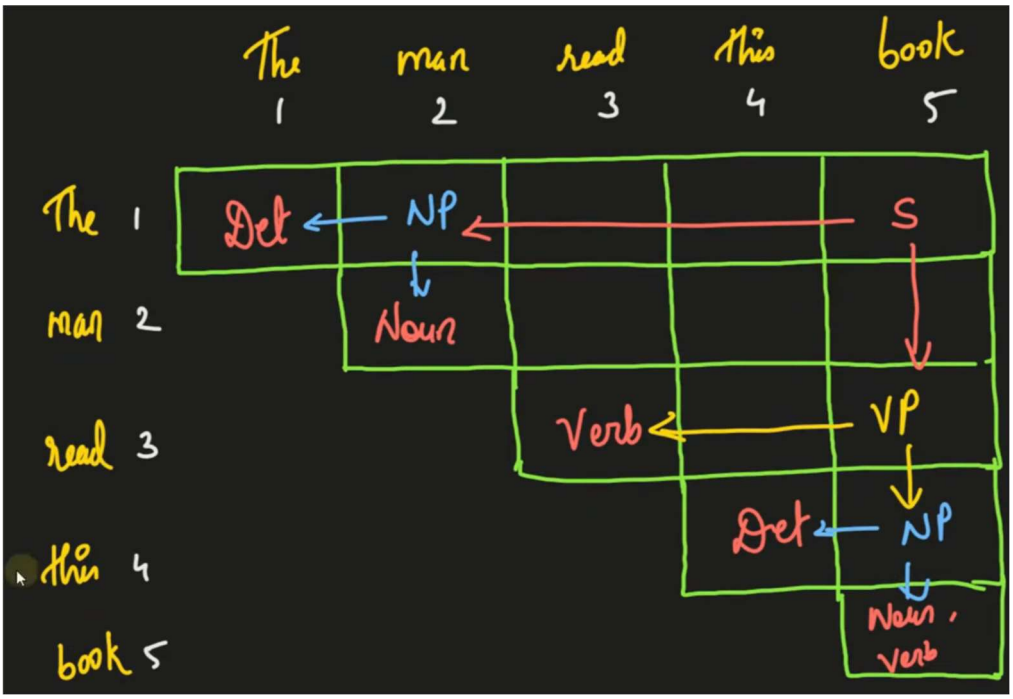
1. **Lexical Ambiguity**
   Many words have multiple possible tags depending on context.
   Example: “book” → noun or verb.
2. **Unknown / Out-of-Vocabulary Words**
   New words, names, slang, or spelling errors are hard to tag correctly when they never appeared in training data.
3. **Context Dependence**
   The correct POS tag often depends on the surrounding words and sentence structure, not just the word itself.
4. **Ambiguous Sentence Structure**
   Sentence patterns can be complex or irregular, confusing rule-based and statistical taggers.
5. **Multi-word Expressions**
   Fixed phrases or idioms behave as a single unit but models treat them separately.
   Example: “in spite of”, “look up”.
6. **Non-standard Grammar/Text:**
   Social media, informal texts or poetry may break standard grammar rules.
7. **Morphological Complexity**
   Languages with rich morphology (Hindi, Marathi, Tamil) have many forms for the same root, making tagging harder.
8. **Data Sparsity & Limited Training Corpora**
   Statistical models need a large amount of labelled data; small or imbalanced corpora reduce accuracy.
9. **Domain Adaptation:**
   Models trained on one domain (news) may perform poorly when tagging texts from other domains (tweets, medical notes).

12. For a given grammar using CYK algorithm parse the statement “The man read this book”

Rules:

S → NP VP	Det → *that* | *this* | *a* | *the*
S → Aux NP VP	Noun → *book* | *flight* | *meal* | *man*
S → VP	Verb → *book* | *include* | *read*
NP → Det NOM	Aux → *does*
NOM → Noun	
NOM → Noun NOM	
VP → Verb	
VP → Verb NP	

https://www.youtube.com/watch?v=ZC6WXy9kX3Q




```
          S
        /   \
      NP     VP
     / \     / \
   Det NOM  Verb  NP
    |   |     |   / \
   the  Noun  read Det NOM
          |          |   |
         man        this Noun
                          |
                         book
```

13. Explain how Conditional Random Field (CRF) is used for sequence labelling. %

Conditional Random Field (CRF) is a probabilistic model used for sequence labelling tasks such as such as POS tagging, Named Entity Recognition (NER), Chunking, etc.
It predicts the most likely sequence of labels for a sentence by looking at both the features of each word and the relationship between neighbouring tags.

Need for CRF: Traditional models like HMM and MaxEnt tag each word individually and often produce inconsistent sequences, while CRFs use the full sentence context to generate a globally consistent and accurate tag sequence.

