Yes. For **your B.Tech 5th-semester NLP (CS24351)**, I'll use the syllabus in your uploaded study material **plus the NLP lecture slides and the actual NLP PYQs you uploaded**, rather than mixing in a generic NLP syllabus.

Your study guide breaks NLP into **53 topics across M1–M5**.

🧠 NLP — COMPLETE MASTER NOTES

CS24351 | M1–M5 | 53 Topics

The goal is not just to memorize definitions. For every topic, use:

Understand → visualize → compress → recall → solve → spaced repetition

And for numerical/algorithmic topics, we'll always do:

Concept → formula → worked example → PYQ → similar question

MODULE I — INTRODUCTION TO NLP

12 topics

Your syllabus covers:

1. Introduction to NLP
2. Applications of NLP
3. Morphology
4. Syntax
5. Semantics
6. Pragmatics
7. Tokenization
8. Lemmatization
9. Stop-word removal
10. Normalization
11. Regular Expressions
12. Language Models overview

---

1. What is NLP?

Natural Language Processing is the area of AI/computer science concerned with enabling computers to process, understand and generate human language.

Two goals:

Scientific goal

Understand language computationally.

Engineering goal

Build systems that process natural language for practical applications.

Your lecture material explicitly distinguishes these two goals.

NLP pipeline

```
Raw Language
     ↓
Preprocessing
     ↓
Lexical Analysis
     ↓
Syntactic Analysis
     ↓
Semantic Analysis
     ↓
Pragmatic/Discourse Interpretation
     ↓
Application
```

Why NLP is difficult

Human language contains:

- ambiguity
- context dependence
- spelling variation
- sarcasm
- idioms
- incomplete information
- multiple meanings
- different structures
- implicit meaning

A classic example:

> **Key Insight**
>
> "I saw the man with the telescope."

Who has the telescope?

The speaker?

The man?

This is **syntactic ambiguity**.

The 2025 CS633 PYQ specifically asks why ambiguity is a key NLP challenge.

---

2. Applications of NLP

Major applications:

Application	NLP task
Search engines	Query understanding
Chatbots	Dialogue/intent
Translation	Machine translation
Gmail	Spam/filtering/autocomplete
Voice assistants	Speech + language
Healthcare	Information extraction
Finance	Sentiment/information extraction
Customer service	Classification/chatbots
News	Text generation
Social media	Sentiment analysis

Your uploaded study guide gives these application examples.

PYQ

Discuss the importance of NLP in healthcare, finance and customer service.

---

3. Morphology

Morphology = study of word structure.

A word can consist of smaller meaningful units called **morphemes**.

Example:

```
unhappiness

un + happy + ness
|     |       |
prefix root   suffix
```

Types

Free morpheme

Can stand alone.

> **Key Insight**
> book

Bound morpheme

Cannot normally stand alone.

> **Key Insight**
> -s, -ed, un-

Inflectional vs Derivational

Inflectional

Changes grammatical form.

```
walk → walked
cat → cats
```

Derivational

Creates a new word/meaning.

```
happy → unhappy
teach → teacher
```

Memory

> **Key Insight**
>
> **Inflection = form**

>

> **Key Insight**
>
> **Derivation = new word**

---

4. Syntax

Syntax concerns how words combine to form grammatical structures.

Example:

```
The boy eats rice.
```

Possible structure:

```
          S
        /   \
      NP     VP
     / \     / \
   Det  N   V   NP
   The boy eats rice
```

This is the foundation of **syntactic parsing**.

---

5. Semantics

Semantics deals with **meaning**.

Example:

> **Key Insight**
> "The dog chased the cat."

Semantic interpretation identifies:

```
Agent = dog
Action = chase
Object = cat
```

---

6. Pragmatics

Pragmatics deals with meaning **in context**.

Example:

> **Key Insight**
> "Can you open the window?"

Literal meaning:

> **Key Insight**
> Are you capable of opening it?

Pragmatic meaning:

> **Key Insight**
> Please open the window.

