Natural Language Processing —

- Language in cognitive science
  - language : structured system of communication that of grammer and vocabulary

- Linguistics : Cnoam chansky
  - Scientific study of language & structure, including grammer, syntax & phonetics
  - study of human capacity to communicate & organiz using diff. tools.

- Ambiguity :
  - ambiguity types
    - i) Lexical : multiple meanings of words
    - ii) structural : How phrases are linked to other link (units)

  - dependency ambiguity

  - pragmatic ambiguity : Most Complex.

- Language analysis : Relationship

1) paradigmatic relationship : easily replace 1 wor
   → Dog barked loudly → dog with fox/wolf.

2) syntagmatic relationship : semantic relation b combine to form a meaningful sequence in
   → We can replace dog with other animal b running or something like that.

Scanned with OKEN Scanner

- Language analysis : Form , Function , meaning.
  - Form - Provides structure (e.g. sentence with correct gram
  - Function - Determines the role (command or question)
  - Meaning - conveys the message (what the command / is about ?)

can you open window

Form - Interogative , Function -

mean - asking for help

1) Form :

2) Function

Action : she runs fast. verb → run.

question | request : can you pass salt ?

3) Meaning :

Focuses on idea.

Literal meaning: (Denoting profession) : she is teach

Implied meaning : (sarcasm)

Cultural meaning

Natural Language Understanding : (NLU)

- literal meaning.
- implied context. - intent, tone, ambiguity

Natural Language Generation (NLG)

Scanned with OKEN Scanner

- Language analysis : Form , Function , meaning.
  - Form - Provides structure (e.g. sentence with correct gram
  - Function - Determines the role (command or question)
  - Meaning - conveys the message (what the command is about?)

can you open window

Form - Interogative , Function -

mean - asking for help

1) Form :

2) Function

Action : she runs fast. verb → run.

question | request : can you pass salt?

3) Meaning :

focuses on idea.

Literal meaning: (Denoting profession) : she is teacher

Implied meaning : (sarcasm)

cultural meaning

- Natural Language Understanding : (NLU)
  - literal meaning.
  - implied context - intent, tone, ambiguity
- Natural Language Generation (NLG)

Scanned with OKEN Scanner

Why NLP ?

- Spam detection : Ham | spam
  Sentiment analysis : Happy | sad | angry
  Text summarization
  Text generation : New content generation
  Topic modeling : Find out text topic / classify text into topic
  Text clustering : group similar data.
  Text similarity :
  language translation
  question answering
  Named entity recognition (NER)

data acquisition :- public da
webscrapping, img to text
pdf to text :→ pyPDF2 lib
For pdf to .txt file
- Data augmentation - gen
Fake data by synonym re
by translation, flipping, a
some noise.

NLP

Natural Language processing.

[Neuro linguistic programming] ~ sudo

NLP :

- Word embedding → map word with number vectors.
  1st convert languge to numbers → For ML
- semantic similarity (~~cosine~~ similarity) → after mapping word with vectors we can perform arithmatic operations
  - king & queen

$$\text{cosinine similarity}(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}|\,|\vec{B}|}$$

1 = very similar
0 = unrelated
-1 = opposite

- common NLP pipeline :

Input text → Tokenization → Stop word removal → POS taggin
→ stemming / Lemmitization → parsing → entity naming →
Document object

Scanned with OKEN Scanner

- NLTK & spacy :

- Text preprocessing :

1) Tokenization : Input → split into tokens (word, no, !)
2) Text normalization : Lowercasing, remove puctuations & special characters using regular expression
3) stopword removal : remove common words. and, as, the
4) Stemming & lemmitization : Lemma – basic form of word
5) part of speech tagging : Identifying meaning of words, tells wheather verb, noun, etc.
6) Named entity recognition : real world object with proper nam (person, organization, location)
7) Bag of words : sklearn has count vectorizer
8) TF-IDF

30.05.2025

- Tokenization : key Terms

Tokenization : NLTK versus spacy

- spacy tokenization concepts :
  - Features
    1) contractions
    2) Abbrevations
    3) URLs, emails
    4) Hyphened words
    5) No.s with units

- Text cleaning
  - Unicode normalization
    If text data may conta symbols, emojis, graph character / special chara remove them or convert machine readable text
  - Regex
  - Spelling correction
    - create

Scanned with OKEN Scanner

- Span :
- Span : contiguous sequence of tokens in a Doc object in SpaCy
- Represents a slice of document, such as
  1) A single word
  2) Multiple words
  3) The entire document.
- Help us extract, analyze & annotate specific parts of text
- Retains all linguistic annotation like POS tags, entities, dependencies.
- Not a copy of tokens, it is a view into original Doc.

- **Pipeline** :

A sequence of components for processing text :
- Tokenizer
- Tagger (POS tagging)
- Parser (Dependency parsing)
- Named Entity Recognizer (NER)

- sentences:
  - .sents: A generator that yields span objects representing sent in Doc.

31.05.2025

- **Stemming**

Stemming } A word to it's base form
Lemmatization } → better

- spacy does not have stemmer, it relies on lemmatization
- NLTK stemmer types : Port stemmer and Snowball stemmer

- **Porter stemmer** : (developed by Martin Porter in 1980)
  - process of removing suffixes from english words
  - performance very fast
  - support english only
  - moderate accuracy
  - No customizable
  - High aggression

NLTK
- Stemming
  - Port stemmer + remove suffix
  - Snowball stemmer
- Lemmatization

pros : simple & fast
- works well in many general purpose appl

cons :
- sometimes over-stems/under stems (aggressive or not enough)
- Hard coded rules make it less diff

- steps : (porter's algo / porterstemmer)
  1a. Common suffix removal
  1b. suffix removal based on specific conditions
  1c. Remove finale if word has more than 1 syllable
  2. remove specific suffixes based on condn
  3. Remove suffixes based on conditions

Scanned with OKEN Scanner

- **Snowball Stemmer** : / English stemmer / Porter2Stemmer
  cons : slightly slower than porter (due to complexity)
  - Improve original with logic & speed (developed by martinporter
  - advanced & language-independent → language support → multiple lang.
  - accuracy - high, customizable, aggressiveness - balanced • Rules easier to understand
- **Lemmatization** :
  - Word to root/base form, known as lemma.
  - goal : (like stemming) → group together diff inflected forms of a word, so that they can be analysed as a single, common entity.
  - more accurate than stemming.
  - Not for longer text $\xrightarrow{\text{reason}}$ higher computation cost
  - better because always produces valid words.

stemming → Fast → because prunning

tokens.lemma → Hash value
tokens.lemma_ → lemma

- **Span** :
  - contiguous sequence of tokens in a Doc object in spacy.
  - single word, multiple words, entire doc.
- **Text cleaning** :
  - after preprocessing, a possible use case : count word occurence.
  1) Replace newlines with spaces
  2) Replace punctuation characters in python with spaces
  3) convert text to lower case
  4) Tokenize cleaned text

