classmate
Date
Page

Unit III

Language Modelling.

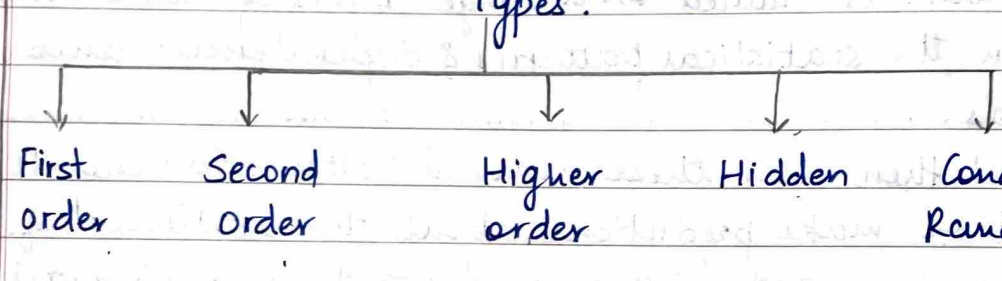
Probabilistic language modeling.

- statistical approach used in NLP & ML.
- used to model the probability distribution of words or sequences of words in a given text.
- involves using probabilistic techniques to estimate the likelihood of different words or sequences of words occurring in a language
- A model is trained on a large corpus of data (text) to learn the statistical patterns & dependencies among words.
- Model then uses these learned patterns to generate new text or make predictions about the likelihood of a particular word / sequence of words occurring given some context.
- one popular approach is n-gram modeling.
- Model considers the probability of a word / sequence of words based on the occurence of previous n-1 words.
- Other commonly used approach is the use of RNNs & variants like LSTM & Gated Recurrent Units (GRU) which are capable of capturing long-term dependencies in text data.
- App→ speech recognition, sentiment analysis, etc.
- It is a fundamental concept in NLP.

Markov Models.

- Type of probabilistic models used in NLP.

- Markov models assume that probability of a word depends on a fixed no. of ~~words~~ previous words.
- also known as the order of Markov Model.
- It means the future word is only influenced by a limited history of previous words, not the whole history of text.
- this assumption makes Markov models computationally efficient & widely used.
- there are several types depending on the order of the Markov assumption i.e. no. of prev words which influence the prediction of new word.



1) First order.
- also known as unigram model.
- probability of a word depends on the frequency of the word itself & not on previous words.
- assumes each word is independent & has no memory of previous words.

2) Second order
- also known as bigram model.
- probability of word depends on the frequency of previous word.
- considers the last one word as context for predicting the next word.

3) Higher order
- also known as n-gram models.

- probability of a word depends of on previous n-1 words.
- Eg. a trigram model considers last two words & as context for predicting the next word.

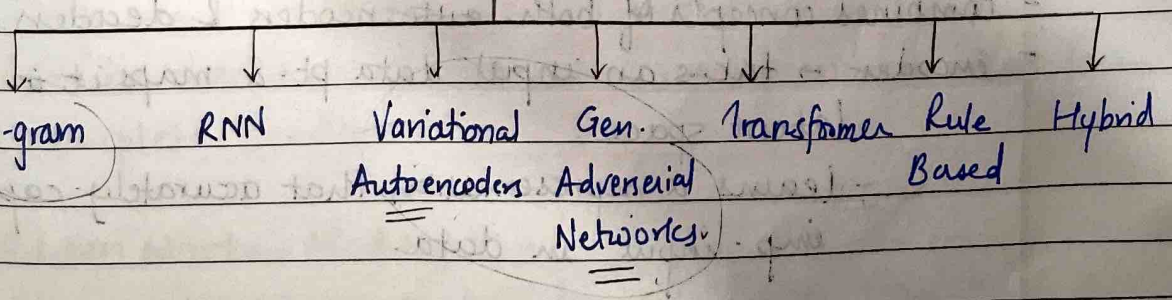
4) Hidden.
- each word (state) generates an observation, but the word itself (state) is not observed.

