Midsem: NLP

Core challenges of NLP

- Ambiguity (ek shabd do arth)
  - Lexical (1 word 2 meaning)
  - syntatic (1 word multiple grammar)
  - semantic (contextual meaning)
- Contextual understanding
- Pragmatic (tone of sentence)
- Low resource Language (Lack of training data)
- Multilingual (1 sentence 2 Language) eg- Hinglish
- Etnical concern (Bias in training data)

(challenges ka positive likh do).

Future of NLP

- zero shot
- decision making
- Multilanguages
- Integration wr other modalities
- No dataset bias

Language modeling (Backbone of NLP).

- It predicts next word kya h

Grammar

- It uses grammar rules of the language.

Parsing Technique

1) Top down - start w/ sentence ka root word
2) [illegible]

Statiscal model.

- Based on probability.

Models

1) Unigram - Each word independent P
2) Bigram - one prev. word P

4 break into smaller unit

S → NP VP

2) Bottoms up - words ko jodke sentence banata hai

NP & VP → S

3) CYK Algo

Dynamic prog. use krta hai for parsing CFG.

3) Trigram - 2 prev. word p

4) Neural Language - uses deep learning arch.

eg - RNN

limit - huge computational resources.

Probabilistic approaches

1) MLE -

P(

Smoothing technique

1) Add one - Adds 1 to word count (Laplace)

2) Good-Turing - Adjust probability for unseen event.

3) Back off

4) Interpolation

Evaluation metrics

① Perplexity (PPL)

- Measures how well a model predicts seq. of words

Performance $\propto \frac{1}{PPL}$

② Cross Entropy

- Measures avg bits to encode a message.

Finite state Automata (FSA).