Critical distinction

```
Syntax     → structure
Semantics  → literal meaning
Pragmatics → contextual meaning
```

🧠 Memory

> **Key Insight**
> **S-S-P = Structure → Sense → Situation**

---

7. Tokenization

Break text into smaller units.

Example:

```
"I love NLP."
```

Tokens:

```
I
love
NLP
.
```

Types:

- word tokenization
- sentence tokenization
- subword tokenization
- character tokenization

Modern systems commonly use subword units because they handle rare/unseen words better.

---

8. Lemmatization

Converts words to their **dictionary/base form**, called a lemma.

Examples:

```
running → run
better → good
children → child
```

It generally uses linguistic knowledge.

---

9. Stop-word Removal

Stop words are common words that may contribute little to certain information-retrieval/classification tasks.

Examples:

```
the
is
a
an
of
to
```

Example:

```
"The cat is on the mat"
```

Possible result:

```
cat cat mat
```

But:

⚠️ **Do not blindly remove stop words.**

For sentiment:

> **Key Insight**
>
> "This is not good."

Removing "not" destroys important meaning.

---

10. Normalization

Converts textual variation into a consistent form.

Examples:

```
HELLO → hello
U.S.A. → usa
can't → cannot
colour → color
```

Possible operations:

- lowercasing
- punctuation normalization
- whitespace normalization
- spelling normalization
- Unicode normalization
- abbreviation expansion

PYQ

The 2023 and 2025 papers explicitly ask normalization and preprocessing.

---

11. Regular Expressions

Regex describes patterns in text.

Important symbols

Regex	Meaning	
`.`	any character	
`^`	beginning	
`$`	end	
`*`	0 or more	
`+`	1 or more	
`?`	0 or 1	
`[]`	character set	
`()`	group	
`	`	OR
`\d`	digit	
`\w`	word character	
`\s`	whitespace	

Example:

```
\d{10}
```

matches a 10-digit number.

Email-like pattern:

```
[a-zA-Z0-9_]+@[a-zA-Z0-9]+\.[a-zA-Z]+
```

Your study material contains practical regex extraction examples.

---

12. Language Models — Overview

A language model estimates:

$$P(w_1, w_2, \ldots, w_n)$$

or predicts:

$$P(w_n | w_1, \ldots, w_{n-1})$$

Example:

```
I want to drink ___
```

Possible prediction:

> **Key Insight**
> water

Core idea

> **Key Insight**
> **Language model = probability distribution over language sequences.**

This becomes the central topic of M2.

🔥 M1 HIGH-YIELD QUESTIONS

2/3 marks

- Define NLP.
- What is morphology?
- Syntax vs semantics.
- What is pragmatics?
- Define tokenization.
- What is normalization?
- What are stop words?
- What is lemmatization?
- Define language model.
- What is ambiguity?

5 marks

- Explain phases of NLP.
- Explain challenges in NLP.
- Explain NLP applications.
- Compare stemming and lemmatization.
- Explain normalization techniques.
- Explain different levels of linguistic analysis.

The actual PYQs repeatedly test phases, preprocessing and ambiguity.

MODULE II — LANGUAGE MODELS, POS & PARSING

12 topics

1. N-gram Language Models
2. Smoothing
3. Rule-based POS tagging
4. HMM-based POS tagging

5. CFG
6. Recursive Descent Parsing
7. Probabilistic Parsing
8. Dependency Parsing
9. Precision
10. Recall
11. F1-score
12. Related parsing evaluation

Your uploaded syllabus explicitly lists these.

---

13. N-gram Language Models

An N-gram model predicts a word using the previous \(N-1\) words.

Unigram

$$P(w_i)$$

Bigram

$$P(w_i|w_{i-1})$$

Trigram

$$P(w_i|w_{i-2}, w_{i-1})$$

General

$$P(w_i|w_{i-n+1}, ..., w_{i-1})$$

Bigram probability

Maximum likelihood:

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

Example:

Corpus:

> **Key Insight**
> I like NLP

> **Key Insight**
> I like AI

Then:

$$P(like|I) = \frac{2}{2} = 1$$

Sentence probability

Using bigram approximation:

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

Include start/end tokens when appropriate:

```
<s> I like NLP </s>
```

🚨 PYQ ALERT

Your actual B.Tech PYQ asks you to:

- calculate bigram probabilities
- apply Laplace smoothing
- calculate perplexity.