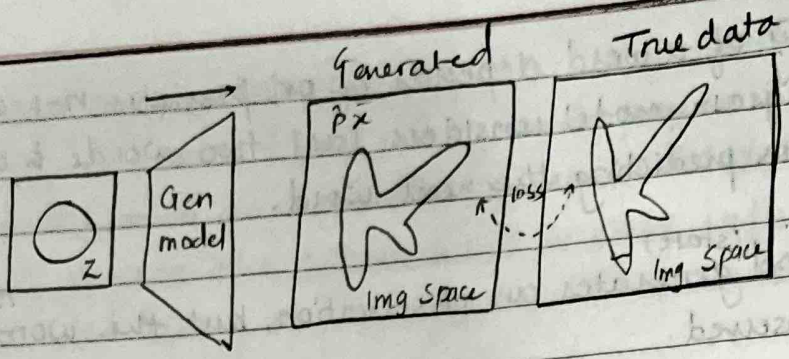
- Markov models can be used to estimate the probability of a word or sequence of words occuring in the text based on their frequencies in the training data.
- can be used for text generation by sampling text words based on their probabilities.
- used in text prediction, speech recognition & POS tagging

Generative Models of Language.

- models that are capable of generating texts (new) which is similar to training data they were trained on.
- they learn the underlying traits & statistical patterns & structures of training data.
- using that, they generate new text that resembles the original data.
- widely used in NLP for various tasks such as text generation, machine translation, etc.

Types:





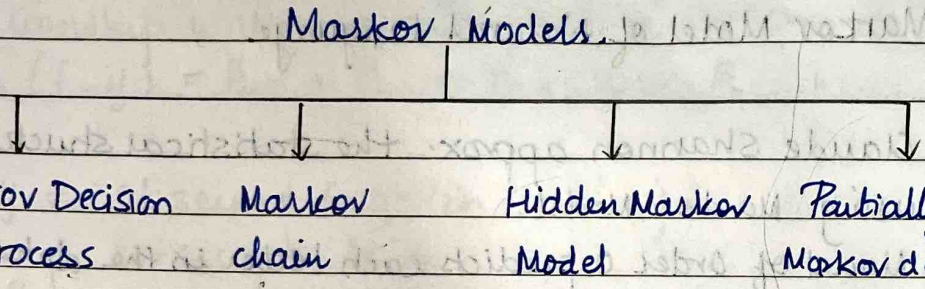
1) GAN.
- 2 components : generator & discriminator
- Both components are trained in a competitive manner.
- Generator is responsible for producing synthetic data from the input it recieves & fool the discriminator.
- The discriminator is responsible for distinguishing fake samples from the real one.
- Every time the discriminator notices changes in the two samples, it adjusts its parameters slightly to make it go away.
- At the end (in theory) the generator exactly produces the true data/~~dist~~ data exactly similar & indistinguishable from the real data sample & the discriminator guessing at random will be unable to find a difference.

2) VAE (Variational Autoencoders)
- main idea is to learn a low-dimensional latent space that captures the essential features of ip data.
- latent space is then used to generate new data samples that resemble original data.
- combines concepts of both ~~auto~~encoders & decoders.
- encoder → takes an input data pt. & maps it ~~is~~ to the latent space.
  - learns a representation that accurately captures the imp. input in data.

- decoders - take sample from the latent space & reconstruct the original data.
  - generate outputs closely resembling the input data.

Markov Models.

- unprecise model used in system which does not have a fixed pattern of occurence.
- In Markov model, it is assumed that the future state only depend on the current state of the system, not the past states.



1) Markov Decision Process.

- used to formalize reinforcement learning problems
- extension of Markov Chains
- sequential Decision Making Process.
- has finite states, finite rewards & finite actions.
- tuple of 4 elements (S, A, P_a, R_a).

2) Markov Chain.

- random ~~chain~~ process in which occurence of future stat is dependent on the ~~mmm~~ current state only & only.
- can be used to generate sequence of words that form a complete sentence

3) Hidden Markov Model.

- every individual state has limited no. of transitions &

→ emmissions.
- Probability is assigned for each transition bet" states.
- ∴ Past states are completely independent of ~~current~~ future state.
- It is called hidden due to its ability of being memory less.
- main goal is to learn about a Markov chain by obs. its hidden states.
- Considering Markov Model X with Y hidden states, for each time stamp, the probability distribution of Y is not dependent on the history of X according to that time.

Markov Model of Natural language

- Claude Shannon approx. the statistical structure of text using Markov Model.
- Model of order 0 predicts each letter in the alphabet occurs with fixed probability. order 0
- A Markov model can be fit to any specific piece of text by
  → counting occurrences of each alphabet & using them as probabilities.
- Eg. i/p → aachggahdehh, M.M. of order 0 will predict P(a) = 3/12 = 1/4, P(c) = 1/12 P(g) = 2/12 = 1/6 & so on
- An order 0 model presumes that each ~~model~~ letter is chosen independently.
- We obtain more refined model → P. of choosing each successive letter depending on the previous letter/letters.

