Fundamental NLP tasks extracts linguistic information such as syntax and semantics from words, sentences and documents. While they can provide useful information for user end Application such as m/c translation and automatic dialogue system.

On the ~~other~~ word level, morphology is a branch of linguistics that investigates the structure and formation of words, the task of morphological analysis studies automatic prediction of morphological features of input words.

Task	Input	Output
Morphological Analysis	walking	walk + ing
Tokenisation	Mr. Smith	Mr. Smith
Pos tagging	I can open this can	PRP MD VB DT NN

Phases in NLP: There are phases in NLP in order to extract meaningful information from Text. Once these phases are completed your are ready with your refined text and then you can apply some m/c learning algorithm model to predict something.

1) **Lexical Analysis:** In this phase, text is broken down into paragraph, sentence and words. It include techniques as follows:-

(I) Stop word remover (removing 'and' 'of' 'the' etc)

(II) Tokenisation (breaking the text into sentence or word)

⇒ Word tokenizer

⇒ Sentence tokenizer

⇒ Tweet tokenizer

(III) Stemming (removing ing, es, s from tail of words)

(IV) Lemmatisation (converting word to their base form)

2. Syntactic Analysis:— This is used to check grammer, arrangment of words and relationship among words.

Example: Truck is eating oranges → The word does not make sense

Here we need to analyse the intent of words in a sentence. Some of techniques used in this phase

→ Dependancy Parsing
→ Parts of speech (POS) tagging

3. Semantic Analysis:— Once tagging and word dependencies are analysed, semantic analysis extract only meaningful information from the text and reject/ignore the sentence that do not make sense.

Eg:- Truck is eating oranges will be ignored from information summary.

4. Discourse Integration:— It's scope is not only limited to word or sentence, rather discourse integration helps in studing the whole text.

Eg:- John got ready at 9AM. Later he took the train to California.

Here, the m/c is able to understand that the word "he" in the second sentence is referring to "John"

5. Pragmatic Analysis:— It is a complex phase where m/c should have knowledge not only about the provided text but also about the real world.

Eg:- Thanx you for coming so late, we have wrapped up the meeting. (Contain Sarcasm)

Date

NLP Process

Text cleaning
↓
Tokenisation
↓
Stop word removal
↓
Stemming and Lemmatisation
↓
Basic Exploration
↓
BOW or TFIDF

* NLP Based problem usually have unstructured data and when the data is unstructured form, then data processing become difficult.

Unstructured :- which is not proper structured like - Video, text, Image, Audio.

Application of NLP :-

(i) Autocomplete feature in Email
(ii) Voice Recognition
(iii) Text to speech
(iv) Chatbot
(v) Voice Assistance (M.Lexa, Siri)
(vi) Sentiment Analysis
(vii) Spam and Ham Email

Component of NLP :- Two component

(i) NLU (Natural language Understanding)
(ii) Natural language generation

⇒ Syntax and semantic are learned by m/c. It is the step where m/c learn actual meaning and context of sentence. But there are few problems mentioned below which occurs while understanding the text.

(a) Lexical Ambiguity
(b) Syntactic Ambiguity

(a) Lexical Ambiguity:- When a word has more than one meaning.

Example:- I like Bats (the Mammal Bat or wooden cricket Bat)

(b) Syntactic Ambiguity:- Presence of two or more possible meaning with in single sentence

eg:- The Chicken is ready to eat

(Chicken dish is ready to eat or Chicken himself is ready to eat something)

(c) Referential Ambiguity:- Here, when it is not clear about the object you are referring to.

Example:- John called Jay, Later, He Laughed.

(Here who is referring to John or Jay)

Natural language generation work in 3 phase

(I) Text planning:- Here useful content is fetched from Text.

(II) Sentence Planning:- Selection of word forming meaningful phrase

(III) Text realisation:- Execution of sentence plan

Phases of NLP:- we have already discussed in previous page

Text cleaning

When the text corpus is given to us, it may have following issues:-

(i) HTML tags
(ii) Upper/Lower case inconsistency
(iii) Punctuations
(iv) Stop words
(v) Words in their root form

Before using the data for predictions, we need to clean it.