- Abstracted computational model. Used to recognise patterns

State (Q) - Set of Finite state

Alphabet (Σ) - input alphabet

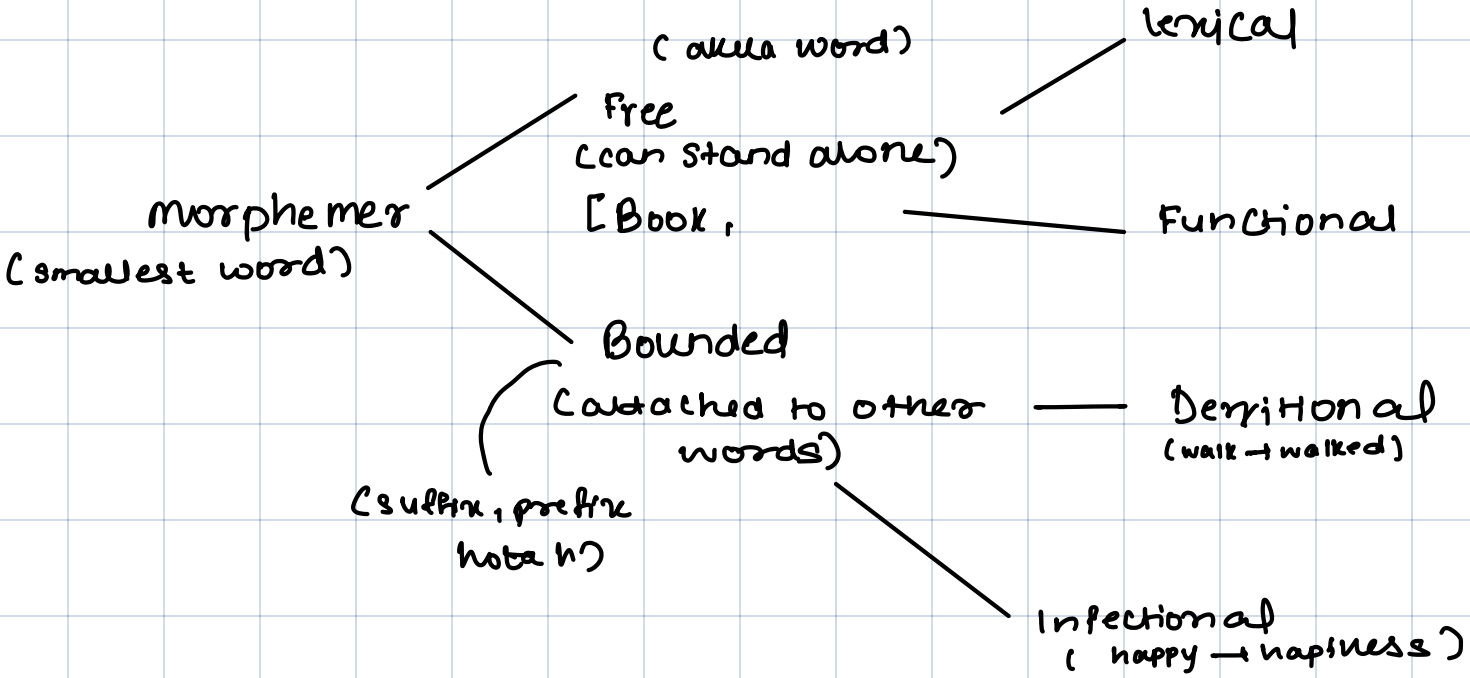
start State (Q_0) - initial state

Accepting state (F) - Accept the string (validate)

Transition Function (δ) - Rule defining bln transitions.

English morphology

- focuses on structure & formation of words



morphological parsing - Analysing words into morphology

- **Tokenisation** - text → words
- **morphological segmentation** - words → morphemes
- **morpheme Tagging** - identifying morpheme ka root
- **Reconstructing meaning** - combining meaning of morphemes.

Tools

- Porter stemmer
- Spacy
- Lancaster stemmer

challenges.

- Ambguity
- complex
- Irregular form

Finite state Transducer (FSA - accept/reject input/output, FST - Maps output to input)

FST structure

- Q - start ka set
- Σ_{in} - input alphabet
- Σ_{out} - output alphabet
- q_0 - start state
- F - Final state
- δ - Transition function

Transition notation

- q_i : current state
- a : input symbol
- b : output symbol
- q_j : Next state

Tokenisation

(Text ko small units mein break krta h)

Types

- word - Text to word ["NLP is cool" → "NLP" "is" "cool"]
- sentence - text to sentence ["Hi. How are you" → "Hi.", "How are you"]
- subword - word to word ["unhappiness" → "un" "happiness"]
- character - text → individual char. ["NLP" → "N" "L" "P"]
- special - special symbol handle krta hai ["visit www.com] → "visit" "..."

Techniques

- Rule - Pre defined rules pe tokenize krta hai
- Statistical - Training data ke basis pe probability
- Neural - dL ka use krta hai

FSA

Accepts / rejects strings only yes/no

state + transision

spell checking . pattern matching

eg check if "cat" is valid
- state

(current, input) → next_state

FST.

converts i/p string to o/p produces new o/p.

state + transision + i/p, o/p pairs

morphological analysis (breaking word)
text translation

eg change running → "run + ing"

(current. state, i/p:o/p) → next. state

NLP Pipeline

Raw Text

remedial

Text cleaning → Lower casing → Tokenisation → Removing stop words → Stemming → Lemmetisation → POS Tagging → NER → Dependency Parsing → Vectorisation

[Just like Raw veg. → Cleaning → Cutting → Cooking → Tasty Food]

Regex → (like search pattern) • Helps find, match, replace words / numbers / patterns

f^n - re.search(), re.findall(), re.replace()
first match, all matches, replaces

symbols - . any char, ^ start of string, $ end of string, * 0 or more repeat, + 1 or more repeat, {} specific no. of rep, [] any 1 char.

eg -
```
import re
match = re.search ('cat', 'I have a cat')   # To find 1 word
print (match.group())
matches = re.findall ('a', banana')   # To find all occurences
```

Error detection.

Identifying wrong word & suggesting right word.

Error detection
- identify krna

Error correction.
- sahi word suggest krna.

Types

- Typing issue
- Cognizitive error (Incorrect knowledge)
- Phonetic error (Fone - phone)
- Contextual error.

Techniques for error detection

- Dictionary - pre defined dictionary (contextual error detect nhi kr payega)
- N-Gram - word sequence ka pattern check krta hai
- Edit distance - minimum possible changes ka count krta hai
- Neural - DL models such as transformers
- Probabalistic - Error ko ek noisy comun. channel se through generate krta hai.

Technique for error correction

- MED - kam changes krta hai
- Contextual correction - Nearest words k hisab se
- Noisy channel model - Bayesian interference ka use krke correction
- Neural spell correction - Transformer, seq2seq jaise DL tech use krta hai

(Group of N-words)

unit-2.