Working of CRF in Sequence Labelling

CRF assigns labels to each word in a sentence through:

(a) Feature-based Representation

CRFs use a wide range of features, such as:

- Current word, prefixes, suffixes
- Surrounding words (previous/next)
- Previous label → current label transitions

These features allow CRFs to capture both lexical patterns and structural relationships.

(b) Learning Feature Weights

During training, CRF learns the importance (weights) of each feature from a labelled corpus. Features that strongly signal a correct tag are given higher weights, while weak or misleading features receive lower or negative weights.

(c) Sequence Prediction

For a new sentence, CRF computes scores for possible tag sequences.
Using the Viterbi algorithm, it selects the highest-scoring label sequence.

Example:
Sentence: “He plays football.”

Relevant features for “plays” may include:

- Ends with “s”
- Previous word is a pronoun
- Next word is a noun

Considering the features of all words and the transitions between tags, CRF outputs the globally best sequence:
He /PRP plays /VBZ football /NN

Applications:

CRF is widely used in POS tagging, Named Entity Recognition (NER) and Chunking.

14. Explain Maximum Entropy model for POS tagging and sequence labelling. %

Maximum Entropy (MaxEnt) is a probabilistic model used to assign the most likely label to each element in a sequence (e.g., POS tags to words) without making strong assumptions.
It is based on the principle of maximum entropy — among all probability distributions that fit the known data (features), choose the one with the highest entropy (most uniform) to avoid bias.

How MaxEnt Works for Sequence Labelling:

Sequence labelling is the task of assigning a label to each element in a sequence, such as each word in a sentence. Examples include POS tagging, chunking, and named entity recognition.

- **Features:** The model looks at useful clues about the word and its context (e.g., word itself, suffix, previous tag).
- **Training:** It learns which features are important by adjusting weights from the training data.
- **Prediction:** For each word, it calculates the probability of each possible tag and chooses the tag with the highest probability.

POS tagging is one of the most common forms of sequence labelling, where each word is assigned a grammatical category (noun, verb, adjective, etc.).

MaxEnt for POS Tagging:

For each word in a sentence, MaxEnt uses contextual features to decide the most likely POS tag.

Example: Sentence: "He plays football"

Features for “plays” might be:

- Word = “plays”
- suffix = “s”
- Previous tag = PRP (He)

Model computes probabilities:

P(VBZ | features) = 0.85

P(NN | features) = 0.10

P(VBP | features) = 0.05

Assign VBZ as the POS tag.

15. What are the limitations of Hidden Markov Model (HMM) and MaxEnt Model for POS Tagging. #

Limitations of Hidden Markov Model (HMM)

- **Strong independence assumptions:** Assumes each tag depends only on the previous tag and each word depends only on its tag, which is often unrealistic.
- **Weak at handling long-distance dependencies:** Cannot use context beyond one previous word or tag.
- **Needs large, labelled data:** Requires a big, labelled dataset to learn accurate probabilities.
- **Fails with unseen words:** Performs poorly on new or rare words that are not present in training data.

Limitations of Maximum Entropy (MaxEnt) Model

- **Computationally expensive:** Training is slow because it uses many features and heavy optimization.
- **Labels predicted independently:** Tags each word separately, which may result in inconsistent tag sequences.
- **Feature engineering required:** Needs carefully designed features for good performance.
- **Needs large, labelled data:** Depends on sufficient tagged examples to learn reliable weights.

16. Explain rule-based, stochastic and hybrid POS tagging. %

1. Rule-Based POS Tagging

- Uses a large set of hand-written linguistic rules created by experts.
- Tags are assigned based on dictionary lookups (lexicon) and contextual rules like:
    i. "If a word ends with –ing, tag it as VBG."
    ii. "If the previous word is a determiner, tag next word as a noun."
- Works well for structured, formal language but struggles with ambiguous or unseen words.
- **Example:**

  "The dogs bark loudly."
  Rule: After a determiner ("the"), select noun → "dogs/NN".

2. Stochastic (Statistical) POS Tagging

- Uses probability and statistics learned from a tagged corpus.
- Common methods include:
    i. HMM Tagging (transition and emission probabilities)
    ii. N-gram based taggers
- Chooses the most likely tag sequence using probabilities.
- Handles ambiguity better than rule-based, but depends on large, annotated data.
- **Example:**

  If in training data,
  P(NN | "bank") > P(VB | "bank"),

  Then the tagger assigns NN (noun) to "bank".

3. Hybrid POS Tagging

- Combines rule-based and statistical methods to get the advantages of both.
- Rules handle clear, deterministic patterns; statistical models handle ambiguous cases.
- More accurate and robust because it uses linguistic rules and probability together.
- **Example:**

  A rule may check if a word ends in "ing" → verb form, but statistical model confirms whether verb fits context.