(1.) HTML Tags removal :- while scrapping text data from a website, you may get HTML tag.

eg:- "I'm am neha from <b> Allahabad <\b>"

(2.) Upper and lower case inconsistency :- Remove inconsistency and convert everything into lower case

```
text_data = text_data.lower()
```

(3.) Remove Punctuations :- Punctuations in the text do not make much senses, hence we can remove eg: !, &, %

Tokenisation :- Process of splitting text, phrases, sentences into smaller unit is called Tokenisation.

eg:- ⊙ Splitting of text into sentences (Sentence is considered as Token)
⊙ Splitting of sentence into words (word is considered as Token)

We can import different types of tokenizer from nltk library

(1) Sentence Tokenizer :- text data will be splitted into sentence

```
from nltk.tokenize import sent_tokenize
```

(2) Word Tokenizer :- Text data will be splitted into word

(3) Whitespace Tokenizer :- based on white space word are splitted.

eg:- I love singing, dancing and cooking.

After tokenization → "I", "love", "singing", "dancing", "and", "cooking"

Stop word removal:- There are word in our Sentence which do not provide relevant information. from the text.

Example:- and, of, it, the etc.

There are multiple library which handle with Stop word removal → NLTK.
→ Spacy
→ Gensim.

stopwords like → I, me, myself, am, is, an, our, ours, you, you're, you've, you'll, you'd. etc.

Stemming & Lemmatization :- When we work on textual document, removal of punctuation, Stop word are not just enough. The word in sentence can be used as present tense, past tense or future tense. for eg:- word 'Go' is used as 'go'/'goes' in present tense but 'went' in past tense.

The inconsistency of data can be affect the model training and Predictions, hence we need to make sure that the words exist in their root forms.

To handle this, there are two methods:-

(1) Stemming:- Stemming is the process of converting the inflected words to their root form. In this method suffix are removed from inflected word so that it become root eg:- 'Going' → ing is suffix will get removed and inflected 'going' world will become 'go'.

Developed → Develop
Developing →
Development
Develops

Python library that is used for stemming

```
from nltk.stem import PorterStemmer
```

Stemming to be continued - - - - -

However there are some word which do not get properly handled by stemming process.

eg:- "went", "flew", "saw"

```
print(porterstem("went"))
```

o/p:- went

Stemming process is not smart enough it just know how to trim suffix part.

Fortunately we have lemmatization

Pros:- Good part of stemming is not only useful for English language, it also useful for other language

Pros:- Computationally fast, bcoz it just trim the suffix

Cons:- It is not useful enough if you are concerned about the valid part. Stemmer can give you some word which do not have any meaning

eg:- 'Goes' → "goe"

✱ Lemmatization:- It is where the words are converted to their root form by understanding the context of word in sentence.

Pros:- The root word which we get after conversion holds some meaning and word belongs to Dictionary.

Cons:- It is computationally expensive

Code:-

```
from nltk.stem import WordNetLemmatizer
word_netlemmatizer = WordNetLemmatizer()
```

```
print(wordnet_lemmatizer.lemmatize("going"))
print(wordnet_lemmatizer.lemmatize("goes"))
print(word_lemmatizer.lemmatize("went"))
```

Output:- going
go
went

But you might be wondering that lemmatizer is unable to normalize the words "going" and "went" into their root forms.
It is because we have not passed the context to it. Part of speech is a parameter which we need to specify. By default it is Noun

Code:-
```
print(wordnet_lemmatize.lemmatize("going", pos="v"))
print(wordnet_lemmatize.lemmatize("goes", pos="v"))
print(wordnet_lemmatize.lemmatize("went", pos="v"))
```

O/P
go
go
go

So far we have looked into many text cleaning and normalization technique. Now it's time to study feature Engineering

Basic Extraction:-

- How many words are present
- How " character "
- How " lowercase "
- " " Uppercase "
- " " Stop words "
- What is avg length of each word

Bag of words : BOW is a technique to extract features from the provided text data. This technique counts the occurance of words in sentense.

If the word is found in sentense, then occurence value increase by 1. else 0.

(*) This technique is simple and easy to implement but it has own limitation too. we will discuss later.

