Natural Language Processing (NLP)

Presentation · April 2024

DOI: 10.13140/RG.2.2.13534.04169

1 author:

Mohana M
Avinashilingam University
23 PUBLICATIONS **29** CITATIONS

SEE PROFILE

Natural Language Processing

Presented by

M.Mohana

What is NLP?

- NLP stands for Natural Language Processing
- Branch of artificial intelligence (AI) that focuses on the interaction between computers and human language
- Helps computers to understand human language and also allows machines to communicate with us.
- Algorithms and models that enable computers to understand, interpret, and generate human language in a meaningful way.
- For instance, Google's keyboard suggests auto-corrects, and word predicts in email writing (words that would be used).
- Translation systems use language modeling to work efficiently with multiple languages.

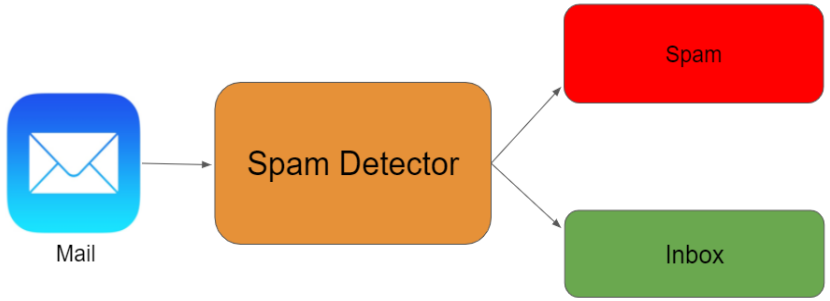


Image Source: Google

How does Natural Language Processing work?

- Converting unstructured data into computer-readable language by NLP attributes.
- Complex algorithms to break down any text content to extract meaningful information from it.
- Collected data is then used to further teach machines the logic of natural language
- Uses syntactic and semantic analysis to guide machines by identifying and recognizing data patterns.