  !"#$%&'()*+,-./:;<=>?@[]^_{|}~
- **Text cleaning / Preprocessing**
  1) Remove newline characters
  2) Remove punctuation characters
  3) Convert text to lower case
  4) Tokenize into words

- Text - Preprocessing, NLP pipeline steps
  1) tokenization
  2) Text normalization : lowercasing, Remove punctuation, & special characters using regular expression.
  3) Stopward removal : remove common words such as and, the, etc
- stemming & lemmatization
- part of speech
- named entity recognition
- Bag of words :
- TF-IDF : Term freq & inverse doc. freq using TfidfVectorizer
- Word embeddings : vector representation of words using word2vec

- **Morphological analysis** :
  analyzing the internal stru. of words

Scanned with OKEN Scanner

- **Stop words** :
  - words such as a, the appear so frequently that they need not be tagged unlike nouns, verbs, modifiers
  - Carry little useful meaning
  - filtered from text to be processed
  - Spacy has built-in list of ~305 english stop words.

  * other languages, we need:
    nlp = spacy.load ("fr_core_news_
    # french mode

- Data Science and NLP : Workflow

NLP:

Decide scope → gather data → Data clean → preprocess text → Explore Data → model data → Derive Insigthts

NLP pipeline

NLP approaches :

1) Traditional NLP (Rule/ Probability/ statistics-based)
2) Machine learning based : Ex : spacy
3) Deep learning / Transformer-based : Ex : Hugging Face
   - Earlier : RNN, LSTM, GRU
   - Now : Transformer based → BERT, GPT-4

- Traditional / modern?

1) goal : sentiment analysis / Text classification / topic modeling.
   - For (< 100k) rows → Tradition
   - For (> 1 billion) rows → modern
2) goal : Text generation / machine translation / ques. answering
   - use modern techniques.

Scanned with OKEN Scanner

- Feature extraction From text
  - numeric ⟷ text

- Count vectorization
  - counts how many times word come in doc/text
  - Also called as Bag of word
  - text doc. to Document term matrix

    Rows = document

    columns = Unique words (tokens)

    values = count of words in that document

  - same representation for both the sentences in DTM Form, ignoring order of words

- TF-IDF (Term frequency - Inverse document frequency)
  - problem in bow : 1) Ignores sequence/order
    2) stop words, common words may get higher importance

    TF-IDF : 1) remains same
    2) solved.

    ↓

    problem : 'The' used frequently → no use

    goal: scale down the imp(wt) of such common words to reduce their impact.

  - Words we want to ignore:
    1) scattered accross many documents
    2) Don't help us distin. betn doc.
    3) Their overall counts is high, which is misleading.

Scanned with OKEN Scanner

TF-IDF :

Example : The repeated 3 times, It notes this & reduce importance of the.

doc1 = Hello I m sharu
doc2 = Holloy I love m
doc3 = I love Ai and I lo

$TF = \frac{1}{4} = 0.25$

TF-IDF Vectorization :

1) TF (Term Frequency) : How frequently term occured. → NLP is eas
$TF(NLP, d) = \frac{1}{3}$
t = Hellow

2) IDF (Inverse document Frequency) : Measures how unique/ rare a term is across all documents.

t = Sharu

$\left(\frac{4}{1+1}\right)+1$ $\quad IDF(t) = \log\left(\frac{N+1}{1+(df(t))} + 1\right)$ $\quad \log\left(\frac{4}{1+2}+1\right)$ $\quad n$ = no. of document t is

2+1 = 3 → more imp $\quad \log\left(\frac{7}{3}\right) = 0.28$ → less imp

3) TF-IDF : $= \log\left(\frac{N+1}{1+df(t)} + 1\right) \Rightarrow \log\left(\frac{N}{1+df(t)}\right)$

for Hello → 0.25 × 0.28 = 0.07.

for sharu → 3 × 0.47 = 1.41 $\quad$ 1 used if zero in deno., 1 ignored.

df(t) = no. of document containing term t

sharu → IDF = $\log\left(\frac{4}{1+1}\right)$

$TF\text{-}IDF = TF\text{-}IDF(t,d) = TF(t,d) * IDF(t)$

word common in / within document → High TF-IDF
word common accross document → low TF-IDF.

DF : The word, 'The' → occured in how many unique documents (document Frequency)

$$IDF = \log\left(\frac{1+N}{1+DF(w)}\right) + 1$$

IFIDF limitation : BoW assumption where only presence & frequency of words matter, not sequence.

- Why normalize TF in TFIDF calculation;
  - Unlike dividing by doc length, used to prevent longer documents from having disproportionately higher values just for being more verbose

- IF-IDF interpretation :

1) High-TF-IDF : Desirable for distinguishing documents - identify keyword
   - Indicates important, unique/ topic specific term
     - High TF : more freq. in spec. doc.
     - High IDF : word rare accross doc.

2) Low TF-IDF : indicates common / less informative words & - happen when : → For common words.
   - Low TF : words appear rarely in document
   - Low IDF : Word is appearing in many documents & is ∴ not useful, the, is

- High TF-IDF : Useful for Identifying keywords or distinguishing terms
- Low TF-IDF : Words either too common or not significant for identifying content.

Scanned with OKEN Scanner

n-grams : sequence of n words

- 1-gram (unigram) : language
- 2-gram (bigram) : natural language
- 3-gram (trigram) : natural language processing

- ngram_range = (1,1) : only include single words (unigrams)
  - I Love NLP : 'I', 'Love', 'NLP'

n-grame_range = (1,2)
  - I love NLP : 'I', 'Love', 'NLP', 'I love', 'love NLP'

- Why we use bigram / trigram ?
  - Capture content / phrases better. → 'not', 'good' vs 'not good'

- norm = ('l2')

Particularly impt unique to doc's core topic
- longer document will naturally have higher TF-IDF values even if words aren't

- L2 Normalization is applied to each document's vector to make comparison fair & scale-independent.
- makes cosine similarity & other distance-based methods works properly.
  - Scales TF-IDF vector so that sum of squares of values equals 1
  - Ensures long documents don't get higher weights just bcoz they have more words.
- Ex: 'cat sat on the mat'

TF-IDF vector : [0.5, 0.5, 0.7, 0.2, 0.3]

L2 norm = sq. root of $(0.5^2 + 0.5^2 + 0.7^2 + 0.2^2 + 0.3^2) = 1.058$

divide each value in vector by L2 norm, i.e 1.058 to normalize vector.

03.06.2025.