(*) Once all steps are done which we have discussed earlier like cleaning, stopword, lemmatization process then apply BOW

Eg: Text

I am too lazy to do anything
Today is friday
Friday is good
Friday is near weekend
I am too lazy

Cleaned_Text

lazi anything
today friday
friday good
friday near weekend

	Today	is	Friday	near	weeken	good
Sent1	1	1	1	0	0	0
Sent2	0	1	1	0	0	1
Sent 3	0	1	1	1	1	0

O/P: ['Today': 1, 'is': 3, 'Friday': 3, 'near': 1, 'weeken': 1, 'good': 1]

(*) how make new column of removed as word count

→ This result for uncleaned text

when we apply BOW in cleaned text

	today	Friday	good	near	weeken
ent 1	1	1	0	0	0
ent 2	0	1	1	0	0
ent 3	0	1	0	1	1

Limitation of BOW:

(1) count vectorizer doesnot understand the meaning of word

eg: (i) I go to sleep at 10 PM and I go to walk at 7 AM.

(ii) I go to walk at 10 PM and go to sleep at 7 AM.

After feature engineering both will result in same column values.

2) There exist so many zeros in data and is called sparse matrix

TF - IDF (Term Frequency - Inverse Document Frequency)

BOW only focuses on frequency of words in sentence. Consider the scenario where BOW is not good enough :-

(I) Suppose we don't want to remove stop words in this case frequency of 'is', 'the', 'a' will also be very high but actually these word do not make sense in the sentence.

(II) Suppose we are processing product reviews on Amazon / Flipkart then the term like "product", "item" are domain dependent and are used too often in our reviews. Hence these word will not help model in learning anything.

(III) The mobile keyword in the phone data set is not giving any value. Keyword like "8 GB", "Splash Proof", "Android OS" will make more sense

Hence TFIDF were most often words are supressed (given lower importance) and unique words (less frequent) are provided a higher weightage in the sentence.

I do not like Vanilla Cake

I do not like Vanilla Icecream.

In above both sentences unique word are 'Vanilla' and 'Icecream'

Formula :-

Suppose no. of words in document A : 6

A : 1 1 1 1 1 1 B : 6

Term	Document A - Term Frequency	Document B - Term Frequency	IDF
I	1/6	1/6	log(2/2) = 0
do	1/6	1/6	log(2/2) = 0
not	1/6	1/6	log(2/2) = 0
like	1/6	1/6	log(2/2) = 0
Vanilla	1/6	1/6	log(2/1) = 0.69
Cake	1/6	0	log(2/1) = 0.69
Icecream	0	1/6	log(2/1) = 0.69

TFIDF = TF * IDF	I	do	not	like	Vanilla	Icecream	Cake
Doc A	1/6 x 0	1/6 x 0	1/6 x 0	1/6 x 0	1/6 x 0	1/6 x 0.69	1/6 x 0.69
Doc B	1/6 x 0	1/6 x 0	1/6 x 0	1/6 x 0	1/6 x 0	1/6 x 0.69	1/6 x 0.69

TFIDF	I	do	not	like	vanilla	Icecream	Cake
Doc A	0	0	0	0	0	0.115	
Doc B	0	0	0	0	0		0.115

Above example you can see, less frequent word have more weight

We can also achieve this by Sklearn

e.g. I do not like vanilla cake

I do not like vanilla Icecream

After using sklearn:

O/P: {'like': 2, 'Vanilla': 2, 'cake': 0, 'Icecream': 1}

Learning

When we apply TFIDF

→ O/P: 'cake': 1.405, 'Icecream': 1.405, 'like': 1, 'Vanilla'

Cake	Icecream	like	Vanilla
1.405	0	0.50	0.50
0	1.405	0.50	

N-gram :- ① N-gram is continous sequence of words or symbol or token in document

① N-gram is continous sequence of n items from given sample or text.

Remember the days we learned how to input an array by first inputting its size ('n')

② N-g Same for n-grams, n-gram is classified as

n	Term
1	Unigram
2	bigram
3	Trigram
n	n-gram

Why we need different types of n-gram??

Ans. Bcoz different type of n-gram suitable for different types of Application. You should try different n-grams on your data in order to confidently conclude which one works the best among all for your text analysis.