4. Semantic Analysis

17. What is Word Sense Disambiguation (WSD)? #

Definition:
Word Sense Disambiguation is the process of determining the correct meaning of a word in context when it has multiple meanings.

Example:

The word “bank” can mean:

- **Financial institution** → “I deposited money in the bank.”
- **Side of a river** → “They walked along the river bank.”

Methods of WSD:

1. **Knowledge-based (Lesk algorithm)** – Uses dictionaries or lexical resources like WordNet.
2. **Supervised learning (Naive Bayes)** – Trains on labelled data to predict word senses.
3. **Semi-supervised (Yarowsky algorithm)** – Starts with a small, labelled dataset and expands iteratively.
4. **Unsupervised learning (Clustering)** – Groups occurrences of words by context to infer senses without labelled data.

Importance:

- Improves machine translation, information retrieval, and text understanding.
- Helps avoid ambiguity in applications like chatbots and question-answering systems.

18. Explain dictionary-based approach (Lesk algorithm) for word sense disambiguation (WSD) with suitable example.

Definition:
The Lesk algorithm is a knowledge-based method for Word Sense Disambiguation that determines the correct meaning of a word by comparing the dictionary definitions (glosses) of its senses with the words in its context. The sense with the most overlapping words with the context is chosen.

Steps of the Algorithm:

1. Identify the target word in a sentence whose sense needs to be disambiguated.
2. Retrieve all possible senses of the target word from a dictionary or lexical database (e.g., WordNet).
3. For each sense, get its gloss (definition) and optionally the glosses of related words (synonyms, hypernyms, hyponyms).
4. Compare the gloss words with the context words surrounding the target word.
5. Count the number of overlapping words between the gloss and the context.
6. Choose the sense with the maximum overlap as the correct meaning.

Example:

Sentence: *"I went to the bank to deposit money."*

- Target word: *"bank"*
- Possible senses:
    1. Financial institution → Gloss: *"a place where money is kept, deposited, or withdrawn"*
    2. Side of a river → Gloss: *"sloping land along the edge of a river"*

Context words: *"deposit, money"*

- Overlap with sense 1: *"money, deposit"* → 2 matches
- Overlap with sense 2: no match → 0 matches

Result: Sense 1 (Financial institution) is chosen.

Advantages:

- Simple and intuitive.
- Uses readily available lexical resources.

Limitations:

- Works only if glosses and context have overlapping words.
- May fail for words with very small glosses.

19. Explain Naïve Bayes supervised algorithm for Word sense disambiguation.

Definition:
Naïve Bayes is a supervised learning algorithm used for WSD. It predicts the correct sense of a word based on the probability of context features, assuming all features are independent.

- Uses Bayes' Theorem:

$$P(s|C) = \frac{P(C|s) \cdot P(s)}{P(C)}$$

where:

- s = sense of the word
- C = context (features like surrounding words, POS tags)

Steps:

1. **Feature Extraction:** Collect features around the ambiguous word, such as context words, POS tags.
2. **Training:** Train the Naïve Bayes model on a dataset where words are labelled with correct senses.
3. **Probability Computation:** For each new occurrence of the ambiguous word, compute P(sense | context) using Bayes' Theorem.
4. **Sense Selection:** Choose the sense with the highest probability as the predicted meaning.

Example:

Target word: *"bank"*
Context words: *"deposit, money"*

Possible senses:

1. Financial institution (Sense 1)
2. Side of a river (Sense 2)

Step 1: Prior probabilities

- From training data:
  - P(Sense 1) = 0.7
  - P(Sense 2) = 0.3

Step 2: Likelihood of context words

Context Word	**P(word | Sense 1)** Financial institution	**P(word | Sense 2)** Side of a river
deposit	0.6	0.1
money	0.7	0.05

Step 3: Compute posterior probabilities using Naïve Bayes

$P(\text{Sense 1} \mid \text{context}) \propto P(\text{Sense 1}) \times P(\text{deposit} \mid \text{Sense 1}) \times P(\text{money} \mid \text{Sense 1})$

$P(\text{Sense 1} \mid \text{context}) = 0.7 \times 0.6 \times 0.7 = 0.294$

$P(\text{Sense 2} \mid \text{context}) \propto P(\text{Sense 2}) \times P(\text{deposit} \mid \text{Sense 2}) \times P(\text{money} \mid \text{Sense 2})$

$P(\text{Sense 2} \mid \text{context}) = 0.3 \times 0.1 \times 0.05 = 0.0015$

Step 4: Choose the sense

- Since $0.294 > 0.0015$, the predicted sense is:
  Financial institution

20. Explain the semi-supervised Yarowsky algorithm for word sense disambiguation (WSD). #

Definition:
The Yarowsky algorithm is a semi-supervised method for Word Sense Disambiguation that starts with a small set of labelled examples and iteratively expands the training data using unlabelled text.