- spam or ham

1) CV + TF-IDF or TF-IDF only
2) classification (spam / ham)
3) Evaluation of model performance
4) checks most impactful words for spam/ham
5) chi-square test of independance
   Ho : No dependence, H1 : Dependence
   i) High test score > chi-square critical
   ii) p-value very low
6) SMOTE 7) Retrain MNB using SMOTE

Scanned with OKEN Scanner

- TF-IDFVectorizer vs TfidfTransformer

- CV + TF-IDF → TF-IDFTransformer → pre computed CountVectorizer output
- TF-IDF only → TfidfVectorizer → Raw text → combines text pre-processing (like tokenizer, stopword removal, term freq calculation & IDF weighting in 1 step.

- Naive Bayes Exmple :
  - Based on conditional probability
  - Naive Bayes : If an email contains word ASAP, how likely it is going to be spam

$$P(\text{spam} \mid \text{ASAP}) = \frac{P(\text{ASAP} \mid \text{spam}) \times P(\text{spam})}{P(\text{ASAP})}$$

- Word sense Disambiguation (WSD)
  - The task of determining the correct meaning of a word based on it's context, especially when the word has multiple possible meanings. [contextualized word embeddings such as BERT & GPT implicitly handle WSD]
  - Importance : For ml translation, information retrieval, que-ans, text summerization.
  - challenges : Ambiguity in natural language is common.

- WSD appraches :
  - Knowledge based : looks at words around to get content
  - Supervised learning : WSD as classification problem using labelled data.. Learns features such as neighbourg words, POS tags
  - unsupervised learning : cluster similar words based on content.
  - Deep learning techniques : use content-aware word embeddings.
    - WordNet : a large, manually created lexical database of English words & their semantic relationships

04.06.2025.

- Semantics :
  - semantics - study of meaning in language.
  - goal : Teaching machines to understand language like human.

TF-IDF is better than BOW, but both ignored words meanings & relations, semantic models like embedding & transformers ← spaCy, en_core_web_lg → English model

BOW → TF-IDF → Word2Vec → Transformers

- Word2Vec :
  - predictive word embedding model introduced by google in 2013

Scanned with OKEN Scanner

- Contiguous bag of words (CBOW): Predicts word given it's surrounding context.
- Skip-gram : Predicts context words from a target word.
- Based on Deep learning. - CBOW / skip-gram.

· Embeddings :

- Dense, low-dimensional vector representation of data.
  - Dense : Every element in vector has value.
  - opposite of sparse : In BOW most are zero.
  - Low dimensional : very few dimensions (100-300) as compared to count vectorization / bow, but much more meaningful

· Text Vectorization method / Techniques in NLP :

- CountVectorizer
- TF-IDF
- Word2Vec
- Transformers

Word2Vec : problem : For every word, there is only 1 vector value in Word2Vec, so no contextual meaning
- Contexts - do not understand context
- Spacy auging problem :-
  1) No context
  2) stop word influence : same imp stopwords

Hugging Face models :

Algorithm : CBOW - given surrounding find target word.

1) Remove full stop
2) Join all words into single string
3) Use a sliding window of 4 → ML is a sub-branch of AI
   target word : 'is', surrounding words : 'machine learning' a 'subt branch'

trainX = each 'surrounding words' (4 word list)
Y = OHE target word

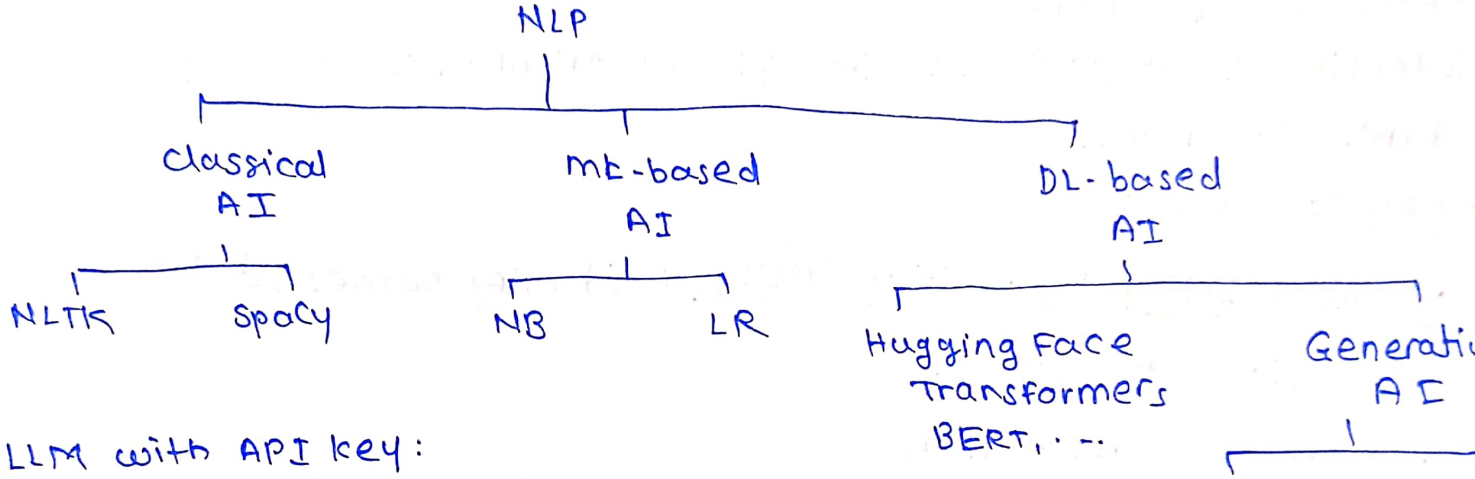
5) To predict, give X & get Y

Scanned with OKEN Scanner

- Skip-gram
  - Inverse of CBOW
  - predict surrounding words

- <u>sentiment analysis</u> :
  - VADER (Valance Aware Dictionary & sentiment reasoner.

- <u>Langchain</u> :



- LLM with API key:
  - OpenAI, Google Gemini
- open source LLM
  - Llama, Gemma, mistral

Ollama: command line utility that we need to install on our machine to use LLMs

- langchain API : programmatic access to local LLM
  - Interface betn appn & LLMs
  - Data aware : connect LLMs with our own data sources
  - agentic aware : we can ask LLM to take actions

- <u>setting up an open source LLM</u> :

- <u>Deep parsing</u> :
  - In-depth analysis of language structure and meaning
  - syntactic parsing: analyzing structure of sentence based on gramme
    POS, dependency parsing, Constituency parsing, Phrase struc trees
  - semantic parsing : goes beyond grammer to understand meanin
    NER, Who did what to whom (semantic role labeling), WSD.
    Question-to-SQL or logical form transformation

Scanned with OKEN Scanner

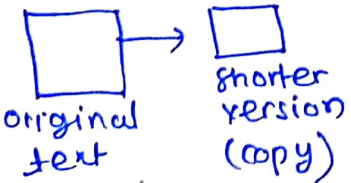
- **Shallow or Deep Parsing** :