Sl. No.	Type of n-gram	Generate n-gram
	Unigram	["I", "reside", "in", "Allahabad"]
	Bigram	["I reside", "reside in", "in Allahabad"]
	Trigram	["I reside in", "reside in Allahabad"]

Here I haven't consider removal of stop words

Now it is clear that unigram means ~~removal of~~ taking one word at a time, Bigram means two word at a time and so on.

Steps → Explore the dataset
→ Feature extraction → Shape (कितने colour, rows हैं)
→ Train – Test Split
→ Basic pre-processing → remove Punctuation, stopwords
→ ~~Code to generate N-gram~~
→ Creating Unigram
→ Creating Bigram
→ Creating Trigram

Doubt What is the difference b/w Tokenised word and Unigram

Numerical on N-gram

Q. Check the probability of ~~<S>~~ I ~~I~~ am ~~not <S>~~ using bigram

~~Ans~~ <S> I am a human <S>
<S> I am not a stone <S>
<S> I I live in Mumbai <S>

Sol Probability of I when S is given → P(I/S)
P " " I " I " " → P(I/I)
" " am " I " " → P(am/I)
" " not " am " " → P(not/am)
" ~~<S>~~ <S> " not " " → P(~~<S>~~ <S>/not)

अब we calculate all of the above for this we count the all individually कितने लिए हम डेटा पर ढूंढेंगे

$$\Rightarrow \frac{C(\langle S\rangle / I)}{C(\langle S\rangle)} \quad \frac{C(I/I)}{C(I)} \quad \frac{C(I/am)}{C(I)} \quad \frac{C(am/not)}{C(am)}$$

$$\frac{C(not \langle S\rangle)}{C(not)}$$

⇒ अब हम sentence 1, 2, 3 में check करेंगे कितने <S> and I कहाँ पर हैं?

<S> and I तीनों ही sentence में हैं ~~Same~~ और फिर divide करेंगे total कितने <S> हैं ~~ab~~ में हमारा $\frac{C(<S>/I)}{C(<S>)}$ आ जायेगा | Same

$\frac{C(I/I)}{C I}$ ~~निकालेंगे~~, $\frac{C(I/am)}{C(I)}$ ——

I I सिर्फ 1 ही sentence में हैं
I am कितने sentence में है
am not कितने है

$\Rightarrow \frac{3}{3} \times \frac{1}{4} \times \frac{2}{4} \times \frac{1}{2} \times \frac{0}{1}$

Total कितने I हैं

$\Rightarrow 0$

Q. Consider the following data

<S> I am Jack </S>
<S> Jack I am </S>
<S> Jack I like </S>
<S> Jack I do like </S>
<S> do I like ~~I~~ Jack </S>

Assume that we use bigram language model based on above data : what is most probable next word predicted by model??

(1) :- <S> Jack ______ (आगे पर last में क्या आयेगा)
(2) :- <S> Jack I do ______ (1, 1, 2, 1 4)

Sol

$P(I/<S>) = C(<S>/I)/C(<S>) = 2/5$

<S> कितने साथ combine हैं

$P(Jack/<S>) = C(<S>)/Jack) = 3/5$

$P(do/<S>) = C(<S>/do)/C(<S>) = 1/5$

I am कितने साथ है combine

$P(am/I) = C(I/am)/C(I) = 2/5$

$P(like/I) = C(I/like)/C(I) = 2/5$

$P(do/I) = C(I/do)/C(I) = 1/5$

$P(\langle s \rangle / Jack) = C(Jack / \langle s \rangle) / C(Jack) = 2/5$

$P(\langle s \rangle / like) = C(like / \langle s \rangle) / C(am) = 1/2$

$P(I / Jack) = C(Jack / I) / C(Jack) = 3/5$

$P(like / do) = C(do / like) / C(do) = 1/2$

$P(Jack / like) = C(like / Jack) / C(like) = 1/3$

$P(Jack / am) = C(am / Jack) / C(am) = 1/2$

1.) <S> Jack I → I, like

2:- <S> Jack I → I, like

Spell-check

finite-state xducer:- A xducer is piece of software that maps one stream of symbols on to another stream of symbols.