So **bigram numerical problems are VVI.**

---

14. Smoothing

Problem:

Suppose:

$$Count(w_{i-1}, w_i) = 0$$

Then:

$$P(w_i|w_{i-1}) = 0$$

One zero probability makes an entire sentence probability zero.

Solution

Smoothing.

Add-one / Laplace smoothing

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + V}$$

where:

- \(V\) = vocabulary size

Example

If:

$$C(the, cat) = 2$$

$$C(the) = 10$$

$$V = 20$$

Then:

$$P(cat|the) = \frac{2+1}{10+20} = \frac{3}{30} = 0.1$$

Other smoothing

Know conceptually:

- Add-one
- Add-k
- Good-Turing
- Backoff
- Interpolation
- Kneser-Ney

Your PYQs include **Kneser-Ney backoff**, so don't skip it.

Perplexity

Measures how well a language model predicts a test sequence.

$$PP(W) = P(w_1, \ldots, w_N)^{-1/N}$$

Lower perplexity:

> **Key Insight**
> better predictive performance.

Memory

> **Key Insight**
> **Perplexity = uncertainty/confusion of the model.**

---

15. Rule-Based POS Tagging

POS = **Part of Speech**.

Examples:

```
Noun
Verb
Adjective
Adverb
Pronoun
Determiner
Preposition
Conjunction
```

Example:

> **Key Insight**
> The boy runs.

```
The → DT
boy → NN
runs → VBZ
```

Rule-based tagging uses linguistic rules.

Example:

> **Key Insight**
> If word ends in "-ly", likely adverb.

---

16. HMM-Based POS Tagging

HMM treats POS tagging as a sequence prediction problem.

Hidden states:

```
POS tags
```

Observations:

```
words
```

Example:

```
The   dog   runs
 ↓     ↓     ↓
 DT    NN   VBZ
```

Two fundamental probabilities:

Transition

$$P(t_i|t_{i-1})$$

Probability of one tag following another.

Emission

$$P(w_i|t_i)$$

Probability of a word being emitted by a tag.

Viterbi

Finds the most probable sequence of hidden states.

$$V_t(s) = \max_{s'} V_{t-1}(s')P(s|s')P(w_t|s)$$

Process

```
Initialize
   ↓
Recurrence
   ↓
Store backpointers
   ↓
Terminate
   ↓
Backtrack
```

🚨 VVI

Your actual PYQs ask you to calculate:

- transition probabilities
- emission probabilities
- Viterbi POS sequence.

---

17. Context-Free Grammar

CFG consists of:

$$G = (V, \Sigma, R, S)$$

where:

- V = non-terminals
- Σ = terminals
- R = production rules
- S = start symbol

Example:

```
S  → NP VP
NP → Det N
VP → V NP
```

Parse Tree

Sentence:

> **Key Insight**
>
> The dog runs.

```
          S
        /   \
      NP     VP
     / \      |
   Det  N     V
   The dog   runs
```

18. Recursive Descent Parsing

Top-down parsing strategy.

Starts with:

```
S
```

and expands grammar rules until terminals match input.

```
S
↓
NP VP
↓
Det N VP
↓
The dog VP
↓
The dog runs
```

Problem

Left recursion can cause infinite recursion.

19. Probabilistic Parsing

CFG rules can have probabilities.

Example:

```
S → NP VP    0.9
S → VP       0.1
```

The parser selects the most probable parse.

This is called a **Probabilistic CFG (PCFG)**.

Your PYQs explicitly ask about PCFG.

20. Dependency Parsing

Instead of constituent phrases, represent relationships between words.

Example:

> **Key Insight**
> John eats apples.

```
    eats
    /  \
John  apples
subj   obj
```

Useful for:

- information extraction
- relation extraction
- question answering

21–23. Precision, Recall, F1

Precision

$$Precision = \frac{TP}{TP + FP}$$

> **Key Insight**
> Of predicted positives, how many were correct?