- **Semantic parsing** : Process of converting natural language into a structured, machine-readable representation of it's meaning.
  - goals : understanding meaning, structure mapping, machine usability

- **Information Extraction** :
  - Technique to automatically extract structured information from unstructured text.
  - Identify & categorize specific pieces of info like names, dates, locations.
  - key concepts :
    - **Named Entity Recognition (NER)** : identify names, dates,
    - Relation extraction (RE) :
    - Coreference resolution : links pronouns/phrases to entities they refer to
      Ex : Barak Obama was born in Hawaii. He was the 44th president
    - Event extraction → Identifies events described in text & their participants

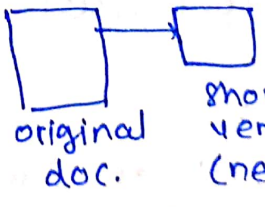
- **Automatic Summarization** :
  - considering a text to generate a shorter version while retaining it's essential meaning.
  - appⁿ : News summerization, Doc. summerization, chat summer, Social media highlights
  - 2 Types :

1) Extractive Summarization



- selects & extracts key sentences/phrases from original text
- Technique : sentence ranking (e.g. TextRank, LexRank) TF-IDF, graph-based
- simpler, faster, preserves original

2) Abstractive summarization.



- Rewrites & synthesizes information
- techniques : seq-to-seq with attention, transformers
- more fluent, human like summary but requires more com...

Scanned with OKEN Scanner

CCEP

- **Anaphore Resolution** :
  - goal : Identify a referent of an expression. (usually pronoun / noun he/she/it)
    Ex: Alice went to library because she needed a book.
    Here she is an anaphor referring to alice - helps understanding text coherence & linguistic analysis
- **Ontology** :
  - Formal representation of a set of concepts within a domain & relationships betn those concepts.
  - Provides a shared vocabulary for everyone, including machine
  - key components : classes (concept), instances (individuals), properties (behavior), axioms, data properties (relationships betn instances & data values)
    relation betn classes/instances
- **Semantic Web** :

- **Markove model** :
  - mathematical model where the next state only depends on current state, & not on full history of past states
    use : Time series, speech recognition, other sequential data analysis
    $X_{t-2} \rightarrow X_{t-1} \rightarrow X \rightarrow X_{t+1} \rightarrow X_{t+2}$
    Markove property : $P(X_t | X_{t-1}, X_{t-2}, \ldots) = P(X_t | X_{t-1})$ ... chain rule
  - markove property → X_t depends on X_{t-1}
  - 1st order → immediate one previous state to predict current state
- **markove assumption** :
  - property : redy on previous → property
  - reality : in NL it is not true many times.
    - It is a simplification & does not fully capture long-range dependencies in lang. It's still widely used due to it's mathematical simplicity & tractability. called markove assumption
- **markove model** :
  - state transition matrix : represents probability of moving from 1 state to another in markove model.
    state after 1 day → VP, after 2 days → (v.p) · P = VP^2, after 3 days = $V \cdot P^3$
- **Markove entropy** : $[x, y, z] \rightarrow x$: probability of being sunny, $y \rightarrow$ being rainy, $z \rightarrow$ being cloudy.
  - Quantifies uncertainty / randomness in a system. In markove, each state has probability. Entropy tells us how unpredictable next state is

06.05.2025.

the . . . . (system, machine, computer, person) → entropy high

great . . . (job) - - → less entropy

very . . . (good, bad) → 0.6 & 0.1 diff → so, less entropy.
(0.6, 0.1)

$H(X_i) = -\sum_{i=1} P(X_i) \log_2 P(X_i)$

- **Statistical Parsing** : A type of syntactic parsing that uses probabilistic models to analyze grammatical structure of sentences.
  - Data driven : Unlike traditional rule-based parsing, it uses probabilities derived from large corpora of lang. data
  - predictive power : It estimates the most likely structure of a sentence based on how often certain struc. appear in training data.

  creating parse tree
  - CFG
  - PCFG (with probability)

  - parse tree generation : output - syntactic / parse tree. Each node represents a syntactic category, edge - grammatical relationship, leaves are words or sentences.
    Learning data : It uses ML to train itself on annotated corpora & generalize to unseen sentences.

Scanned with OKEN Scanner

- **CFG & PCFG :**

- CFG (Context free grammer) : A formal grammer that defines how sentences can be structured using a set of production rule
  The cat sleeps ⇒ The cat = NP , VP = sleeps , NP → Det N, VP → V

- PCFG (probabilistic CFG) : Extension of CFG that assigns a probability to each production rule.
  - Helps choosing most likely parse tree when a sentence has valid multiple parsing
  - Ex: A noun phrase can be constructed by determinary followed by noun / 2 nouns.

CFG	PCFG
S → NP VP	S → NP VP 0.6
sentence → noun → verb	S → NP 0.2
↓ ↓	S → VP 0.2
The cat sleep	

sentence :- The cat sleeps

Steps :

1) Preprocessing
   - Tag words with POS
     - The (Determiner) Det
       cat : Noun (N)
       sleeps : verb (V)

2) apply grammer using PCFG
   S → NP VP
   NP → Det
   VP → V

3) generate parse tree.

- **text categorization :**
  - process of assigning predefined categories or labels to text data
  - How?

1) Machine learning
   - aim : classify text into categories.
     such as topics, sentiments, spam/ham

2) Rule based
   - manually crafted rules, keyword based matching
   - uses dictionaries, regex / pattern matching.

Scanned with OKEN Scanner

- **Morphological Analysis** :
- analyzing the internal structure of words to understand their meaning & grammatical function.
- breaks into morphemes - smallest units of meaning in a language.
  - Free morpheme - Standalone words
  - Bound morpheme - can't stand alone (e.g. -ing in running)
- Root word : core unit of meaning (e.g. happy in unhappiness)
- Affixes : mainly prefix & suffix
- Inflection : changes for tense/number/gender etc (e.g. run becomes runs in present tense)
- Derivation : creating new words using prefix & suffix.
- compounding : combining 2 or more words.

- **Shallow parsing / light parsing / chunking** :
- Technique to identify phrases (chunks) in a sentence without analyzing full gramatical structure.
- Focuses on grouping words into syntactically correlated units such as
  Ex : My dog likes his food.
  - Noun Phrases → My dog
  - Verb Phrases → likes
  - Prepositional Phrases → his food.