What is morphology ⇒ morphology is study of ways words are built from morphenes

* **Morphemes:-** morphene made up of two things
  1. Stem → which is corewood
  2. Affix → which either change meaning of word or Gramatical function

morpheme

Un happy ness → Hence 3 morpheme i.e un+happy+ness

prefix | Affix | Stem | Suffix

Process of determining morphemes:-

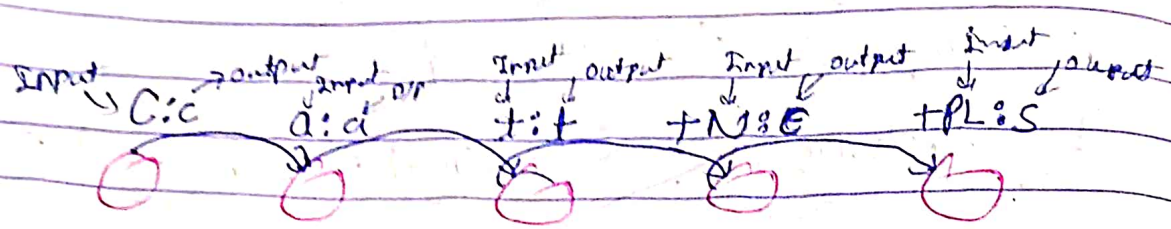
Cats :- cat + N + PL
(cat → word, N → noun, PL → plural)

Cat :- cat + N + SG

FST (Two Tapes)

Lexical set {[c|a|t|+N|+PL]}

Surface {[c|a|t|s]}



ε : Shows Empty

Note:- (I) FSA represent set of strings
eg: {walk, walks, walked, love, loves, loved}

(II) An FST represents a set of pairs of strings
{(walk, walk+V+PL), (walk, walk+N+SG), (walked, walk+V+PAST) ---}

(III) FST have Input : output pairs on labels where as FSAutomata only has Input pair

✡ WordNet:- A Lexical Dictionary of English words

⊙ When it comes to NLP, understanding meaning of words, as well as pre-processing textual data, can be a challenging task. To support this, we often use lexicons.

⊙ We often map the text in our data to lexicon which in turn help us understand the relationship between those words.

⊙ A really useful lexical resource is WordNet. Its a unique semantic n/w help us to find word relation, synonyms, grammar etc.

WordNet is large lexical database of English words. Nouns, verbs, adjectives and adverbs are grouped into set of cognitive synonyms called synsets.

⊙ The NLTK module includes the english WordNet with 155,287 words and 117,659 synonym sets.

```
from nltk.corpus import wordnet
```

~~words~~:-

Semantic:- study of meaning

Semantic Analysis:- Text का meaning findout करने का process होता है। Text का meaning findout करने का जो Process है वो meaning नहीं होता जो हमें समझता है वो meaning होता है जो computer को समझता है।

Ambiguity:- ~~एक~~ word के multiple meaning ~~नहीं~~ निकले that is known as ambiguity

अब कोई Text दिया है तो सबसे पहले semantic analysis उस Text की जो Ambiguity है उसे हटाता है। और ये सब होता है WordNet की वजह से, So basically WordNet is a dictionary which consist of several words.

and with the help of wordNet, semantic analysis can identify the actual meaning of word in sentence.

Wordnet is जीवनसाथी of Semantic Analysis.

What is hypernym and hyponym
जो Class होता है that is Hypernym and class के अंदर जो element होते हैं वो Hyponym होते हैं।

eg:- Animal → It is Hypernym
Dog → It is Hyponym

why it is useful → Thus is useful for Question Answering task.

~~# Ambiguity!-~~

eg:- She killed him with a baseball bat

bat → mammal, wooden object

bat के दो meaning and 'word sense disambiguation' Problem को इस ∧ दूर करता है मतलब अगर एक word के multiple meaning हैं पर sentence के According कौन सा meaning सही है, ये पता करना is called 'word sense disambiguation'

Word Sense Disambiguation के चार Approach होते हैं।
(I) Knowledge based Approach
(II) Hybrid
(III) Supervised and Unsupervised.