Recall

$$Recall = \frac{TP}{TP + FN}$$

> **Key Insight**
> Of actual positives, how many did we find?

F1

$$F1 = 2\frac{Precision \cdot Recall}{Precision + Recall}$$

Memory

> **Key Insight**
> **Precision = purity**

>

> **Key Insight**
>
> **Recall = coverage**

🔥 M2 MUST-SOLVE

1. Bigram probability
2. Trigram probability
3. Laplace smoothing
4. Perplexity
5. HMM transition matrix
6. HMM emission matrix
7. Viterbi
8. CFG parse tree
9. Top-down parsing
10. Bottom-up parsing
11. PCFG
12. Precision/Recall/F1

Actual exam evidence strongly supports these as high priority.

MODULE III — SEMANTICS & LEXICAL RESOURCES

11 topics

1. Lexical semantics
2. WSD
3. WordNet
4. Lexical database
5. Semantic similarity
6. Cosine similarity
7. Vector Space Model
8. TF-IDF
9. Word2Vec
10. GloVe
11. FastText

---

24. Lexical Semantics

Study of meaning at the word level.

Relationships include:

Synonymy

```
big ↔ large
```

Antonymy

```
hot ↔ cold
```

Hyponymy

```
dog → animal
```

Dog is a type of animal.

Meronymy

Part-whole relationship.

```
wheel → car
```

---

25. Word Sense Disambiguation

Words can have multiple meanings.

Example:

> **Key Insight**
>
> I went to the bank.

Which bank?

- financial institution
- river bank

WSD determines the correct sense from context.

Pipeline

```
Word
 ↓
Context
 ↓
Candidate senses
 ↓
Disambiguation
 ↓
Correct sense
```

---

26. WordNet

WordNet is a lexical database organized around **synsets**.

It connects concepts using relationships such as:

- synonymy
- hypernymy
- hyponymy
- meronymy
- antonymy

Example:

```
animal
  ↓
mammal
  ↓
dog
  ↓
German Shepherd
```

---

27. Lexical Database

Stores structured linguistic information.

Can contain:

- words
- senses
- definitions
- relationships
- grammatical information

WordNet is an important example.

---

28. Semantic Similarity

Measures how similar two words/concepts are.

Example:

```
car ↔ automobile
```

high similarity.

```
car ↔ banana
```

low similarity.

---

29. Cosine Similarity

For vectors A and B:

$$cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Interpretation:

```
1   → same direction
0   → orthogonal
-1  → opposite direction
```

For many word-vector applications, positive cosine similarity indicates stronger alignment.

---

30. Vector Space Model

Represent words/documents as vectors.

Example:



Words with similar contexts tend to have similar representations.

---

31. TF-IDF

Term Frequency-Inverse Document Frequency.

Measures how important a word is to a document relative to a corpus.

TF

$$TF(t, d) = \frac{\text{count of } t \text{ in } d}{\text{total terms in } d}$$

IDF

Basic form:

$$IDF(t) = \log \frac{N}{df(t)}$$

TF-IDF

$$TFIDF(t,d) = TF(t,d) \times IDF(t)$$

Rare words:

> **Key Insight**
> higher IDF.

Common words:

> **Key Insight**
> lower IDF.

---

32. Word Embeddings

Represent words as dense numerical vectors.

```
"king"
 ↓
[0.21, -0.44, 0.73, ...]
```

Key principle:

> **Key Insight**
> **You shall know a word by the company it keeps.**

Words appearing in similar contexts acquire similar representations.

---

33. Word2Vec

Two architectures:

CBOW

Context → target

```
the [___] sat
      ↓
     cat
```

Skip-gram

Target → context

```
cat
 ↓
the, sat
```

CBOW

Given surrounding words, predict missing center word.

Skip-gram

Given center word, predict surrounding words.

Why powerful?

Learns semantic relationships from context.

Your actual PYQs repeatedly ask **Word2Vec and CBOW**.