- **Deep parsing** : / Full parsing / syntactic parsing :
- Builds a complete hierarchical structure of a sentence, typically represented as a parse tree or dependency tree.
- captures grammatical relations betn all words in sentence (subject, object, etc.)
- Outputs : 1) consistency parse tree (phrase structure tree)
  2) Dependency tree (shows how words depend on each other)
- Ex : my dog likes his food :
  - Identifies POS, e.g My = Determiner, dog = Noun
  - Builds syntactic grouping (phrases) e.g : My dog = noun phrase
  - construct 2 trees

- **Tokenization** : process of breaking up the original text into individual pieces (tokens.
- Types : sentence tokenization, word tokenization, character tokeni

Scanned with OKEN Scanner

- Spacy tokenization concept
  - Features
    - contraction (we're)
    - Abbreviations (L.A., U.S)
    - URLs, emails - recognized as single token.
    - Hyphenated words (snail-mail) correctly tokenized
    - No. with units & currency.

Feature	NLTK (Natural language toolkit)	spacy
main Focus	- Research & flexibility	industry-oriented, Fast, effic
Ease of use	- more loco level, verbose	- High level cleaner.
Speed	- slower	- Very Fast (written in cytho
Tokenization	- customizable, Default-Basic	- Fast, accurate, uses pre-trained models.
POS tagging	- uses classic models	- Neural nws
NER	- weak	- strong
Dependency parsing	- not built in, need libraries	- Built in
Lemmatization	- Basic wordnet based	- accurate, context-awa
sentiment analy	- Requires custom models/ vader	- Not included, but can be added.
Language supported	- mainly English	- most languages with pre trained models
size	- lightwt	- Large due to pre-trained models.

Scanned with OKEN Scanner

NLTK sentence tokenizer types :

Tokenizer	Description	Pros	Cons	Use cases
Punkt Sentence Tokenizer (default)	uses pre-trained data	Handles abbreviat^n well, no aditional training needed	For highly domain specific task, additional custmiz^n may needed	general purpose English text tokenization
Regex Tokenizer	Uses regex to identify sentence boundaries	Fully customizable, suitable for struc. text	Requires creating correct regex for complex cases, can miss linguistic nuances	Domain-specific tokenizat^n (e.g. splitting on bullet points)
Treebank word tokenization	considers lingu. stic structures	Designed for lingu. stic accuracy, Handles punctuati. ons well	mainly for Engl.. Not effective for informal or noisy text	Parsing linguistically accurate data.
Whitespace tokenizer	splits text into sentences based on whitespace	Very fast, Has a simple implemen. tation	ignores punctu^n, Not suitable for complex text	For text having lines separated by new lines or white spaces.

- NLTK word tokenizer types

Tokenizer	Description	Pros	Cons	Use cases
Punkt word Tokenizer	same	Same	Same	same
Regex Tokenizer	Same	same	same	Dates, Email addr., Domain specific
Treebank word Tokenizer	same	same	can't → ca, n't	same
Whitespace tokenizer	Multi-word (e.g. New York)	Preserves multi words	requires pre-defi. ned list	Domain specific
Tweet Tokenizer	tweets & social media	handles hashta gs, mentions, emojis etc.	limited to informal text	tweets & social media

Scanned with OKEN Scanner

- **parts of speech (POS)** :
- categorization of words in a sentence, based on syntactic function & relationship with other words.
- POS tagging: assign specific POS labels to each word in text

POS tag

Coarse POS tag	Fine-grained POS tags
↓	↓
Broad categories - Noun, verb, adjective	more specific sub-categories - plural noun, past-tense verb, superlative adjective.

- POS coarse tags:

Noun (NN), verb (VB), Adjective (JJ), Adverb (RB), Pronoun (PRP), preposition (IN), Conjunction (CC), Interjection (UH): word that expresses strong emotions / exclamations (oh, wow), Determiner (DT): word that introduces a noun & provides context / specificity (the, this, some), Particle (RP): small word associated with a verb to create phrasal verb (wake up, eat in go out)

- Sample fine-grained tags for a coarse-grained tag VB (verb)

Fine grained POS	Defination	Example.
VB	Base form of verb	run, eat.
VBZ	Third person singular present.	runs, eats
VBP	Non-third person singular present	run, eat
VBG	Gerund / present participle	running, eating.
VBN	Past participle	broken, eaten

- **Entities** : specific items or concepts mentioned in text that have a distinct meaning.
  - Named entity recognition (NER) places : karad, person → sharu,
  - NER : process that identifies & classifies named entity in unstru. text
  - spacy auto. identifies entities during text processing.
  - .ents → attribute on Doc object to access entities.

Scanned with OKEN Scanner

- GPE : Geo-political entities e.g. nations

- **Noun & noun chunks** :
  - Noun chunks : group of words that together function as noun.
    - Continuous span of words that acts as noun phrase.

      noun chunk : The big red apple

      - Head : $\underbrace{\text{apple}}_{\text{main noun}}$, modifiers → the big red.

- **Visualizing using Displacy** :
- Displacy : module in spacy library for visualizing NLP annotations such as dependency parse trees & named entities.
- style parameter : allows to decide type of visualization, e.g: dep = dependency parse trees, ent = named entities.
- Visualizations are interactive - we can clicks on tokens to view additional details

- **Phrase / Pattern matching** :
- Finding specific sequences of words (phrases) in a text based on predefined models patterns.
- similar to regex but designed for linguistic patterns.
- spacy tool - Matcher.
- create Matcher object & apply it to a piece of text.

- **Feature extraction from text** :
- ML algo → numerical i/p
  - **count vectorizer / Bag of Words** :
    - Just count how many times each word appeared in the text.
    - converts a collection of text documents into Document-Term-Matrix (DTM) where → Rows = Document.
      Columns = unique words (tokens)
      Values = count of words in that doc.
    - Called BOW : Ignores order, but keeps frequency info.
    - doc 1 : Hello doc 2 : sharmishtha Hello doc 3 = Hey Hello sharu

	Hello	sharmishtha	~~hello~~	hey	~~hello~~	~~sharmishtha~~
doc1	1	0	~~0~~	0	~~0~~	0
doc2	1	1		0		
doc3	1	1		1		

Scanned with OKEN Scanner

- Count vectorization / BoW : Problems :

- Sequence of word is imp. But BoW do not consider the sequence.
- BoW will have same representation for both sentences in DTM form, ignoring order of words.

- Specific words : (sat, lay, played) : Appear in only one doc -> strongly contribute to that document's uniqueness -> High IDF
  ... Excellent for distinguishing betn documents.
- Moderately common words (cat, dog, mat) : appear in some docs but not all .. -> moderate TF-IDF -> useful for classn / clustering
- This represents / ensures the words with high frequency but low relevence do not dominate representation.