Log Linear Models.

- widely used in NLP.
- key advantage is flexibility.

- statistical model used to analyze relationships between categorical variables.
- In log-linear models there are 2 variables, dependent & independent.
- dependent: contingency table representing frequency/count of obs. in diff categories
- independent: categorical ~~values~~ variables determining structure of contingency tables.
- goal is to estimate parameters of model that best fit the observed frequencies in the table.
- assumes that natural logarithm of cell follows a linear relationship with predictors.

$$ln(f_{ij}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n$$

f_{ij} = observed freq in cell (i, j)
β_0 = intercept term
$\beta_1, \beta_2 \ldots$ = coefficients associated with independent values $x_1, x_2 \ldots x_n$ respectively.

N gram model.

- statistical model.
- predicts the probability of words or sequence of words based on previous n-1 words in a text.
- widely used in NLP for text generation, speech recog. etc.
- 'n' in n-gram → no. of words / tokens.
- n = 1 → unigram → considers each word indep.
  n = 2 → bigram → pairs of consecutive words.
  & so on.
- higher value of n capture more context & provide better language modelling accuracy.

- By considering that n-1 are useful for predicting a given word, n gram model can be defined as:

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

- based on Markov assumption that current words only depend on n-1 preceeding word.
- ∴ n-gram is also known as $(n-1)^{th}$ order Markov Model.
- a large amount of data corpus is taken as training data.
- it has the curse of dimensionality. growth of n grams as n ↑.
- do not capture long-term dependencies.
- capture local dependencies.

Parameter Estimation & Smoothing.

Estimation

- involves calculating the probabilities of n-grams based on observed frequencies in training data.
- counts can be obtained by scanning training corpus & tallying occurrences of n-grams.
- max. likelihood estimation: probability of n-grams is estimated by dividing count of that n-gram by its (n-1) gram prefix.

$$P(w_n | w_{n-1}) = \frac{\text{Count}(w_{n-1}, w_n)}{\text{Count}(w_{n-1})}$$

- This causes sparse data problems.
- Smoothing is applied to address the issue.

Smoothing.

- used to handle data sparsity.

- assigns non-zero probabilities to unseen n grams.
- helps in generalizing the model & making predictions for previously unseen contexts.
- common technique - back off.
- involves splitting n grams whose count in training data fall below predetermined threshold & also those whose count exceeds it.

Large Scale Models.

- With increase in monolingual data scaling of language → it is reqd. to handle sets of billions / trillions of words.

Evaluating language model.

- best way to include it in an application.
- check the performance of that application.
- called extrinsic evaluation.
- → expensive at times.
- ∴ intrinsic evaluation is used.
- intrinsic → a metric is used to quickly evaluate the improvements in language model.
- citeias used in intrinsic evaluation:

1) Coverage Rate
- measures % of n-grams in test set
- Sometimes some unknown words appear, they are called out of vocabulary words.
They cannot be handled by this type.

2) Perplexity
- considers the fact that among two probabilistic models the one which fits the data (test) is the better one.

When we apply an algo. in NLP it works on numbers. We cannot directly feed text into that algo.
∴ BOW is used to preprocess data by converting it into bag of words which keeps count of occurrences of freq. used words.

classmate Date

Bag-of-Words

- used to simplify & represent text data
- doc/text is represented as unordered collection of words, disregarding grammar, word order & context -
- focuses solely on the presence / absence of words in text & their frequencies.
- used to convert text into fixed length vectors.
- focuses on two things -
  1) Vocabulary of known words
  2) Measure of presence of these known words.
- only considers frequency of words.
- Working :

1) ~~Tokenization.~~
   - ~~the text is split into individual words.~~

1) Data collection.
   - First step is to collect data.
   Eg. The dog sat
   The dog sat in the hat
   The dog with the hat.

2) Tokenization.
   - The text is split into individual words.
   - involves removing punctuation, converting all to lowercase
   The example, words/tokens are
   The, dog, sat, in, ~~the~~, hat, with.