~~Why~~ My Synonyms & Antonyms:-

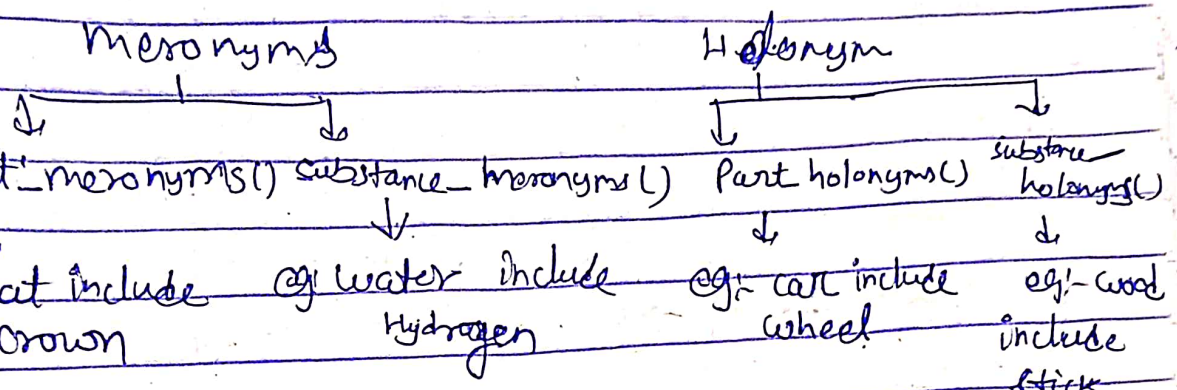
~~Word net cover this quite well.~~ Synonyms means which have similar meaning in some, or all, sense. For eg:- Car ⇒ motorcar, automobile etc

Antonyms mean opposite meaning of word. eg:- Fat is opposite of thin

Meronyms and Holonyms:-

Meronyms express the 'component-of' relationship. That is, a relation between two concept, where Concept A makes up a part of Concept B.

Holonym express the membership of relationship relation b/w two concept, where Concept B is member of concept A.



Entailment:- It is semantic relationship b/w two ~~words~~ Verbs. The relation is unidirectional.

eg:- Snoring entails sleeping but sleeping does not entail snoring

Multilingual WordNet:- It is even utilized by google translate as a part of translation process. This include include not only best known ones such as French, German, Chinese etc. but also less know Kannada, Albanian etc.

WordNet can used to perfor language translation.

POS Tagging

Why Not tell Someone?
Adverb Adverb Verb noun punduation mark

We use short representation referred to as "tags" to represent the categories.

Lexical Term	Tag	Example
Noun	NN	Paris, France, Someone, Rivers
Verb	VB	work, train, learn, run, skip
Determiner	DT	the, a

अगर Noun Plural है तो उसका Tag होगा NNS

It is a process of converting a sentence to a form. (what ch form?

Ans → list of words or, list of tuples (word, POS tag)

अगर suppose हमें words के साथ उसका POS tag भी लिखना है तो इस तरीके से लिख सकते हैं eg: [("Nena", "NN"), ("ate", VB)]

जितने भी word होते हैं दुनिया में हर word का POS tag होता ही है और वो या तो major category में We का रख सकते हैं or Subcategories में

(i) Open Class :- Noun, Verb, Adjective, Adverb
(ii) Closed Class :- Preposition, determiner, Conjuction, Pronouns

eg:- The' determiner में आता है

* एक Word को किस Pos Tagger के अंदर आना चाहिए वो Tagger decide करता है। एक-एक words कई होते हैं जिनके 2 Pos tag हो सकते हैं

eg! I cry everyday
Verb
My comany's name is CRY
Noun

पर ये कैसे decide करता है कौन सा verb है कौन Noun?
Answer है neighbour word को देखकर is CRY में CRY noun के जैसा ही use हो रहा and I cry में cry verb के जैसा

* Application:- (1) Text to speech. मतलब आपने कुछ Type करा और computer उसे voice से बोले पर कभी कभी spelling same होती है लेकिन वो past में किसी और तरीके से Pronounce होगा and present में किसी और तरीके से

eg:- read → Past
→ Present

(II) Searal Bar
(III) Parser

* Problem:- Major problem is ambiguity

* Tags set for english:-
→ Open class :- new-new words include time to time
→ closed class :- Fixed