- L2-Norm :
  - Formula : $\|V\|_2 = \sqrt{\sum_{i=1}^{n} V_i^2}$

  so, normalized vector V_{norm} will have components $V_{norm,i} = \frac{V_i}{\|V\|_2}$
  - After normalization, the sum of squares of values in new vector will indeed be 1 :
  - The main goal of L2 normalization is to transform any vector into a unit vector. A unit vector is a vector lenght = 1 & preserves directn
    This is crucial for :
    - Fair comparison : long length of doc., high TF-IDF score so for this reason

How? = doc1 : "apple apple orange"

Length (L_2 norm) : $\sqrt{2^2+1^2} = \sqrt{5} = 2.23$

Normalized d1 : $[2/2.23, 1/2.23] \approx [0.891, 0.448]$

doc2 : "apple apple apple apple orange orange"

L2 Norm = $\sqrt{4^2+2^2} = \sqrt{20} = 4.4$

$[4/4.47, 2/4.47] = [0.895, 0.44]$

nearly similar values after normalization.

- makes cosine similarity & other distance-based methods work properly.

Cosin. simi $= \frac{A \cdot B}{\|A\|_2 \|B_2\|_2}$

if $\|A\|_2 = 1$, $\|B\| = 1$ ... normalized, then cosine simil $= A \cdot B$

1) Calculate L2-Norm -> dist -> $\sqrt{A^2+B^2+C^2}$ -> on vector data (L2_norm)
2) Normalize by dividing by A / L_2 norm
3) Calculate Cosine similarity - To know similarity in direction / orientation betn 2 vectors, irrespective of magnitude.

Scanned with OKEN Scanner

- Example:
  - Text : "cat sat on the map"
  - Vector after TF-IDF : [0.5, 0.5, 0.7, 0.2, 0.3]
  - L2 norm = $\sqrt{(0.5)^2+(0.5)^2+(0.7)^2+(0.2)^2+(0.3)^2} = 1.058$
  - Normalize vector = $\left[\frac{0.5}{1.058}, \frac{0.5}{1.058}, \frac{0.7}{1.058}, \frac{0.2}{1.058}, \frac{0.3}{1.058}\right]$
    $= [0.47, 0.47, 0.66, 0.18, 0.28]$
  - new vector squared = [0.222, 0.222, 0.437, 0.036, 0.080]
  - $0.222 + 0.222 + 0.437 + 0.036 + 0.080 = 1$

- **Pipeline class** :
  - sklkit-learn pipeline class : combines preprocessing & modeling steps into a single object.
  - pipeline : A sequence of data processing steps, where each step is a tuple : ('name', transformer/estimator)

```
pipeline = Pipeline([
  ('scaler', standardscaler()),
  ('svm', SVC())])
pipeline.fit(x_train, y_train)
y_pred = Pipeline.predict(x_test)
```

- **Stemming** :
  - Reducing words to their base/root form
  - Simplify words by removing prefixes/suffixes, so that the diff. variations of a word all map to same stem/base form.
  - Improves efficiency & effectiveness of NLP appⁿ
  - Sometimes → mistakes
  - Spacy do not have stemmer, it relies on lemmatization
  - NLTK stemmer types : Porter stemmer & Snowball stemmer.

1) **Porter stemmer** :
  - Process for removing suffixes. , Developed by Martin Porter in 1980.
  - -ed, ing, ion, ions are removed.
  - Uses 5 steps of word reduction, each with it's own set of mapping rules

Scanned with OKEN Scanner

step	original word	modified wr after step.
1a. common suffix removal	caresses	caress
1b. suffix removal based on specific conditions	ponies	poni
1c. Remove the final e if the word has more than 1 syllable	trouble	troubl
2. Remove specific suffixes based on conditions	happiness	happi
3. Remove suffixes based on conditions	agreed	agree.

- <u>snowball stemmer</u> / English stemmer / porter2 stemmer.
  - snowball is a name of stemming language
  - Improves original porter stemmer in logic & speed.
- <u>Lemmatization</u> :
  - Reduces words to their root / base form, called lemma
  - Like stemming, goal: to group together diff. inflected forms of a word so, that they can be analyzed as single common entity.
  - running, runs, ran, ⟶ <u>run</u>
  - Ensures that resulting base form is valid word.
  - Hence, more accurate than stemmer.
  - Better for chatbots, Q & A, text summarization, but not for longer texts (due to higher computational costs.
- <u>stemming vs lemmatization</u> :

aspect	stemming	Lemmatization
method	Rule-based removes affixes	Dictionary-based. consider word meaning.
output	may produce non-words	always produces valid wor
accuracy	Less accurate, may produce incorrect roots	more accurate, context-sensitive.
speed	Faster & computationally more efficient	Slower, computationally m expensive.
use cases	Information Retrieval	Linguistic precision (Text analysis)

Scanned with OKEN Scanner

chunking :

- Dividing words into chunks depending on their syntactic form
- used in shallow parsing.
- chunk is usually a group of words that share a common syntactic role in sentence, such as :
  - Noun phrases (NP) : group of words that f^n as a noun
  - verb phrases (VP) : —| —| —| —| —| verb
  - preposition phrases (PP) : —| —— beginning with preposition.
- Performed using POS tagging as preprocessing step
- NLTK supports flexible chunking
  - Users can use / construct patterns using regular expressions / preset gram.
  - Noun phrases : Built around a noun. → A new laptop
  - verb phrases : Built around a verb. → Bought a new laptop.
- chunks patterns = """ NP : {<DT>? <JJ>* <NN>}
  VP : {<VB.*><NP|PP>} """

NP : uses an optional determiner (DT), followed by zero / more adjectives (JJ), & a noun (NN)

VP : uses a verb (VB followed by any suffix .*), followed by a noun phrase (NP) or a prepositional phrase (PP)

POS tags in output :

VB : verb, base form (e.g, "Let", "Try")

IN : preposition (e.g, "us", "we")

DT : Determiner (e.g., "The")

- Traditional NLP app^n:

i) sentiment analysis : Positive / Negative

- Technique : Rule based technique using VADER (Valence Aware Dictionary & sentiment reasoner) → Specially when dealing with social media text.

i) Lexicon & rule-based approach :

lexicon : relies on pre-built dictionary (lexicon) of words, phrases & even emojis, each assigned to sentiment intensity score.

range : -4 (extreme -ve), +ve (extreme +ve)

ii) Rules : incorporates set of grammer & syntactical rule. acc. to human lang. common patterns.

Scanned with OKEN Scanner

- No training required.
- limitations:
  - sarcasm & irony
  - lack of target identification.

2) Text classification : Label the output.

3) Topic modeling : Finding theme with corpus of text.

- Valence = emotional intensity.

Compound score : normalized value betn -1 to +1, like final score

- Text classification algo:

  Small data (<10k rows) : Naive Bayes

  medium (<100k rows) : Logistic regression, Decision tree.