Key Principle: Based on "one sense per collocation" (a word has the same sense in similar local contexts) and "one sense per discourse" (a word tends to keep the same sense throughout a document).

Steps:

1. Begin with a small set of labelled examples.
2. Extract context features like collocations or nearby words.
3. Train a classifier on this small, labelled set.
4. Apply the classifier on unlabelled text, selecting high-confidence predictions.
5. Add these predictions to the training set.
6. Repeat until convergence**.**

Example:

- **Target word:** plant
- Labelled examples: industrial facility → contexts like "factory," "machinery," "production."
- Algorithm finds unlabelled sentences: "The new plant produces cars efficiently."
- Labels them as industrial facility and expands the training set for the next iteration.

21. Explain semantic analysis and demonstrate lexical semantic analysis using an example.%

Semantic analysis:

- Focuses on understanding the meaning of words and sentences in a given context.
- Resolves ambiguities when a word or phrase has more than one possible meaning.
- Plays an important role in machine translation, question answering, and chatbots, where correct meaning must be derived.

Example: “bank” may refer to a river bank or a financial institution depending on the context.

Lexical Semantic Analysis

- Lexical semantics studies word meanings and relationships between words in a language.
- It involves identifying synonyms, antonyms, hypernyms, hyponyms, and homonyms to understand meaning connections.
- It also focuses on Word Sense Disambiguation (WSD) for determining the correct sense of a word in context.
- Tools like WordNet are often used to map these relationships.

Example:

Sentence: “She saw a bat flying near the cave.”

- The word “bat” has two possible meanings:
    1. A flying mammal
    2. A sports equipment
- Lexical semantic analysis uses context words like “flying” and “cave” to infer that “bat” refers to the animal, not the sports equipment.

22. Explain with suitable example the following relationships between word meanings: Homonymy, Polysemy, Synonymy, Antonymy, Hyponymy, Hypernymy, Meronymy, Holonymy. %

1. **Homonymy**
   - **Definition:** Words that share the same spelling or pronunciation but have entirely different and unrelated meanings.
   - **Example:**
     - "Bank" → (1) a financial institution, (2) the side of a river

     The two meanings are unrelated, and context helps to identify the correct meaning in a sentence.
2. **Polysemy**
   - **Definition:** A word that has multiple related meanings that originate from a single core concept.
   - **Example:**
     - "Head" → (1) part of the body, (2) leader of a group

     Both words are connected through the idea of being at the top of something.
3. **Synonymy**
   - **Definition:** Words that have the same or very similar meanings and can often be used interchangeably.
   - **Example:**
     - "Happy" ↔ "Joyful"

     Both words mean the same thing.
4. **Antonymy**
   - **Definition:** Antonyms are words that have opposite meanings.
   - **Example:**
     - "Hot" ↔ "Cold"

     Both words mean the opposite of each other.
5. **Hyponymy**
   - **Definition:** A hyponym is a word that represents a specific member of a broader category. It is a "type of" relationship.
   - **Example:**
     - "Rose" is a hyponym of "flower."

     A rose is a type of flower.

6. **Hypernymy**
   - **Definition:** A hypernym is a word that represents a general category for more specific words. It is a “supercategory” relationship.
   - **Example:**
     - “Vehicle” is a hypernym of “car,” “bus,” “bike.”

     A vehicle is a general category that includes these items.
7. **Meronymy**
   - **Definition:** Meronymy describes a part–whole relationship, where one word refers to a part of something larger.
   - **Example:**
     - “Wheel” is a meronym of “car.”

     A wheel is a part of a car.
8. **Holonymy**
   - **Definition:** A holonym is a word that refers to the **whole** entity, while its related word represents a part of that whole. It is the opposite of meronymy.
   - **Example:**
     - “Car” is a holonym of “wheel.”

     A car is the whole that contains wheels.

5. Pragmatic and Discourse Processing

23. Explain reference resolution in detail.

Reference Resolution is the process of identifying what entity a referring expression (such as a pronoun or noun phrase) refers to in a text.

It is essential for understanding the meaning of sentences because natural language often contains pronouns (he, she, it, they), definite descriptions (the book, that man), or other referring expressions.

Types of References:

1. **Anaphora:** Refers back to something mentioned earlier.
   - Example: Ravi bought a laptop. He is very happy with it. (He → Ravi, it → laptop)
2. **Cataphora:** Refers to something mentioned later in the text.
   - Example: Before he entered the room, Ramesh knocked on the door. (He → Ramesh)
3. **Coreference:** Detects when two or more expressions refer to the same entity.
   - Example: The Prime Minister gave a speech. The leader spoke about reforms. (The Prime Minister = The leader)

Steps in Reference Resolution with an example:

Example text: "Ravi bought a laptop yesterday. He likes it a lot."