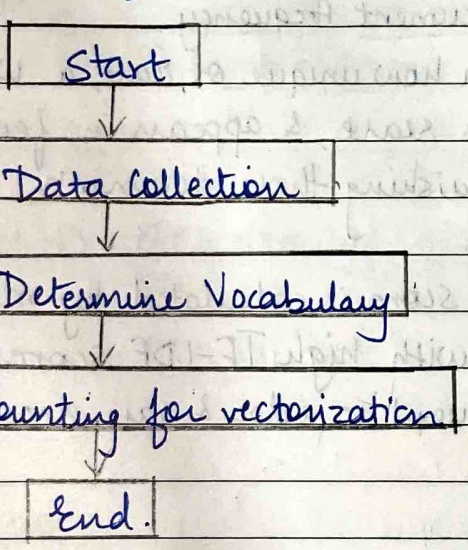
3) Word counting
   - for each text/doc, vector representation is created based on the vocabulary.
   - vector has same length as vocabulary.
   - each element represents count of specific word in doc.
   Eg. The dog sat [1, 1, 1, 0, 0, 0]

	the	dog	sat	in	hat	with	cheese
①	1	1	1	0	0		
②	2	1	1	1	1		
③	2	1	0	0	1	1	

The dog sat in the hat [2,1,1,1,1,0]
The dog with the hat [2,1,0,0,1,1].

4] Vector representation:
- resulting vector → bag of words vector.
- can be a sparse vector.

- used for feature extraction
- can be visualized using table which contains count of words corresponding to word.
- approach is simple & flexible.
- Steps:



Disadvantages:
1) model disregards order of words.
∴ It loses info. about sentence structure, grammar & word dependencies.
2) Contextual info.
- model doesn't capture context in which words are used
- Treats each word independently, ignoring the relationships bet^n words.
3) range of vocabulary.

TF-IDF

- Stands for Term Frequency Inverse Document Frequency
- calculation of how relevant a word in a ~~text~~ corpus is to a text.
- it considers two things: 1) how often a word appears in a doc
  2) how rare / unique that word is across all documents.

TF & IDF.

- **Term Frequency.**
  - tells you how often a word appears in a document.
  - If a word appears multiple times, it is likely to be more important for that document.
- **Inverse Document Frequency.**
  - tells you how unique or rare a word is across all documents
  - if word is scarce, & appears in few docs, → more important in distinguishing those documents.

- TF-IDF score is calculated by multiplying TF & IDF val
- ∴ words with high TF-IDF scores are ones that are both frequent in a doc & rare across entire collection of docs.

---

* Key concepts:

1) **Term Frequency.**
   - no. of instances of a given word t in doc. d.
   - for each ~~t~~term in doc → an entry exists with value of TF.

$$tf(t,d) = \frac{\text{count of } t \text{ in } d}{\text{No. of words in } d.}$$

---

2) **Document frequency.**
   - count. of occurrences of ~~of~~ a word t ~~in~~ across a series of docs.

$df(t)$ = Occurrence of t in documents.

3) Inverse Document frequency

- tells how relevant the word is.
- as tf considers all terms equal, it is not possible to use it to find weight of words in doc.
- first, find doc. frequency of term t by counting no. of docs containing term.

$$df(t) = N(t)$$

↑ No. of docs cont. t.

$$idf(t) = \frac{N}{df(t)} = \frac{N}{N(t)}$$

* More common word is considered less significant.

$$tf\text{-}idf(t,d) = tf(t,d) * idf(t)$$

* Applications

1) Document classification
helps in classifying the type & genre of document.
2) Topic Modelling
helps predicting topic for a corpus.
3) Information retrieval
extract important info. from a corpus.
4) Stop word filtering.
helps in removing unnecessary words from text.

Word Embedding.

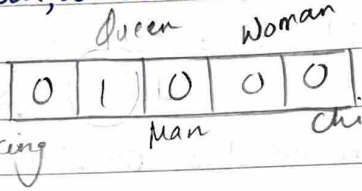
- numerical representation / vector representation of words.

Word Embedding.

- Frequency based
- Prediction based.

1) Frequency based.

- Vector of size equal to vocabulary.
- each element in vector corresponds to word from vocabulary.
- encoding of given word is vector in which corresponding element is set to 1 & all others 0.
- Eg - King, Queen, Man, Woman & Child.
  to Encode queen,



- one hot encoding.
- very ineffective & not smart.
- as vocab increases, size of vector increases ∴ memory space reqd. also increases.

2) Prediction based.

- popular in NLP.
- give probabilities of words.
- able to achieve algebraic operations tasks.
- Word2Vec & Doc2Vec.

Word2Vec. (Algo)

- It is a prediction based embedding.
- technique for NLP.
- uses neural networks model to learn word associations in a huge corpus of text.
- the trained model can detect synonymous words or suggest

additional words for partial sentences.
- represents each distinct word with a numeric value called vector.
- vectors - chosen carefully so that they capture semantic & syntatic qualities of words.
- cosine function can indicate the level of semantic similarity between words represented by those vectors.
- Word2Vec utilizes either of 2 architectural models to produce vectors:

1) Continuous Bag of Words (CBOW)
2) Continuous skip gram.