  Large (>1m rows) : gradient boosted tree/go to transformer

- Universal Networking Language (UNL)
- Frameworks designed to represent & process info in a way that can be universally understood accross languages & systems.
- Relations - agt(agent), obj(object), plc(place), ben(beneficiary)
- aims to represent semantic relationship & concept in stardized way
- UNL represent info as hypergraph, where concepts (universal words) as nodes & relationship betn them as arc. where arc - labeled with specific semantic relation
- structure : headword (contraint1, 2...)
- purpose of constraint : disambiguate meaning of headword, making it precise & universal

  cat (icl animal) : cat is a kind of animal.
- attributes : modifiers that add gramatical / specific semantic details

  @past for tense, @pl for plural.
- BoW (no semantics) vs Word2Vec (semantics)

Feature	BoW	Word2Vec semanti
- meaning	Ignores semantics	Captures word meaning
- representn	Sparse vector (word counts)	Dense vectors (embedd
- content	No content awareness	uses content in training
- output	Document-term matrix	Word embeddings

Scanned with OKEN Scanner

Feature	count vectorizer	TF-IDF	Word2Vec	Transformers
· Type	Frequency-based	Frequency + weighting	Neural word embeddings	Deep context embeddings
· output	sparse matrix of token counts	sparse matrix of wted token counts	Dense vector for each word	Dense vec. for each word, context aware
· contexual	NO	No	No	Yes
· word meaning	Ignores meaning	Ignores meaning	captures semantic similarity	captures meaning in context.
· vector size	vocabulary size	vocabulary size	Fixed (300)	Fixed (768/1024)
· memory efficient.	No	No	Yes	No
· Training needed	No	No	Yes	Yes
· Handle synonyms	NO	NO	NO	To some extent
· content sensitivity	Fixed meaning	Fixed meaning		Diff. vect. per usage.
· Ex. model	CountVectorizer()	TFIDFVectorizer()	gensim.model. Word2Vec or spaCy	BERT, GPT, RoBERTa.
· Limitations	Ignores order and meaning	Ignores order & meaning	semantic similarity	slow, large, need GPU

	Word2Vec	Transformers
	2013	2017
core idea	Static word embedding	Context-aware (Dynamic) word/token embedding
content handling	Learns from local content windows (e.g. words immediately surrounding a content window	- Self-attention mechanism to wt-imp.
word embedding	Fixed, regardless of usage	- Changed based on content.
archit.	Shallow N/N	Deep NN
	Static word embedding	Dynamic token embeddings

Scanned with OKEN Scanner

- Shallow vs Deep parsing.

Feature	Shallow Parsing (chunking)	Deep parsing
Defination	Identifies chunks such as nouns/verbs without full structure	Analyzes complete grammatical structure including all dependencies
Purpose	Provides spatial syntactical structure	Provides detailed syntactic/semantic analysis.
Output	group of words (e.g NP, VP)	Full parse tree with hierarchical grammer relations
speed	Faster	Slower
Complexity	Simple	High
Exmple o/p	"The big dog" → [NP The big dog]	"The big dog barked" → Full tree showing S → NP VP → V etc.
Use cases	Information extraction, NER, POS tagging	Question answering, machine translation, formal reasoning
Tools	NLTK's RegexpParser, SpaCy's noun chunks	Spay dependency parser, standFord parser.

Deep parsing

Syntactic parsing

- analyzing the stru. of sentence based on grammer rules (syntax)
- POS, dependency parsing (who modifies whom) consistency parsing, phrase structure trees, Ex: NLTK, Spacy, BERT

↓

- Analyzes sentence grammer struct. to revel how words relate, using grammer rules to build sparse tree or hierarchy

↓

- Techniques:
  - Dependency parsing: Focus on relationships (depen.) bet words not phrase struct. (Ex: Transition b.)
  - Top-Down: Start from root (start symbol) & tries to generate input (Ex: recursive descent parsing)
  - Bottom-UP: Start from i/p & reduce to start symbol. (Ex: Shift reduce parsing)
  - chart parsing: use dynamic prog. to reduce redundent work (Ex: Early parser)

Semantic parsing

- goes beyond grammer to understand meaning of sentence
- NER, Semantic role labelling (who did what to whom), WSD, que to SQL / Logic transformation in QA system.

↓

- Process of converting natural lang into structured, machine readable represent^n of it's meaning.
- Involves mapping sentences to Formal represe. such as logic based expressions, database queries, executable commands.

↓

- Understanding meaning
- Structure mapping (SQL queries, knowledge graphs (lambda calculus)
- machine usability: enabling downstream task like que.-ans, natural lang. interface

Scanned with OKEN Scanner

• semantic parsing components

1) Input representation : Raw text (e.g : What is population of France?)

2) output representation : Represent text in machine-readable form
   - sql query / lamda calculus / logic expression

3) semantic roles : Identify entities, action, relationships in sentence
   - Tools : semantic role labelling : NLTK, NL to SQL : Hugging face models

• Extractive vs abstractive summarization :

Feature	Extractive	Abstractive
• method	copy from input	Generate new text
• Coherence	medium	High
• complexity	Lower	Higher
• models	TextRank, LSA	BART, T5
• common usage	quick summarization	Conversational AI, GPT

• Extractive text summarization Using TF-IDF :

• Steps :

1) sentence tokenization : split document into individual sentence.

2) Build TF-IDF matrix : So, common words low scores, specific → high score.

3) Score sentences : For each sentence, calculate avg TF-IDF score of the words in that sentence → gives you sentence importance score.

4) Rank & select top sentences → select N top sentence

• <u>Why this works</u>?

Words that are unique & meaningful occur less frequently, sentence containing such words stand out & get higher scores. These are more likely to capture key ideas of article.

• Anaphore resolution : Identify referent of an expression

• coherence resolution : Broader than anaphora resolution
   - Resolves all types of referential relationships in text, including noun-noun references.
     - Ex : The cat slept on the mat. The animal was tired.
       Here animal refers to the cat.

Scanned with OKEN Scanner

- **Pragmatics** :

- Study of how content influences the interpretation of meaning in communication.
- Focus : goes beyond literal meanings to uncover
  - implied meanings
  - speaker intenions
  - situational relevance
- main content : Linguistic (surrounding text), physical (Time, place, env.) social (relationships, Roles, social norms bet° speakers)

- **Discourse analysis** :

- Study of how sentences, phrases & words are structured & connected in communication to form coherent & meaningful units such as conversations, stories / arguments.
- key aspects :
  - coherence : overall logical flow & connectedness of discourse.
  - speech act :
  - Listner effect
  - Turn-taking
  - context
  - presuppositions
  - Narration

- **Semantic web** :

- Extension of WWW that aims to make data on web machine readable so that both humans & machines can understand, share & use it.
- Standards used :
  - RDF (Resource Description Framework) : A standard model for data interchange by representing info. such as triples.
  - OWL (Web ontology language) : Defines ontology, allows for more complexity than RDF
  - SPARQL : powerful lang. to retrieve & manipulate RDF data

- **Statistical / Probability based approaches** :

- statistical models :

1) Vector models : convert text to numerical vectors so that mathematical & ML techniques can be applied, Ex : TFIDF, Word embedding,
Contexual models (BERT)

Scanned with OKEN Scanner

- statistical models (probability models)
- Use probabilities to model language patterns without necessarily representing text as dense vectors
- Focus on likelihood of word sequences
- Ex : n-gram models, Hidden markov models (HMM)

- markove model (why dot product) :
- The current state is a probability distribution over possible situations (like 'It is 70% likely sunny, 20% rainy, 10% cloudy right now'), & transition matrix encodes how states evolve probabilistically.
- multiplying current state vector by transition matrix using a dot produc efficiently calculates the next state probabilities in 1 step.