N-Gram (Help in predicting the next item)

- unsmoothened
  - (It can yield 0 probab. for unseen events)
- Smoothened.
  - (Improve by redistributing probability)

P (Phone | Please turn off your cell)

Uni - P (Phone) Bigram: P (Phone | cell) Tri (Phone | your cell)

How to handle unseen probability :-

- Smoothening.
  (Agar koi data training mein nahi ho aur wo testing aa jaye toh wo unseen probab. hota hai)
  - Laplace (Add-on)
    - Har word ki frequency mein +1 add kiya jata hai.
  - Good Turing Kneser-ney - Adjust probability of unseen events. Probability redistribute across n-gram levels.
- Back off
  - Agar Trigram na ho paye, toh bigram / unigram use
  - Agar Bigram na ho toh unigram

(Basically back off krke ek level niche)

Interpolation

- combine probability from tri, bi, unigram to get a better guess.

POS tagging (Assigns word classes to token for synatic analysis).

↳ Helps us understand sentence structure, disambiguate word meaning.

- penn tree-bank common POS tagging technique.

Appln - machine translation.

Method of POS tagging.

① Rule Based - Pre-defined grammar rules

② stochastic - Probability-based models

③ Transformation - rules & stochastic ka combination

POS category

① Noun (proper, common), Verb (Main, Auxiliary), Adjective, Adverb → open (new word add ho sakta hai)

② Preposition, Pronoun, Article, conjuction → closed (fixed words)

Rule based POS tagging

stage 1 :- uses dictionary

stage 2 :- uses large list of handcrafted rules.

Tagging rule

Rule 1 - Agar previous word article ho toh saare verb tag nikalo

Rule 2. Agar next verb tag ho saare verb nikalo.

Hidden Markov model (HMM). (Generative model)

- statistical model used for sequences.

① Hidden state - POS Tags (hidden) observe layer se hidden ko predict karte hai

② Observation state - (visible)

Parameter
- ① Transition - one state to another
- ② Emission - Hidden se observation ke chances
- ③ Initial - starting state ki prob. distribution

Transition prob.

P(tag at t | tag at t-1)
= P(ti)

Emission prob.

P(word at t | tag at t)
= P(wi)

Goal - Find the most likely sequence of tags given the words

Maximum entropy markov model.

Assigns probability without any bias & only on known data.

q model = P(tag | context)

* No strong independence assumption like in HMM.

Advantage

- Can use many overlapping features
- Better for complex lang. patterns

Eg -

If previous word "the" next word likely noun.

Appln

App 1) Text Analysis 2) Speech recog 3) Text to speech 4) Machine translation 5) DNA Analysis 6) Music

3 important problems in HMM.

- decoding

① Viterbi Algo. (used to find hidden state)
uses DP # $O(N^2T)$ time complexity

② Initial probability assign

③ Transition & emission probab calculate

④ Best path find out krke output seq find out karo.

② (Likelihood:) (Forward Algo) / TN^2

(used to est. probab. of observation sequence).

calculate total probab. of different possible state sequence of HMM.

③ Forward / Backward Algo (Learning).

uses supervised L
↳ (we use MLE)

unsupervised hua toh it is true HMM case
↳ (we use forward / Backward Algo).

steps

① Initial prob est.

② calculate transition / emission probab.

③ Reestimate prob.

④ Repeat till convergence

Unit-3

CFG (context free grammar)

- set of rules that help build sentences (like a recipe)

Non terminals (replace hence) - S - sentence, NP - noun phrase, VP - verb phrase

Terminals - Actual words - cat etc (final word)

Production rules - Rules that show non terminals break

S → NP VP (noun followed by verb)

NP → Det N (noun phrase made of determiner & noun)

Treebank

- syntatic - syntax pe focus krta hai
- semantic - Focus on word meaning & relationship.

- collection of sentences that've been parsed into tree structures

- Penn Treebank is a famous treebank. It contains thousands of sentences, each drawn as a tree with NP, VP etc.