- **Mention Detection:** Find all referring words or phrases (like pronouns or noun phrases) in the text.
  In example: *Ravi*, *laptop*, *He*, *it*
- **Candidate Identification:** List all possible entities or mentions that each reference might point to.
  For He → possible candidates: *Ravi*
  For it → possible candidates: *laptop*
- **Feature Matching:** Compare gender, number, and context to check compatibility between the referring expression and candidates.
  He must match a *singular male* → matches *Ravi*
  it must match a *singular object* → matches *laptop*, not *Ravi*
- **Selection:** Choose the most likely entity based on context, grammar, and proximity in the sentence.
  He → most recent male noun = *Ravi*
  it → most recent inanimate noun = *laptop*
- **Linking:** Connect the reference to its antecedent, ensuring meaning is clear in the passage.

Final Result:

Ravi bought a laptop yesterday. He (→ Ravi) likes it (→ laptop) a lot.

24. Explain Anaphora Resolution using Hobbs algorithm.

Anaphora Resolution

Anaphora resolution is the process of identifying the antecedent of a pronoun or noun phrase in discourse. It is essential in NLP for understanding text meaning, connecting pronouns like he, she, it, they to the correct earlier noun phrase.

Hobbs Algorithm:

Hobbs Algorithm is a syntax-based approach to resolve pronouns (like he, she, it, they) in sentences. The main idea is to use the syntactic structure of sentences, represented as parse trees, to find the correct antecedent (the noun a pronoun refers to).

Steps:

1. Start at the NP node containing the pronoun.
2. Go up the tree to the first NP or S node.
3. From there, perform a left-to-right, breadth-first search of the parse tree.
4. Choose the first NP that agrees with the pronoun in number and gender as its antecedent.

Example:
"John went to the park. He played football."

Resolution:

1. Locate the pronoun "He" in the parse tree of Sentence 2.
2. Move up to the first NP or S node.
3. Traverse left-to-right through previous sentences' parse trees:
    - Look for NP nodes: finds "John" in the first sentence.
4. Check number/gender match:
    - "John" is singular, male, matching pronoun "He".
5. Resolve pronoun: "He" → "John"

25. Explain Anaphora Resolution using Centering algorithm.

Centering Algorithm:

Centering Algorithm is a discourse-based approach for resolving pronouns and references in text. It works by keeping track of important entities in each sentence so the system can correctly identify who or what a pronoun refers to.

Cf (Forward-looking Centers): List of entities (NPs) in the current sentence, ranked by salience; candidates for pronoun resolution.

Cb (Backward-looking Center): Most prominent entity from the previous sentence; likely referred to by pronouns.

Steps of Centering Algorithm:

1. **Identify Cf:** List all noun phrases (NPs) in the sentence, rank by salience.
2. **Determine Cb:** Select the most prominent entity from the previous sentence.
3. **Resolve Pronouns:** Pronouns refer to Cb or the highest-ranked Cf.
4. **Update Centers:** Update Cb and Cf for the next sentence.

Example

1. “Alice bought a book.”
2. “She gave it to her friend.”

Resolution:

- Sentence 1 Cf = [Alice, book] → Cb = —
- Sentence 2 Cf = [Alice, book, friend] → Cb = Alice
- Pronoun “She” → refers to Cb (Alice)
- Pronoun “it” → refers to book (next-highest ranked Cf)

Result:

- “She” → Alice
- “It” → book

6. Applications of NLP

26. Explain Machine translation approaches used in NLP. %

Machine Translation is an application of NLP that automatically converts text or speech from one language to another. It understands sentence meaning with context and not just individual words, so the translation is natural rather than literal.

Machine Translation Approaches:

Rule-Based Machine Translation (RBMT)

- Uses hand-written grammar rules, dictionaries, and linguistic knowledge for translation.
- Works by analyzing source language grammar and generating target language output using predefined rules.
- **Advantages:** Accurate grammar, useful for well-structured languages.
- **Limitation:** Hard to create and maintain; struggles with idioms and informal text.
- **Example:** SYSTRAN (early versions)

Statistical Machine Translation (SMT)

- Learns translations from large bilingual corpora using probability and frequency counts.
- Chooses the translation that has the highest statistical likelihood based on training data.
- **Types:** Word-based, Phrase-based, Syntax-based SMT.
- **Advantages:** Learns directly from real data, so it adapts well to natural, everyday language.
- **Limitation:** Often produces phrase-by-phrase translations without deep meaning.
- **Example:** Google Translate (before 2016) used Phrase-Based SMT.