$X_t = [1\ 0\ 0]$ & $P = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.4 & 0.5 & 0.1 \\ 0.2 & 0.4 & 0.4 \end{bmatrix}$ at $t=0$, we are 100% sunny state.

$X_t \cdot P = [1\times0.6 + 0 + 0,\ 0.3\times1 + 0 + 0,\ 0.2\times1 + 0 + 0]$

$= [0.6, 0.3, 0.2] \Rightarrow$ 60% pehances of sunny, 30% rain, 20% clou

aspect	machine learning (ML)	Deep Learning (DL)
Approach	Relies on hand-engineered features (TFIDF, ngrams)	Learns features auto. fram raw data.
Algo	SVM, Naive Bayes, random Forest, Logistic regression	NN like RNNs, LSTMs, GRUS, Transformers
Feature engineering	manual & domain specific	Not required, the model learns representations
performance	good for small datasets	Excels with large dataset & complex pattems
complexity	simpler models, easier to interpret	complex architecture, header to interpret.
Training time	Faster, requires less computational power	slower, needs high-performance H/w.
Data requirements	Performs well small to medium dataset	requires large amount of labele data for good results
appn	sentiment analysis, text classification	machine translation, que-ans, summarization
Scalability	Limited to Feature scope defined by user	Highly scalable to new task to domains
Ex	-BOW + SVM - TFIDF + NB	- BERT - GPT - Transformer - based models.

Scanned with OKEN Scanner

• sequence-to-sequence (seq2seq) :
- models trained to convert sequence from 1 domain (eglish sentence) to sequence in another domain (French sentence)
- suited when both input & output are sequence of varying lengths.
- key components :
a) Encoder : processes the i/p sequences & encodes it into fixed length vector (context vector/thought vector) using RNN varients like LSTM/GRU
b) Decoder : generates o/p sequence 1 element at a time (uses context vector from encoder & it's own internal state to predict next element in o/p sequence using RNN.
- key components
- Use : Machine state translation from 1 language to another, text summarization, speech recognition, chatbot, image captioning
• BERT : (Bidirectional Encoder Representations from transformers)
- Powerful, developed by google
- Helps computer understand human lang. better.
- Read word according to context.
- older models : Read left to right, BERT : Read entire sentence.
• padding & attension mask :
- padding : make it of fixed length.
- Attention mask : where are the actual tokens.
• Token Type IDs :
- Used to differentiate betn segments in multi-segment input (e.g. Que-ans sentence-pair)
- Helps the model understand which token belongs to which sentence.
- Note : Token type ID are 1 only for token in second sentence
• Speech Processing :
1) Articulatory phonetics : Study of how sound produced
2) Speech sounds : consonants, vowels, supracegmental
3) phonetic transcription : represents speech sounds visually using international phonetic alphabet
4) Computational phonology : applies algo, ML & formal models to study phonological patterns, speech sounds & their rules
5) Digital signal processing : manipulate (audio, speech, images, sensor data) using math & computational techniques.
6) Automatic speech recognition : converts spoken lang. to text.
7) Text-to-speech : Written to spoken.

Scanned with OKEN Scanner

- speech processing
- Language models: ML model that understands, generates, predict text based on learned patterns.
- Types:
  1) statistical language models (traditional)
  2) Neural language models (DL)

- NLP appⁿ:

1) Lexicon in NLP: structured & curated vocabulary / dictionary used in NL processing that contains words along with info as: Meanings (definations), POS, morphological features, synonyms/ants, semantic relations (hypernyms, hyponyms) uses context / syntactic behavior
   - appⁿ: senti. analy, NER, POS, spell check, machine translation, word sence disabiguation, que-ans.
   - Examples of lexicon resources: WordNet, FrameNet, propBank & VerbNet.
2) Dictionary in NLP: A structured collection of words & their associated info such as defination, POS, pronounciation, word freq, usage, morphological forms.
   - Examples: Merriam-Webster API / oxford API, NLTK's WordNet interface, Hunspell & Aspell
3) Thesaurus in NLP: provides synonyms, antonyms & related words, improving lang. generation & text understanding
   - appⁿ: Text summarization, Paraphrasing & text writing, keyword expansion in search engines.
   - Ex: Roget's Thesaurus: used for synonym extraction
     WordNet synsets: group words by similar meaning
     ConceptNet: knowledge graph that includes synonym relations.
4) Transliteration: process of converting text from one script into another while preserving phonetic sounds.
   - focus on conveying meaning, aims to represent pronounciation of words in diff. script.
5) Grammer check:
   - NLTK not support grammer check
   - can use library → Language tool.

Scanned with OKEN Scanner

OpenCV : color spaces :
- Loads img in BGR format by default.
- convert to other color spaces using cv2.cvtcolor()
- common conversions :
  - gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  - rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) → simplifies edge detection & filtering.
  - hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV) → Good for object tracking (color detection)

- HSV : (Hue, Saturation, Value)
- Hue (H) : Represents pure color itself, 0°/360° = Red, 60° = Yellow, scaled in OpenCV to a range of 0-179 to fit within 8-bit integer (179 corresponds → 359 degrees)
- Saturation (S) : Represents purity/intensity of color. Indicates how much of color mixed with white/gray, 0 = completely desaturat (that color is gray), 255 = Fully saturated → most vibrant colo
- Value (V) : Brightness of color (0 = pure black, 255 = max brightn

- Opencv : Arithmetic & Bitwise Operations :
- pixel manipulation allows us to directly modify img data for
  - Image enhancement
  - precise control over region of interest & mask manipulation.
- Types :

1) Arithmetic operations
- perform mathematical oper. on pixel values using constants/ betn 2 images
- adjust brightness (add/subtract) enhancing contrast (multiply/divide)
- Img blending & overlay

2) Bitwise operations :
- Apply logical operations on individual bits of pixel value
- masking specific regions, combining binary images, shape extraction.

- Template matching limitations & soln :
- requires same size & orientation - Templates must match scale & angle of object in source img, fails under real-world variation.
- object rotation, zoom, lighting changes, or partial blockage will re accuracy.
- No semantic understanding.
- purely based on pixel similarity
- Alternative : Feature-based matching, object classn, Fast but appro

Scanned with OKEN Scanner