34. GloVe

Global Vectors for Word Representation.

Uses global word co-occurrence information.

Word2Vec vs GloVe

Word2Vec	GloVe
Predictive	Count/co-occurrence based
Local context	Global statistics
Neural training	Matrix/co-occurrence objective

35. FastText

Extends word embeddings using **subword information**.

Example:

```
playing
```

can be represented partly through character n-grams.

Advantage:

> **Key Insight**
> Better handling of rare words, morphology and unseen forms.

🔥 M3 HIGH-YIELD

⭐⭐⭐⭐⭐

- TF-IDF numerical
- Cosine similarity numerical
- Word2Vec
- CBOW
- Skip-gram
- WordNet
- WSD
- GloVe vs Word2Vec
- FastText
- Semantic similarity

Actual PYQs heavily emphasize **TF-IDF, Word2Vec and CBOW**.

MODULE IV — NEURAL APPROACHES

8 topics

1. CNN
2. RNN
3. LSTM
4. Transformers
5. Encoder-Decoder
6. Transfer Learning
7. BERT
8. GPT

---

36. CNN for NLP

CNNs can detect local patterns.

Example:

```
"I absolutely love this movie"
```

A filter may learn:

```
"love this"
"absolutely love"
```

Useful for:

- sentiment classification
- text classification

- sentence classification

Architecture

```
Embedding
   ↓
Convolution
   ↓
Activation
   ↓
Pooling
   ↓
Dense
   ↓
Output
```

---

37. RNN

Designed for sequential data.

```
x1 → h1 → h2 → h3 → h4
      ↑    ↑    ↑
     x2   x3   x4
```

Hidden state carries information forward.

$$h_t = f(x_t, h_{t-1})$$

Problem:

Vanishing gradients

Long-term information can become difficult to retain.

---

38. LSTM

Long Short-Term Memory addresses long-range dependency problems using gates.

Three major gates:

- forget gate
- input gate
- output gate

Conceptually:

```
x_t ──→ ┌──────────┐ ──→ h_t
        │   LSTM   │
h_t-1 → │Cell State│
        └──────────┘
```

Memory

> **Key Insight**
>
> **Forget → Write → Read**

---

39. Transformers

Transformer architecture relies heavily on **attention**, allowing relationships between tokens to be modeled without processing strictly sequentially like a basic RNN.

Core components:

- self-attention
- multi-head attention
- positional information
- feed-forward network
- residual connections
- normalization

High-level

```
Tokens
 ↓
Embeddings
 ↓
Positional information
 ↓
Self-Attention
 ↓
Feed Forward
 ↓
Repeated layers
 ↓
Output
```

---

40. Encoder–Decoder

Used for sequence-to-sequence tasks.

Example:

```
English sentence
      ↓
   Encoder
      ↓
Representation
      ↓
   Decoder
      ↓
French sentence
```

Applications:

- translation
- summarization
- question answering

---

41. Transfer Learning

Instead of training from scratch:

```
Large dataset
     ↓
Pretrained model
     ↓
Fine-tuning
     ↓
Specific NLP task
```

Advantages:

- less task-specific data
- faster training
- strong performance

---

42. BERT

Bidirectional Encoder Representations from Transformers.

Encoder-based Transformer model.

Key idea:

> **Key Insight**
> Understand a token using context from both directions.

```
The bank is near the river.
          ↑
      context
 ←──────────────→
```

Excellent for language understanding tasks.

---

43. GPT

Generative Pre-trained Transformer.

Autoregressive/generative approach:

```
Previous tokens
      ↓
Predict next token
      ↓
Next token
      ↓
Repeat
```

Useful for:

- text generation
- dialogue
- summarization
- coding
- question answering

BERT vs GPT

BERT	GPT
Encoder-oriented	Decoder/autoregressive-oriented
Language understanding	Generation
Bidirectional context during pretraining	Left-to-right next-token prediction
Strong classification/extraction	Strong generation

🧠 MODEL EVOLUTION

Remember this:

```
N-gram
  ↓
RNN
  ↓
LSTM
  ↓
Transformer
  ↓
BERT / GPT
```

The central problem:

> **Key Insight**
>
> **How do we represent context better?**

MODULE V — NLP APPLICATIONS & ETHICS

10 topics

1. Text classification
2. Sentiment analysis
3. NER
4. Machine translation
5. Rule-based MT
6. SMT
7. NMT
8. Chatbots/dialogue systems
9. Bias/fairness
10. Explainability

---

44. Text Classification

Assign text to categories.