Neural Machine Translation (NMT)

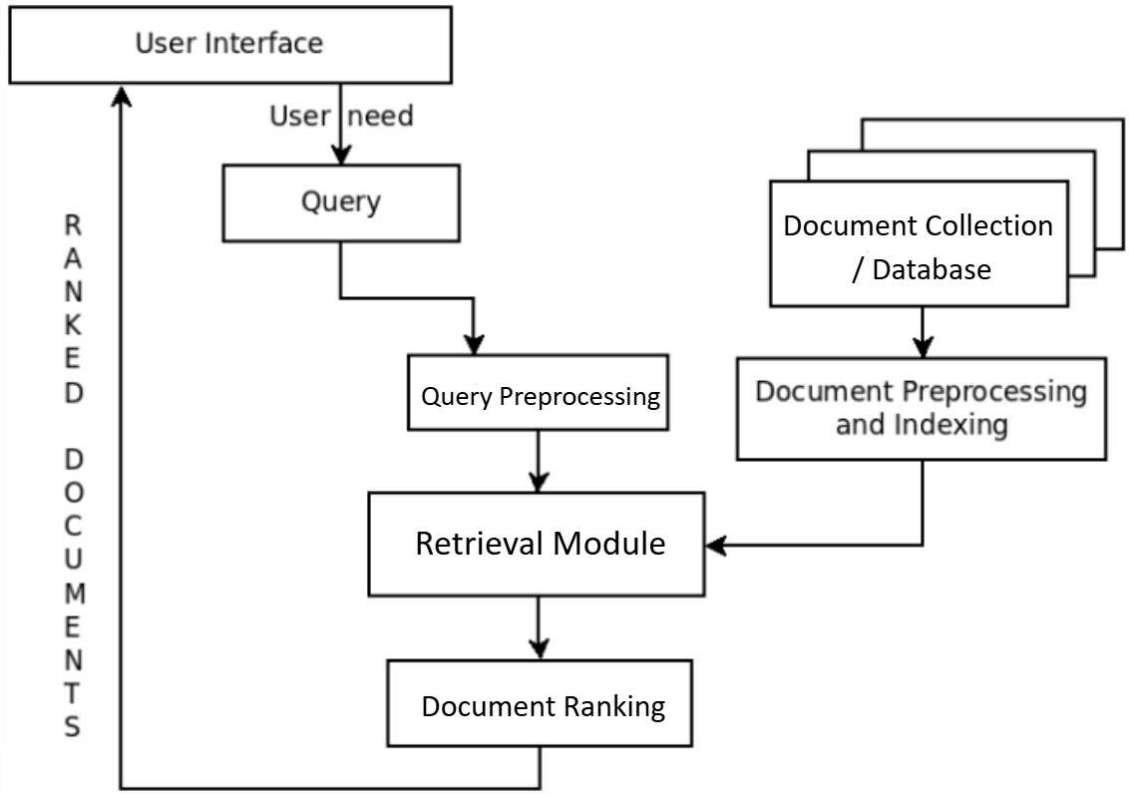
- Uses deep learning models like RNNs, LSTMs, and Transformers to translate entire sentences.
- Understands context and relationships between words better to produce fluent, human-like translations.
- **Advantage:** Produces natural and context-aware translations using deep learning.
- **Limitation**: Needs large datasets and high computational power.
- **Example:** Google Translate (after 2016), Microsoft Translator, and DeepL.

Hybrid Machine Translation

- Combines RBMT + SMT or SMT + NMT to get the strengths of each method.
- Rules improve grammar accuracy, while statistics/neural networks improve fluency and naturalness.
- **Advantage:** Combines strengths of multiple methods to give more accurate and reliable translations.
- **Limitation:** More complex to design and maintain.
- **Example:** Modern SYSTRAN and PROMT.

27. Explain the information retrieval system.

An Information Retrieval System helps users find relevant information from large collections such as documents, websites, databases, and digital libraries. It stores, processes, and retrieves information based on user queries, focusing on returning the most relevant documents rather than exact word matches.



1. Query Input & Preprocessing

- The user enters their information need through the User Interface, which converts it into a text-based query (e.g., “machine learning basics”).
- The system cleans the query using tokenization, stop-word removal, stemming/ lemmatization, and spelling correction to prepare it for matching.

2. Document Collection / Database

- Stores all searchable documents like webpages, PDFs, and articles.
- Serves as the source from which relevant information is retrieved.

3. Document Preprocessing and Indexing

- Documents are cleaned and converted into an index for fast searching.
- Key steps include: Tokenization, Stop-word removal, Stemming and building an inverted index.
- Makes retrieval quick without scanning entire documents.