Rule Based POS Tagging:-

Input → Tag Dictionary or Lexicon + Handwritten rules → output (word, tag)
Input
Tag Diff

सबसे पहले Input आता है जो कि एक string or sentence होता है जिसमें words होते हैं।
अब second step में Tag Dictionary होती है जिसमें वहाँ सारे words होते हैं इसलिए होता है कि अगर sentence में जितने भी words हैं उनको सारे pos tag assign हो सके। But कुछ words ऐसे होते हैं जिनके 2 or 2 से ज्यादा Pos tags होते हैं। (which is called Ambiguity)

Eg:- I play cricket everyday.
(verb)
I want to perform a play.
(noun)

अब जब Ambiguity की Problem आती है तो step 3rd आता है i.e Hand written rules. Hand written rules में बहोत सारे rules लिखे होते हैं पर particular sentence के लिए कौन सा rule Appropriate है तो ये Findout करता है

I play cricket everyday.
(verb) (noun)

but पहले ये sentence को देखे (I/p) → फिर ये tag dictionary or Lexicon के पास जाएगा जहाँ पर I, cricket, everyday सबको को POS tagging मिल गयी but play can be used as verb, noun अब ये Hand written rule के पास जाएगा जहाँ ये Findout होगा कि sentence के according most appropriate क्या है? यहाँ पर verb है ∴ play verb के जैसे use हो जाएगा

Application of Pos tagging:
1. Named Entity Recognition
2. Question Answering System
3. Chatbot
4. Word sense Disambiguation

POS tagging & Preprocessing steps

Important:- Hidden Markove model is very important model. This generate model based on set of I/p sequence.
For this we will study Markove property
(M) memory less (future state depends on previous state)
(A).

Watch CampusX ⇒ You Tube video for POS tagging + HMM

Q. Suppose we have:-

Sentence 1	Can	Nitish	google	IsDTw
Sentence(2)	Will	Ankita	google	IsDTw
Sentence(3)	Ankita	loves	will	
Sentence(4)	Will	loves	Google	
Sentence(5)	Nitish	loves	IsDTw	

Now the Question is first train above after training of above 5 sentence Predict the sentence ⇒ Will will Google IsDTw

Sol^n First we manually train above 5 sentence

Sentence(1) Can(M) Nitish(N) google(V) IsDTw(N)
Sent(2) will(M) Ankita(N) google(V) IsDTw(N)
Sent(3) Ankita(N) loves(V) will(N)
Sent(4) Will(N) loves(V) Google(N)
Sent(5) Nitish(N) loves(V) Google(N) IsDTw(N)

Now we will plot Emission table through above analysis.

Total 2 बार Nitish आया है and Total noun 10 है।

	Noun (N)	Modal (M)	Verb (V)
Nitish	2/10	0	0
Learn	0	0	2/5
Ankita	2/10	0	0
Will	2/10	1/2	0
Google	2/10	0	2/5
Can	0	1/2	0
Crow	3/10	0	0

will model के तौर पे 1 बार use हुआ है total model 2 है।

Now we will plot ~~emission~~ Transition Table :-

To draw this table we will assume that every sentence start with 'S' and ends with 'E'