Example:

```
Email
 ↓
Classifier
 ↓
Spam / Not Spam
```

Other examples:

- topic classification
- toxicity detection
- intent classification
- news categorization

---

45. Sentiment Analysis

Determine emotional polarity.

```
"I love this phone."
       ↓
    Positive
```

Typical classes:

- positive
- negative
- neutral

Can be:

- document-level
- sentence-level

- aspect-level

46. Named Entity Recognition

Identifies entities and assigns types.

Example:

> **Key Insight**
> "Elon Musk founded SpaceX in 2002."

Possible labels:

```
Elon Musk → PERSON
SpaceX    → ORGANIZATION
2002      → DATE
```

NER vs POS

POS	NER
Grammatical category	Entity category
noun, verb, adjective	person, organization, location
Usually word-level grammar	Identifies meaningful entities

The 2023 CS633 PYQ explicitly asks this comparison.

47. Machine Translation

Automatic translation between languages.

```
English
   ↓
MT System
   ↓
Hindi
```

48. Rule-Based MT

Uses manually constructed linguistic rules.

```
Grammar
+
Dictionary
+
Rules
 ↓
Translation
```

Advantages

- interpretable
- linguistically explicit

Problems

- expensive rule creation
- difficult to scale
- struggles with language variability

---

49. Statistical Machine Translation

Learns translation probabilities from parallel corpora.

Conceptually:

$$P(target|source)$$

Uses statistical models to choose likely translations.

---

50. Neural Machine Translation

Uses neural networks, commonly encoder-decoder/Transformer architectures.

```
Source
 ↓
Encoder
 ↓
Representation
 ↓
Decoder
 ↓
Target
```

Compared with traditional systems, NMT can learn richer distributed representations and context.

---

51. Chatbots & Dialogue Systems

A dialogue system interacts with users through language.

Typical architecture:

```
User
 ↓
NLU
 ↓
Dialogue Manager
 ↓
Response Generation
 ↓
User
```

Components

NLU

Understand user intent/entities.

Dialogue manager

Decides what should happen next.

NLG

Generates response.

---

52. Bias & Fairness

NLP systems can inherit bias from:

- training data
- labels
- historical patterns
- model design
- deployment context

Example:

A hiring classifier trained on historically biased hiring decisions may reproduce that bias.

Fairness

System should avoid unjustified systematic disadvantage.

---

53. Explainability

Users should be able to understand why a model made a decision, especially in high-stakes applications.

Examples:

```
Prediction
   ↓
Important evidence/features
   ↓
Explanation
```

Important for:

- healthcare
- finance
- law
- hiring

🔥 COMPLETE NLP FORMULA SHEET

Bigram

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

N-gram

$$P(w_i|w_{i-n+1}^{i-1})$$

Laplace

$$P = \frac{C + 1}{C(context) + V}$$

Perplexity

$$PP(W) = P(W)^{-1/N}$$

Precision

$$P = \frac{TP}{TP + FP}$$

Recall

$$R = \frac{TP}{TP + FN}$$

F1

$$F1 = \frac{2PR}{P + R}$$

TF

$$TF = \frac{f_{t,d}}{\sum_k f_{k,d}}$$

IDF

$$IDF = \log \frac{N}{df(t)}$$

TF-IDF

$$TFIDF = TF \times IDF$$

Cosine

$$\cos\theta = \frac{A \cdot B}{\|A\|\|B\|}$$

RNN

$$h_t = f(x_t, h_{t-1})$$

🏆 ACTUAL PYQ PATTERN

I checked the uploaded NLP papers rather than inventing a PYQ list.