- In both, word2vec considers both individual words & sliding window of context words surrounding the individual words.
- In CBOW, model predicts current word from window of surrounding context words.
- order of context words does not influence prediction.
- In continuous skipgram, model uses current word to predict surrounding window of context words.
- skipgram: weighs nearby context words more than distant context words.
- CBOW faster, skip gram is better for infrequent words.
- After model is trained, vectors are placed in such a way that words that share common context are placed closed & more dissimilar ones are placed away from one another in the space.

Doc2Vec.

- extension of Word2Vec.
- neural network based algo for generating vector

representations of docs/paras.
- allows to capture semantic meaning & contextual info of an entire document rather than individual words.
- estimates dist representation of documents just like word2vec estimates ^ representation of words.
- 2 architectures used:

1) Distributed Memory Model of Paragraph Vectors (PV-DM)
- identical of CBOW.
- provides unique document identifier as piece of additional context.

2) Dist. BOW for Paragraph Vector. (PV-DBOW)
- identical to skip gram model
- except that it attempts to predict window of surrounding context words from para identifier instead of current word.

- can capture semantic context / meanings around words.

BERT.

- stands for Bidirectional Encoder Representation from Transformers.
- NLP model introduced by Google in 2018.
- used for question answering, sentiment analysis etc
- BERT utilizes a transformer architecture
- it allows it to consider the context of the word by looking at the previous & final ~~word~~/following words.
- this directional nature enables it to capture more dependencies & relationships between words.
- BERT is pretrained on a large corpus of ~~data~~ unlabeled text data.

- learns to predict masked words within a sentence & understand the relationship between different sentences.
- after pre-training, BERT can be fine-tuned by training it on task-specific labeled data.
- allows Bert to adapt to different NLP tasks.
- BERT is pretrained simultaneously on 2 tasks:
1) Language modeling
2) next sentence prediction.
- As a result of training process, BERT learns latent rep. of words & sentences in context.
- After pre-trained, BERT can be fine-tuned with fewer resources on smaller datasets to optimize its performance on NLP tasks, sequence-to-sequence based lang. generation.
- pretraining > fine tuning (expensive)

Topic Modelling.

- recognizing words from topics present in the document / data corpus.
- useful as extracting words from docs takes more time & is complex than extracting them from topics in doc.
- Eg. there are 1000 docs & 600 words in each. To process this it req. 600×1000 = 6,00,000 threads.
- When we divide document containing certain topics, if there are 5 topics present, processing → 5×600 = 3000 threads.
- imp. topics take that make text processing easier in NLP:
1) Removing stopwords & punctuation marks.
2) Stemming.
3) Lemmatization
4) Encoding.
- Stemming involves reducing the words in a corpus to root form.

- types: Porter, Lancaster & Snowball.
- Topic modelling is done using LDA
- Topic modelling refers to identifying topics that best describe set of documents.
- topics emerge only during topic modelling process.
- unsupervised approach of recognizing & extracting topics.
- done by extracting patterns of word clusters & tf in the document.

Documents → LDA → Creation of topics → Topic alloc to documents.

Latent Dirichlet Allocation.

- probablistic generative model.
- assumes that docs are represented as mixture of topics.
- each topic is characterized by distribution over words.
- underlying intuition is every document exhibit multiple topics, each word in a doc. is generated based on the topics present in the document.
- aims to uncover these latent topics & corresponding word distributions.

Steps.

1) Specify no. of topics K → determines no. of topics to be discovered in the corpus.
2) Provide collection of docs you want to analyze. Preprocess by tokenization.
3) LDA uses iterative process to estimate topic-word & doc-topic est distribution.
   algo. randomly assigns each word to a topic.
4) Iterative updating the topic assignment for each word continues till convergence / specific no. of iterations reached.

5) Once model converges, examine topic-word & document-topic distribution.
- has various apps like document clustering, info retrieval, text summarization etc.

Latent Semantic Analysis.