CNF

- rewriting CFG so that every production rule is simple

① $A \rightarrow BC$: A non terminal produces two non terminal

② $A \rightarrow a$: A non terminal produces a terminal

* useful because it makes rule binary & predictable

How to convert CFG to CNF

- Eliminate empty space
- remove unit productions
- RHS ka start sym nikalo
  null value nikalo
  a terminal nikalo
  Non T → T
- ★ Binarize the rules
- ★ Replace mixed rules

Useless
null
empty
Terminal
long RHS

Eg:

CFG $S \to \$aAB$

$R \to a$

$X \to AB$

$S \to \$aNonAB$

$S \to RX$

CNF

(Mix rule)

1) $R \to a$ $S \to RAB$ (converting terminal to non T)

2) $X \to AB$ $S \to RX$ (Binarising)

G CNF

CNF (Greibach normal form)

Every rule must start with terminal by non terminals

$A \to aB$

Terminal

why better? - Every sentence starts with a real word.

eg $A \to BC$

① $A \to aD$ (BC) (or any other Terminal)

Terminal

CYK Algo- Refer to ppt

* Parsing Techniques

Top-down Parsing - start from S to generate i/p string
root → terminal
S → NP VP
NP → det N VP → ND

Bottom up parsing - start from the input string
Build tree from word upward.
Det, N, V

Dynamic Parsing. After parsing result store krta hai. to avoid redudant computation.

Shallow Parsing. (chunking) - Bina parse tree ke sidha extract meaningful info. (Fast)

Lexical Parsing - Parsing Accurate (Because ye direct lexicons ko parse karta hai).

Collins Parser - Probability parser using lexicalised CFG rule. Enhance · Enhancing accuracy.

CKY Parsing Algorithm

Example: Parsing a Simple Sentence

We have the following **context-free grammar (CFG) in Chomsky Normal Form (CNF)**:

$$S \rightarrow NP\ VP$$

$$NP \rightarrow Det\ N$$

$$VP \rightarrow V\ NP$$

$$Det \rightarrow "the"$$

$$N \rightarrow "cat" \mid "dog"$$

$$V \rightarrow "chased" \mid "saw"$$

CKY Algo

It is a bottom up Algo

Requires grammar in CNF.

CKY Parsing Algorithm

Input Sentence

Let's parse the sentence:
"the cat chased the dog"

Step 1: Construct the CYK Table

We create a **triangular table** for the sentence:

Word	the	cat	chased	the	dog
1	Det	N	V	Det	N
2	NP	-	VP	NP	-
3	-	S	-	S	
4	-	-	S	-	
5	-	-	-	S	

Probabilistic CKY Algorithm

Step 1: CYK Table Initialization

Each cell **(i, j)** stores:

- Possible **non-terminals**
- The **maximum probability** of generating the substring

Word	John	chased	the	dog
1	NP (0.2)	V (0.6)	Det (0.6)	N (0.5)
2	-	VP (0.7 × 0.5 = 0.35)	NP (0.8 × 0.5 = 0.4)	-
3	-	-	S (1.0 × 0.2 × 0.35 = 0.07)	-
4	-	-	-	S (1.0 × 0.2 × 0.7 × 0.4 = 0.056)

Step 2: Parse Tree Extraction

We extract the best **parse tree** using backtracking.

$$S \rightarrow NP(John) \quad VP(V : chased, NP(Det : the, N : dog))$$

Probabilistic CKY Algo

CKY Algorithm

Grammar in CNF:

1. $S \rightarrow NP\ VP$
2. $NP \rightarrow Det\ N$
3. $VP \rightarrow V\ NP$
4. $Det \rightarrow "the" \mid "a"$
5. $N \rightarrow "cat" \mid "dog"$
6. $V \rightarrow "sees"$

Probabilistic Grammar (PCFG):

1. $S \rightarrow NP\ VP$ (1.0)
2. $NP \rightarrow Det\ N$ (0.9) | $NP \rightarrow N$ (0.1)
3. $VP \rightarrow V\ NP$ (0.85) | $VP \rightarrow V$ (0.15)
4. $Det \rightarrow "the"$ (0.6) | $Det \rightarrow "a"$ (0.4)
5. $N \rightarrow "cat"$ (0.5) | $N \rightarrow "dog"$ (0.5)
6. $V \rightarrow "sees"$ (1.0)

	The	Cat	Sees	a	Dog
1	Det(0.6)	N(0.5)	V(1.0)	Det(0.4)	N(0.5)
2	NP(0.9 *0.5)		V(1.0)	NP(0.9 *0.5)	
3	NP(0.9 *0.5)		VP(0.85 *0.5)		
4			S(1.0 *0.9*0.5*0.85 *0.5)		
5					S

The cat sees a dog.

Dependency Grammar

- word 2 word relation not phrase
- each word depends on another word

eg. "Economic news had little effect"
sub verb obj

had → verb → it is the root. All others depend on it.