2023 B.Tech IT351 End

The paper asks:

- NLP applications
- spelling errors
- bigram MLE
- smoothing
- perplexity
- neural language models
- POS
- HMM
- Viterbi
- parsing
- PCFG
- CNF

- word vectors
- CBOW.

2023 Mid

It asks:

- NLP challenges
- noisy-channel spelling correction
- POS
- edit distance
- bigram model
- Laplace smoothing
- perplexity
- NLM
- HMM
- Viterbi.

2023 CS633

It asks:

- phases of NLP
- normalization
- lemmatization vs stemming
- bigram frequency/probability
- Laplace smoothing
- neural language models
- TF-IDF
- PMI
- Word2Vec
- HMM POS tagging
- NER
- CKY
- parsing applications.

2024 CS633

It adds numerical emphasis on:

- edit distance
- bigram probability
- add-1 smoothing
- empirical probability
- Good-Turing-style probability estimation
- HMM/Viterbi
- word vectors
- CBOW
- PCFG
- CNF

- top-down vs bottom-up parsing.

2025 CS633

It asks:

- phases + ambiguity
- preprocessing
- N-gram
- smoothing/unknown words
- TF-IDF + PMI
- Word2Vec
- POS
- HMM
- top-down vs bottom-up
- CKY.

🚨 THEREFORE: YOUR NLP VVI LIST

If I rank based on the **actual uploaded papers**, not generic internet predictions:

Tier S — MUST MASTER

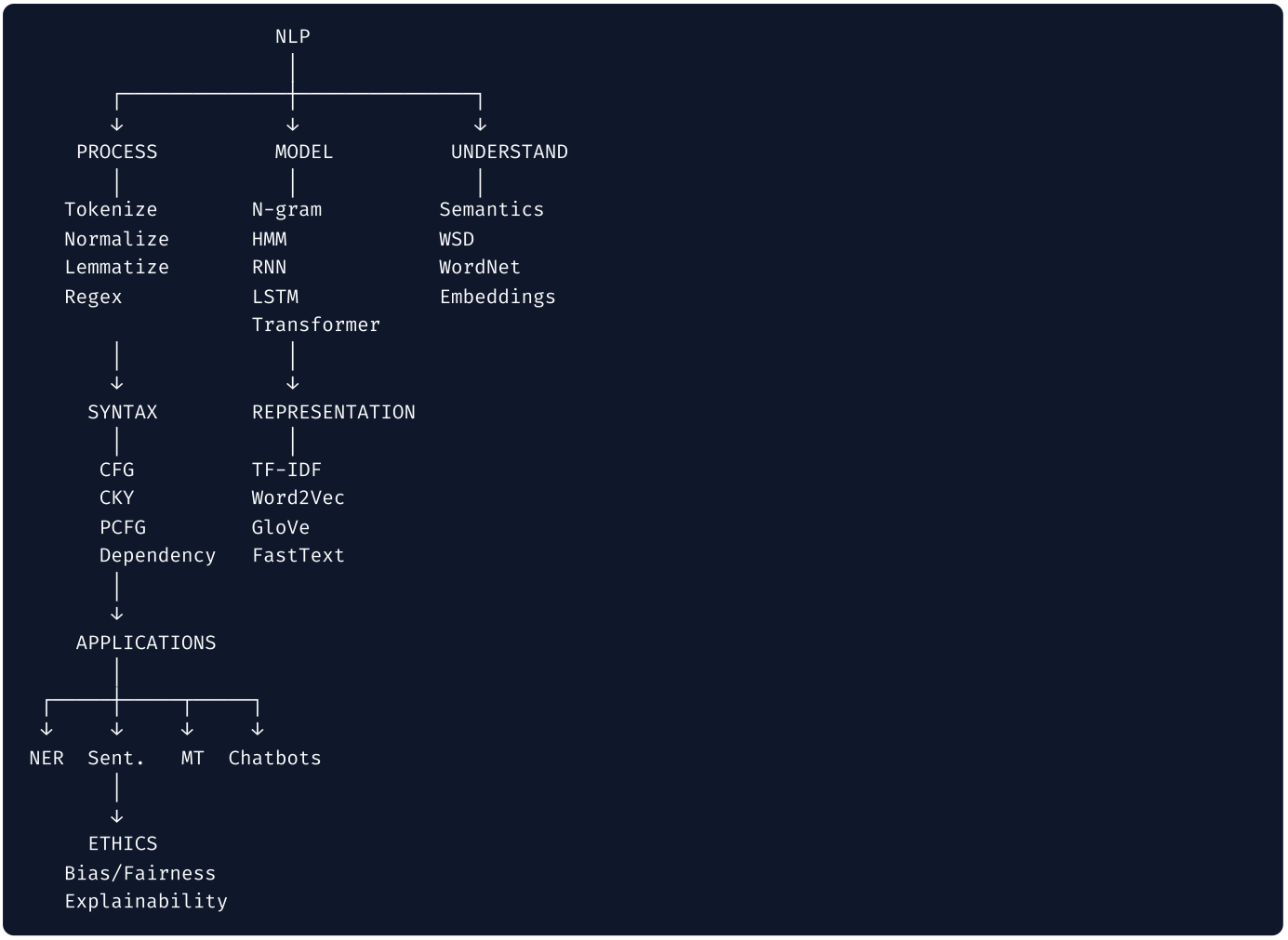
1. **Bigram/N-gram numerical problems**
2. **Laplace smoothing**
3. **Perplexity**
4. **HMM transition/emission probabilities**
5. **Viterbi**
6. **POS tagging**
7. **Top-down vs bottom-up parsing**
8. **CFG/PCFG**
9. **CNF**
10. **CKY**
11. **TF-IDF**
12. **Word2Vec**
13. **CBOW**
14. **NLP phases**
15. **Preprocessing**
16. **Edit distance**