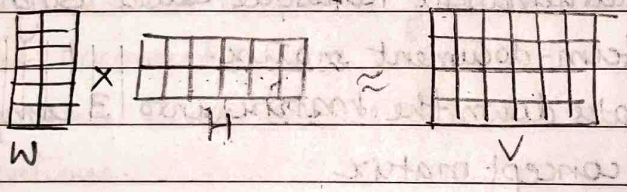
- technique used in NLP.
- analyzes relationship between documents & terms they contain.
- aims to capture the latent / underlying semantic meaning of words & docs by representing them in a lower dimensional space
- it assumes that words with similar meanings tend to appear in similar contexts.
- process involves constructing term-doc matrix where, row = term ; Column = document.
- entries in matrix indicate frequency of term within a doc.
- applies mathematical technique called Singular Value Decomp. to the term-document matrix
- SVD breaks down the matrix into 3 components:
  1) term concept matrix
  2) singular ~~no~~ value matrix
  3) document concept matrix.
- term concept & document concept matrices capture underlying semantic relationships bet^n terms & documents.
- Documents are compared by cosine similarity betn any two columns
- Values: 1) close to 1 → ~~words~~ very similar docs
  2) close to 2 → very dissimilar documents.

* Applications:

1) Compare docs in low-dim space.
2) Find similar docs across languages after analyzing base set of translated docs.
3) Find relations between terms.
4) Analyse word association in text corpus.

Non-negative Matrix Factorization.

- statistical method that helps us to reduce the dimensions of input corpus.
- it uses factor analysis method to give comparatively less weightage to words have less coherence.
- group of algorithms in ~~ma~~ multivariate analysis and linear algebra.
- Matrix V is usually factorized into 2 matrices - W & H.
- all three matrices have no negative elements.
- non-negativity makes the resulting matrices easier to inspect.

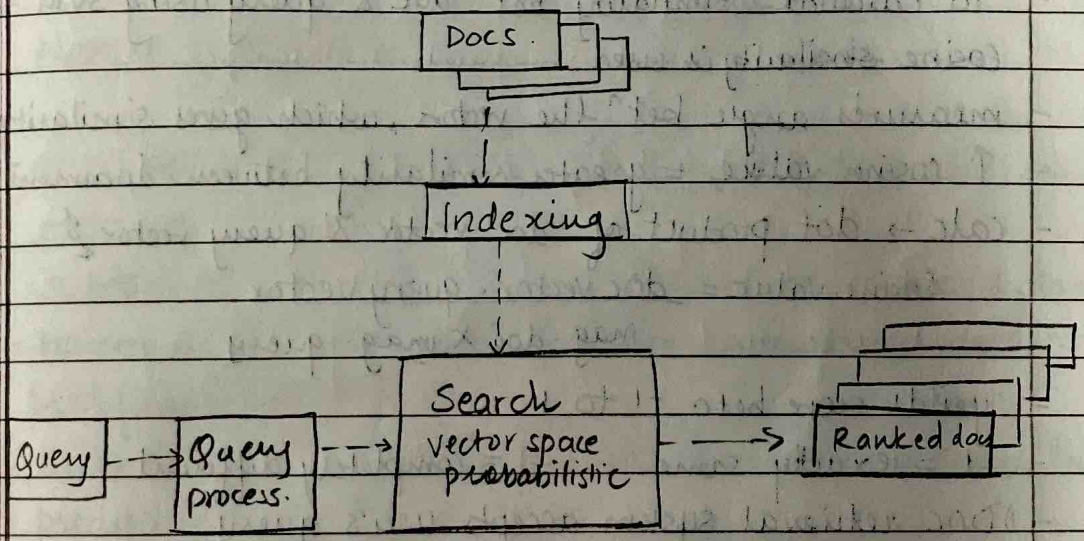


- problem is not exactly solvable, ∴ it is approximated.
- app: numerology, astronomy, comp visions, document clustering, bioinformatics etc.

Unit IV
Information Retrieval Using NLP.

Information Retrieval.

- process of obtaining relevant information from a large collection of data or documents.
- involves searching for & retrieving info that matches a user's queries or info need.
- focuses on storage of text documents & retrieval.
- a word in a doc is a unit of text in the system.
- this unit is indexed & ready for retrieval.
- terminologies used:
  collection → set of docs to satisfy user queries
  term → lexical item in collection.
  query → user's information need in set of items.



1) Collection of documents is analyzed & organized to create an index.
2) The user expresses Info. need through query - set of search terms / question. → Entered into search engine
3) Search engine / database system retrieves relevant

documents based on the query.

4) Retrieved docs are ranked on the basis of relevance to the query.

5) Relevant documents are presented to the users in a meaningful way, such as list of search results.

Vector Space Model.