	N	M	V	E	Sum
S	3/5	2/5	0	0	5
N	0	0	5	5	10
M	2	0	0	0	2
V	5	0	0	0	5

noun से start होने वाले

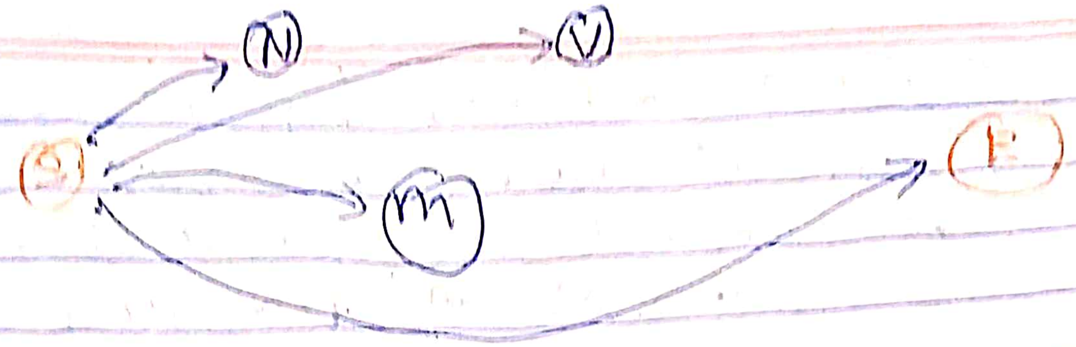
noun के बाद noun आया है

noun के बाद end हो गया है

Now हर Row के sum से divide करेंगे to get Actual Transition Table

	N	M	V	E
S	3/5	2/5	0	0
N	0	0	5/10	5/10
M	2/2	0	0	0
V	5/5	0	0	0

Now we will make Hidden markove model



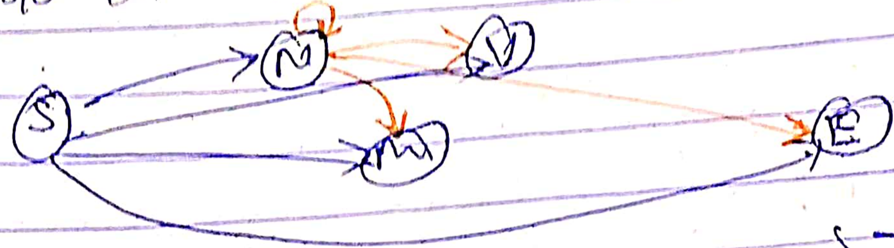
हमारे सभी words को Emission Table से ले कर के

Nitesh loves Ankita will google can [illegible]

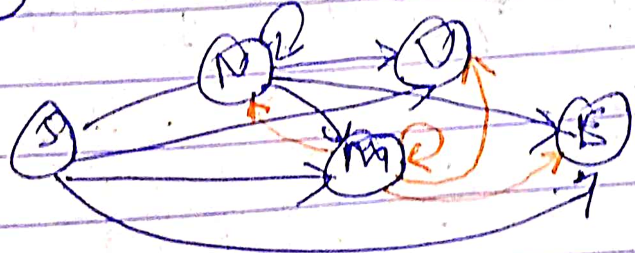
Step (I)

हमने यहाँ draw कर लिया Start से सभी तक जाना और (सभी की Transition से लिया गया)

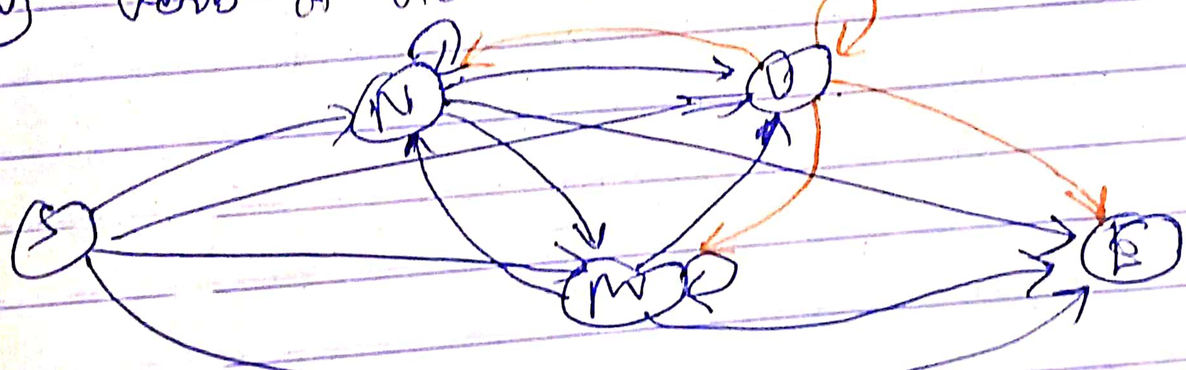
Step (II) अब हम Noun से सब जगह जायेंगे



Step (III) अब model से सब जगह जायेंगे



Step (IV) Verb से सब जगह जाना



Now हमें Will will google

→ हमें पहले सब combination try करना है।

→ first we will assume all words are nouns

now we will calculate हमारे noun होने की probability