Ambiguity - when one sentence has multiple parse trees

unit-IV

for - Relationship b/w two things

× Shreya likes coffee

Shreya(x) ⇒ likes (x, coffee)
relation

All Girls like coffee.

∀(x) Girls(x) ⇒ likes (x, coffee)

Unit- 4

Nikhil ⇒ Likes (x, Tea) FOL – Nikhil(x) ⇒ Likes(x, Tea) FOL could

Human

Logical components – symbol

∧	AND	connective
∨	OR	
¬	NOT	
→	IMPLIES	
↔	IF AND ONLY IF	

LHS RHS

∀x Boys(x) ⇒ Likes (x, Tea)

∃x Boys(x) (x, Tea)

First Order Logic (FOL)

DOL

components

1) constants
specific objects
(John, Apple)

2) Variables
represent arbitary obj
(x, y, z)

3) Predicates
Relationships blw them
Human (John) → John is a human.

4) Functions
Function takes i/p & give o/p
Father (John) = David (David is John's Father)

5) Quantifiers
1) Universal (∀) (For all)
2) Existenial (∃) (For some)

∧ ∨ → ↔

- some boys are intelligent

∃x : boys(x) ∧ intelligent (x)

some object and

- Every gardener likes the sun

(∀x) gardener(x) ⇒ likes (x, sun)

All obj Implies for gardener (obj)

- You can fool all of the people some of the time

(∃t) (∀x) person(x) ∧ time(t) ⇒ can-be-fooled (x, t)

∀x ∀t

Requirement of knowledge rep in NLP.

- expressiveness
  - complex ideas easily.
- efficient reasoning
  - logical deduction
- clarity
  - clear
- scalability
  - large data
- consistency
  - no contradiction allowed
- Modularity
  - knowledge can be updated easily.

FOL for semantic meaning

Natural lang are ambiguous → multiple meanings

FOL gives a mathematical structure

- helps machine understand & reason better.

Thematic agents

"John Broke the window with a hammer".
- John: Agent
- window: Patient / theme
- hammer: Instrument

"Mary felt happy"
- Mary: experiencer

"John gave mary a book"
- mary: recipient

"He flew from India to USA"
- India: source
- USA: Goal

Syntax-driven semantic analysis

- assigning meaning to sentence based on syntactic structure

"John saw mary with book"
We know who has the book

Semantic Attachment

- Attach semantic meaning

Buy → trigger financial decision

Semantic

- meaning of words
- decontextual understanding
- Bank → financial inst.
- word meaning
- used in machine trans.

Pragmatic

- Based on context
- contextual understanding
- can you pass the salt? Here a, not a ques
- Used in chatbot

Reasoning + Winograd

↳ Test for common sense in AI. Harder than turing.

↳ process of drawing conclusion logically

↓

need for machine to think like human

↓

- rule
- cause
- effect

} winograd

↳ solves ambiguity using real world knowledge not just patterns

Description Logic (DL)

- used to combine aspects of both FOL & Set theory.

Key Features

① Concept (classes)

- collection of individual objects

② roles

- Relationship bt ind. eg- hasChild, worksAt

③ Axioms

- (Terminological Box) (Humans subclass of Animal)

Existential qualf. (∃)

∃ R.C ← role, ← class

∃ haschild. human

universal qualf.

∀ R.C

∀ haschild. human

DL (FOL & Set theory combination)

- concept (classes) — collection of objects
- role (relationship) — properties
- Individual (object) — real entities

Axioms:

T-box - Rules & definations

A-Box - Assertions

First order L	Description L
• General rep. of facts & rules	• Special knowledge for concepts & rltnship
• Very expressive	• less expressive
• complex, hard to automate uses variables, quantifiers ∀x (Dog(x) → Animal(x))	• simpler, optimal. uses classes, roles, individuals • Dog ⊑ Animal
• Harder to decide	• Easier, Automated
• General AI, Expert sys	• semantic web

Unit-5

Splitting text into meaningful sections

Discourse segmentation. (A grp of sentences that are logically connected)

- each sentence = a coherent unit

eg: news article → intro, event details

Challenges

- Hidden Boundaries
- Shifting Topic
- Different writing style
- Ambiguous sentence
- context Understanding.

case study. (fake news detection

Problem: Automatically detecting fake news.

challenges faced:
- lack of progression
- mix unrelated events
- contains contradicted/omitted context.

example: "vaccine hv side effects. Also a scientist in Norway discovered Alien DNA in vaccine".