4. Retrieval Module

- The processed query is compared with the indexed documents.
- Retrieval models may include:

    - Boolean Model
    - Vector Space Model (TF-IDF, cosine similarity)
    - Probabilistic models (BM25)
- The retrieval module finds all documents that match the query.

5. Document Ranking

- Retrieved documents are ranked by relevance using scoring methods:
    - TF-IDF scores
    - Cosine similarity
    - BM25
- The most relevant documents appear at the top.

6. Ranked Documents Returned to User Interface

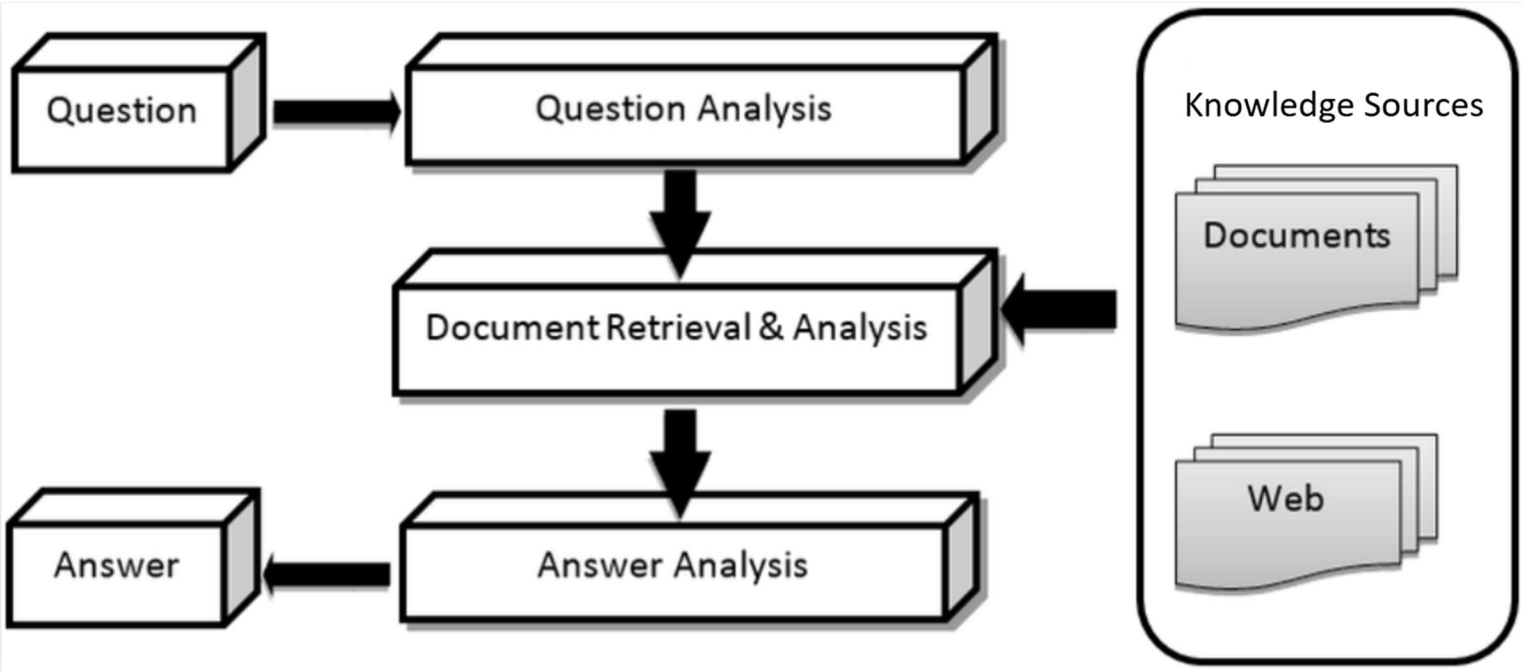
- The ranked list of documents is shown to the user through the UI.
- Results are ordered from most relevant to least relevant.

Example:
Searching "what are basics of machine learning" on Google:

- The query is cleaned through tokenization and stop-word removal.
- Google searches its indexed web pages.
- Documents are ranked using models like TF-IDF and BM25.
- The most relevant results are shown at the top.

28. Explain Question Answering system (QAS) in detail.

- A Question Answering System (QAS) is an NLP-based system that understands questions asked in natural language and gives direct, accurate answers instead of just returning documents.
- It intelligently analyzes the user's query, searches relevant information, and presents the correct answer in clear natural language.