- documents & queries are represented as vectors in hd space
- vectors consists of the features which represent words in the collection.
- each dimension corresponds to a particular term/word.
- In SVM, each doc & query is represented as a vector & length of vector = total no. of items in the collection.
- value of each dimension in vector represents importance / presence of corresponding term in the doc/query.
- To calculate similarity betⁿ doc & query using SVM → cosine similarity is used.
- measures angle betⁿ the vectors, which gives similarity.
- ↑ cosine value = greater similarity between document & query
- calc → dot product of doc vector & query vector &

$$\text{Cosine value} = \frac{\text{doc vector} \cdot \text{query vector}}{\text{mag} \cdot \text{doc} \times \text{mag} \cdot \text{query}}$$

- yeilds score betn -1 to 1.
- 1 = exactly same -1 = completely different.

1) Doc retrieval system accepts user's query
2) creates vector representation.
3) compare it with all documents
4) result is sorted, which is a list of ranked documents according to similarity with query.

Evaluation of IR systems.

1) Precision 2) Recall.

Precision → fraction of returned relevant docs.

Recall → fraction of all possible relevant docs contained in ~~doc~~ return set.

- Let's consider

T → Total Ranked doc. related to query

R → Relevant documents from T

N → Remaining Irrelevant docs

U → Relevant docs as a whole for the query.

$$\text{Precision} = \frac{|R|}{|T|}$$

$$\text{Recall} = \frac{|R|}{|U|}$$

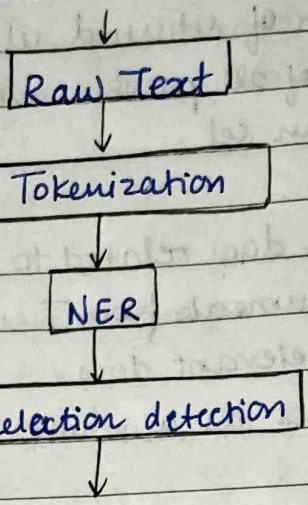
Named Entity Recognition

- also known as entity identification, entity chunking, entity extraction.
- subtask of info. extraction that seeks to locate & classify named entities ~~mea~~ mentioned in unstructured data to predefined categories.
- automatically identifies entities & classifies them into predefined categories.
- entities can be names of people, organizations, locations, times, quantities etc.
- Eg. In the sentence "Steve Jobs founded Apple in USA" we identify 3 types of entities:

<Person> → Steve Jobs

<Company> → Apple
<location> → USA.

- recognition of these entities can be done through ML & NLP.



NER in NLP.

- with NER, one can extract key info to understand what a text is about.
- API's for NER. (Types).

1) Open Source NERs.
free & flexible & entail a gentle learning curve.

a) Stanford NER
- Java tool developed by Stanford Uni
- standard library for entity extraction.
- based on conditional Random Fields
- provides pretrained models for extracting person, organization, location etc.

b) Spacy.
- Python framework
- has excellent statistical system to build customized & NER

c) NLTK
- Suite of libraries for python
- widely used for NLP tasks.
- has its own classifier to recognize name entities called ne_chunk.

2) SaaS NERs.
- ready to use, low-code & cost effective.
- easy to integrate with platforms
- Monkey learn.

NER system building Process.

1) Data Collection & Annotation.
- collection of representative dataset that contains text docs containing named entities relevant to app. or domain.
- Annotate the dataset by manually labelling the named entities & assigning entity type.

2) Pre-processing.
- Split the text into individual tokens to form basic units of analysis. (tokenization)
- Divide the text into separate sentences if necessary (sentence segmentation)

3) Feature Extraction.
- extract relevant features from the text to aid recognizing named entities.
- features → neighbouring words, POS tags, word embeddings etc.

4) Model Selection.
- choose appropriate NER model architecture based on the req. & resources.
- commonly used → rule based, statistical models (CRFs) & DL models (RNN, BERT or GPT).

5) Training
- split the annotated dataset into training & eval. dataset

- Train NER model using training data.
- Here the model learns to classify ~~NER~~ named entities based on provided features.

6) Evaluation
- Evaluate the trained NER model using test data to assess performance.
- Common evaluation metrics → precision, Recall, F1 score.
- These measure the model's ability to correctly identify & classify named entities.

7) Deployment
- Once NER achieves satisfactory performance, integrate it into your desired application or pipeline.
- Apply NER model to new, unseen data to recognize & extract named entities automatically.

Evaluating NER system entity extraction.

1) Evaluation Metrics.

a) Precision.

measures how many of the system's pred. entries are actually correct.