→ not logically connected. It requires discourse analysis.

Coherence.

It is used to test the output quality.

- It should show relation b/w utterances
- Relationship b/w entity.

Discourse segmentation
- ① unsupervised
- ② supervised

① Unsupervised Discourse
- no labels
- finds topic change naturally
- uses lexical cohesion
  - related words
  - sticking together

② Supervised
- Labelled training data.
- It show where exactly topic changes.
- use cue words
  - "however"
  - "because"

Text coherence

Result - "Rahul is late. he will be punished"

Explaination - "Rahul fought w/friend. He was drunk"

Parallel - "He wants food. She want money"
Both want smth at same time

Elaboration - Rohan is from delhi. Ram is from delhi"
Both mean smth but no related and give clear explaination.

Occassion - "Rahul took money, he gave to rohan"
SI inferred from SO.

Building Hierarchical Discourse Structure

Let us now try to build a hierarchal discourse structure with the help of a group of statements. We generally create the hierarchical structure among the coherence relations to get the entire discourse in NLP.

Let us consider the following phrases and serially number them.

S1: Rahul went to the bank to deposit money.

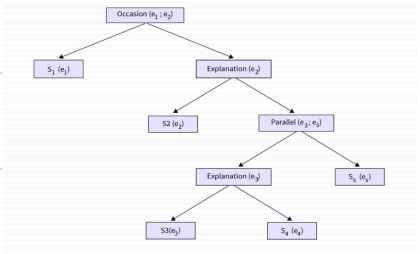
S2: He then went to Rohan's shop.

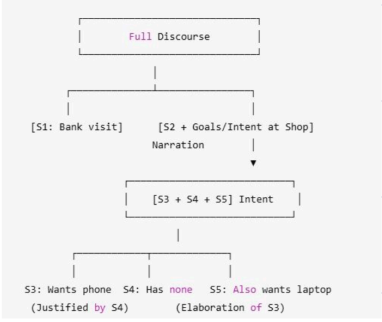
S3 : He wanted a phone.

S4 : He did not have a phone.

S5: He also wanted to buy a laptop from Rohan's shop.

Now the entire discourse can be represented using the below hierarchal discourse structure.



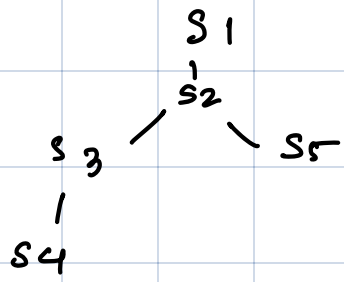


S1 & S2 : occasion

S2 leads to
↓
S3 : explaination

S5 : parallel goal

S4 explain S3.



(Big Act → detail → sub det → exp)

Reference Resolution -

when we see "he", "it" what does it refer to?

He, His → Rahul

For example, look at the below sentences.

- Rahul went to the farm.
- He cooked food.
- His farm was very big.

Antecedant - first time it being mentioned

Referent - whom we refering to

co-ref - 2 word refering to same.

Word Sense

(words having multiple meanings)

eg - Bank → financial
↳ riverside.

How to parse ?

- dictionary
- supervised
- unsupervised
- semi-supervised

Why Important ?

→ correct understanding, Improves Translation, Better search result, AI wrak sentiment analysis.

Tools

- Porter stemmer
  - stemming algo
    =
    (chops off suffix
    "-ing", "-ly", "ed"
  - eg- playing → play
    faster → fast
  - Very fast but not accurate.

- Lemmatizer.
  - Pos tagging algo
  - reducing word to its dictionary form.
  - eg- "am" "be" "is" → be
    "running" → run
  - More accurate but slower.

Code

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
words = [" "]
stems = [stemmer.stem(word) for word in words]
```

Code -

```
from nltk.stem import WordNetLemmatiser
lemmatiser = WordnetLemmatiser()
words = [" "]
lemmas = [lemmatiser.lemmatise(word, pos='v') for word in words]
```

Wordnet

- Large lexical database of eng
- Group words into synsets (set of synonyms)
- Provide relationships (synonym, antonymete).

Brown corpus

- First text corpora in eng
- 1 million words categorised by topics
- Used for lang modelling POS tagging

Framenet

- Semantic Project
- How word evokes frames
- eg- 'commerce_buy' includes buyer, seller, goods, price