1. **Question Analysis**
   - This module processes the user's question to understand what is being asked.
   - It identifies the question type (who, what, when, where), keywords, and expected answer type (person, place, date).
   - Performs NLP tasks such as tokenization, POS tagging, named entity recognition, and syntactic parsing.
2. **Document Retrieval & Analysis**
   - Searches the Knowledge Sources (documents, web pages) to find relevant information.
   - Uses Information Retrieval techniques like TF-IDF, BM25, or search engines to fetch passages likely to contain the answer.
   - Performs initial filtering and analysis of retrieved text to remove irrelevant data.
3. **Answer Analysis**
   - From the retrieved passages, this module extracts the exact answer.
   - Uses techniques such as pattern matching, semantic similarity, and deep learning models like BERT.
   - Can either select a phrase from text (extractive) or generate an answer (generative).

4. **Knowledge Sources**

- The system searches different sources for answers, such as:
    - Documents (stored articles, manuals, datasets)
    - Web pages (online content)
    - Databases or knowledge bases (if extended)
- These sources form the information base from which the QAS retrieves content.

Example of a Question Answering System (QAS):

User asks: **"What is the capital of Japan?"**

1. **Question Analysis:**
   The system identifies this as a factual *"what"* question and expects a *location* as the answer.
2. **Document Retrieval:**
   It searches stored documents and online sources for information about Japan and its capital.
3. **Answer Analysis:**
   From sentences like *"Tokyo is the capital city of Japan,"* the system extracts the correct phrase.
4. **Final Answer Returned to User:**
   "The capital of Japan is Tokyo."

Applications of QAS

- Virtual assistants (Siri, Alexa, Google Assistant)
- Customer support chatbots
- Search engines

29. Explain Text summarization in detail.

Text summarization is an NLP technique that automatically generates a short and meaningful summary of a longer text while preserving the important information.
It helps users quickly understand the main content without reading the entire document.

Types of Text Summarization:

A) Extractive Summarization

- Selects the most important sentences or phrases directly from the original text without generating new ones.
- Works by scoring sentences using features like:
  - TF-IDF
  - Similarity between sentences
- **Example:** Highlighting key sentences from a news article.

B) Abstractive Summarization

- Generates new sentences that capture the overall meaning of the text, similar to how humans write summaries.
- Uses deep learning models like:
  - Sequence-to-Sequence (Seq2Seq)
  - Transformers (BERT, T5, GPT)
- **Example:** Writing a short paragraph that captures the overall meaning of a long report.

Steps in Text Summarization:

1. **Preprocessing:** Clean the text using tokenization, stop-word removal, stemming/ lemmatization, and sentence splitting.
2. **Understanding & Scoring text:** Identify important sentences or build meaning representation using statistical or semantic methods.
3. **Summary Generation:** Extractive: selects top ranked sentences; Abstractive: generates new, meaningful sentences.
4. **Post-processing:** Arrange sentences properly, remove repetition, and ensure grammatical correctness.

Applications of Text Summarization:

- News summarization
- Summarizing emails, reports, research papers
- Automatic summarization in search engines
- Meeting transcript summarization

Asked once: (will add later)

1. Introduction to NLP

1. Explain generic NLP system. #
2. Explain the applications of NLP. #
3. Explain the ambiguities associated at each level with example for NLP.

2. Word Level Analysis

4. Explain the significance of regular expression in NLP.
5. Illustrate the concept of tokenization and stemming in NLP. #
6. Explain Good Turing Discounting. #
7. Define affixes. Explain the types of affixes. #
8. Describe open class words and closed class words in English with examples. #
9. Explain perplexity of any language model. #
10. Explain the role of FSA in morphological analysis.
11. Explain FSA for nouns and verbs. Also design a Finite State Automata (FSA) for the words of English numbers 1-99.
12. Explain role of FST in morphological parsing with an example. Design FST for regular and plural nouns.

3. Syntax Analysis

13. Explain Shift Reduce Parser in NLP with an example.
14. Explain Hidden Markov Model for POS based tagging.
15. Construct a parse tree for the following sentence using the given CFG rules:
    The tall girl sings.
    Rules:
    S → NP VP NP → Det Adj N | Det N VP → V | V NP Det → "the"
    Adj → "tall" N → "girl" V → "sings"
16. Explain the use of Probabilistic Context Free Grammar (PCFG) in NLP with example.
17. Compare top-down and bottom-up approach of parsing with example.

4. Semantic Analysis

18. Explain Unsupervised method (Hyperlex). #

5. Pragmatic and Discourse Processing

19. Explain following Syntactic and Semantic constraints on Coreference:
    1) Number agreement 2) Person & Case agreement. #
20. Compare and contrast Hobbs algorithm and Centering theory.
21. Illustrate the reference phenomena for solving the pronoun problem.
22. What are five types of referring expressions? Explain with the help of example.
23. Explain three types of referents that complicate the reference resolution problem. #

6. Applications of NLP

24. Explain information retrieval versus Information extraction systems.