The repeated appearance of these topics across the uploaded papers makes them substantially safer priorities than simply relying on a generic "important NLP topics" list.

🧠 NLP MASTER MEMORY MAP

Don't remember 53 disconnected topics.

Remember this:



🎯 53/53 CHECKLIST

M1 — 12/12

- [ ] Introduction
- [ ] Applications
- [ ] Morphology
- [ ] Syntax
- [ ] Semantics
- [ ] Pragmatics
- [ ] Tokenization
- [ ] Lemmatization
- [ ] Stop-word removal
- [ ] Normalization
- [ ] Regex
- [ ] Language models

M2 — 12/12

- [ ] N-gram
- [ ] Smoothing
- [ ] Rule-based POS
- [ ] HMM POS
- [ ] CFG
- [ ] Recursive descent
- [ ] Probabilistic parsing
- [ ] Dependency parsing
- [ ] Precision
- [ ] Recall
- [ ] F1
- [ ] Parsing evaluation

M3 — 11/11

- [ ] Lexical semantics
- [ ] WSD
- [ ] WordNet
- [ ] Lexical database
- [ ] Semantic similarity
- [ ] Cosine
- [ ] VSM
- [ ] TF-IDF
- [ ] Word2Vec
- [ ] GloVe
- [ ] FastText

M4 — 8/8

- [ ] CNN
- [ ] RNN
- [ ] LSTM
- [ ] Transformer
- [ ] Encoder–Decoder
- [ ] Transfer Learning
- [ ] BERT
- [ ] GPT

M5 — 10/10

- [ ] Text Classification
- [ ] Sentiment Analysis
- [ ] NER
- [ ] Machine Translation
- [ ] Rule-Based MT
- [ ] SMT
- [ ] NMT

- [ ] Chatbots
- [ ] Bias/Fairness
- [ ] Explainability

Total: 53/53.

One important point: **the 53-topic list comes from your uploaded B.Tech study guide, while the PYQ prioritization above comes from the actual uploaded university papers.** Where the papers contain additional material such as edit distance, PMI, Kneser-Ney or CKY, I've included it because your exam evidence shows it matters. I have not silently presented those as separate items in the 53-topic syllabus count.