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

b) Recall.

measures how many actual entries are correctly identified by the system / correctly labelled ÷ actual labellable entities.

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

c)

c) F1 score.
- harmonic mean of precision & recall.
- balanced measure of system's performance

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

Relation Extraction.

- NLP process
- involves identifying & extracting relationships between entities in a text.
- goal is to ~~specify~~ determine specific relationship that exists between two or more entities.

1) - first, the relevant entities in the text are identified & extracted.
- Entities can be people, locations, organizations etc.

2) After extraction of entities, the text is parsed to determine syntatic dependencies bet^n words.
- help to understand sentence structure

3) After understanding sentence structure, next step is to classify relationships bet^n entities.

4) Finally, the classified relationship is extracted from text, along with corresponding entities.

- can be used for Q-A systems, sentiment analysis, info retrieval.

Reference Resolution.

- also known as anaphora resolution.
- identifying & resolving ~~relations~~ the references/pronouns in text to their corresponding antecedents.

- aims to determine what a pronoun, noun phrase etc are referring to expression is referring to in the context.
- referred entity is called reference.
- reference to an entity which is previously introduced is called anaphora
- the refering expression is called anaphoric.
- Two refering expressions used to refer to same entity is called corefer.
- Typically involves 2 tasks:

1) Coreference resolution
task to find out refering expressions referring to same entity.

2) Pronomial anaphora resolution
- task to find out antecedent for single pronoun.
- subtask of coreference resolution.

Algos.
1) log linear 2) Centering 3) Hobbs.

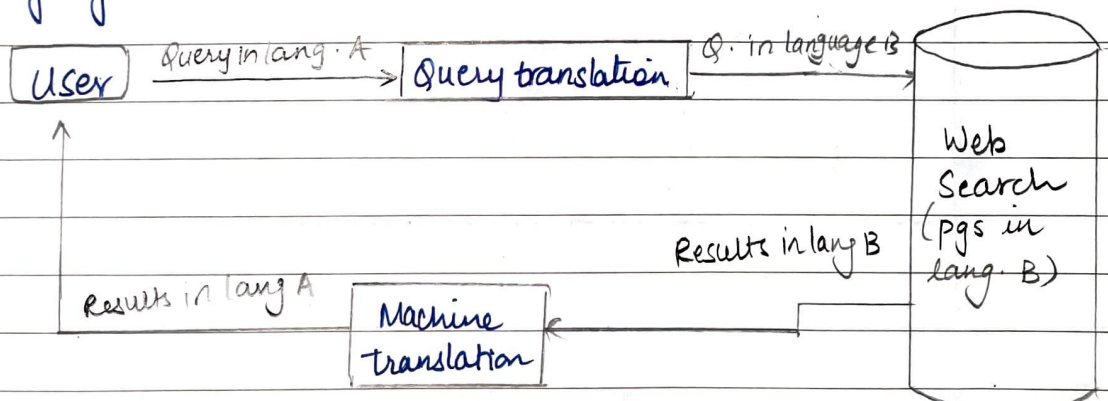
Coreference Resolution:

- task of finding all ~~resolu~~ expressions refering to the same entity.
- imp. step in doc summarization, QA ing & info. extraction.
- process of resolving pronouns to identify which entities they are pointing/refering to.
- entities resolved may be person, location, event etc
- Eg. John went to the store. He bought some groceries'.
In this case, 'John' & 'He' both refer to same person, coreference resolution aims to establish this relationship.
- goal is to improve understanding of text by creating links or associations bet^n different mentions that refer to same entity.

- involve analyzing linguistic features such as pronouns, nouns etc
- used in info extraction. Q-A & summarization.

Cross lingual Information Retrieval (CLIR)

- 'Around 7k languages exist in the world, out of which only very few are used across globe.
- English has greatest presence across internet.
- Huge amount of e-text is being generated these days.
- CLIR is process of retrieving relevant info from a doc. written in one language, given a query expressed in another language.
- aims to bridge the language barrier & enable users to access info. in languages they might not understand.
- The users can search documents databases in multiple languages & retrieve info in a form that is useful to them, even though they don't have linguistic competance in that language.



- documents in target language are converted to source lang.
- Doc translation provides more accurate translation due to richer contexts.
- requires huge resources & as translation needs to be done online.

Challenges.

1) Translation ambiguity
   - one or more translation is possible of the source sentence.
2) Phrase identification
   - in some languages words are not separated by white spaces.
   - may lead to the incorrect translation.
3) OOV problems
4)