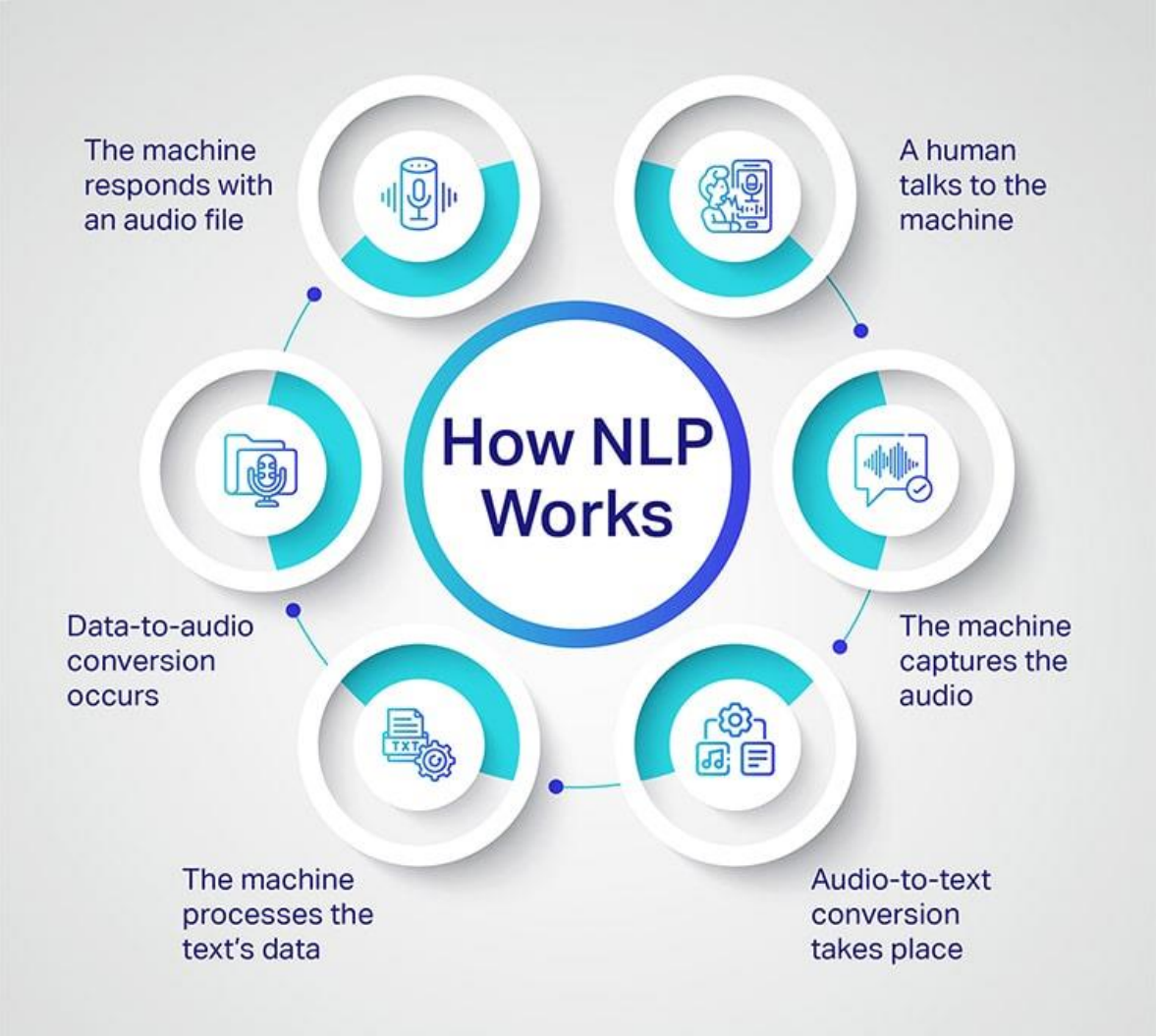
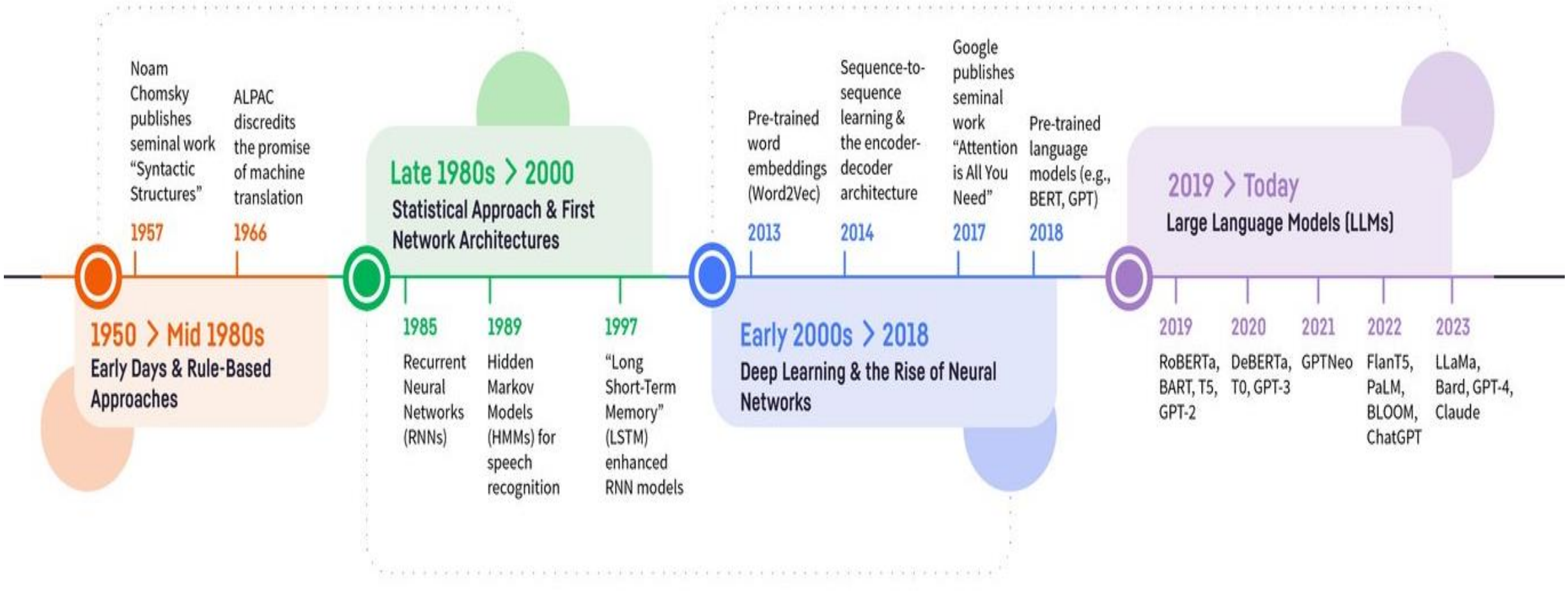


Image Source: Google

Why natural language processing is important?

- **Facilitates Human-Computer Interaction**: users to communicate with devices, applications, and systems using spoken or written language
- **Text Understanding and Analysis**: enables organizations to analyze customer feedback, social media content, news articles, and other textual information to understand trends, sentiments, and patterns
- **Automates Routine Tasks**: asks such as text summarization, sentiment analysis, document classification, and information extraction, improving efficiency and productivity.
- **Enhances Search and Information Retrieval**: search engines and information retrieval systems by understanding user queries and returning relevant results
- **Supports Multilingual Communication**: machine translation and language understanding capabilities, NLP facilitates communication across different languages
- **Enables Sentiment Analysis and Opinion Mining**: sentiment analysis analyzes and interprets the sentiment or emotion expressed in text data
- **Empowers Chatbots and Virtual Assistants**: understand user queries, provide information, perform tasks, and offer personalized recommendations, enhancing customer service and user experience.

History of NLP



https://blog.dataiku.com/nlp-metamorphosis

Components of NLP

Natural Language Understanding (NLU)

- NLU focuses on interpreting and extracting meaning from human language input.
- It involves techniques such as text parsing, entity recognition, sentiment analysis, and intent detection.
- NLU systems aim to comprehend the content of text or speech input to extract relevant information and understand the user's intentions or queries.
- Examples of NLU applications include chatbots that understand user queries, sentiment analysis tools that analyze emotions in text, and voice assistants that interpret spoken commands

Natural Language Generation (NLG)

- NLG, on the other hand, deals with creating human-like text or speech output based on structured data or input from NLU systems.
- NLG systems generate coherent and contextually relevant text or speech by combining linguistic rules, templates, and sometimes machine learning models.
- NLG applications include text summarization, language translation, chatbot responses, and content generation for news articles or reports.

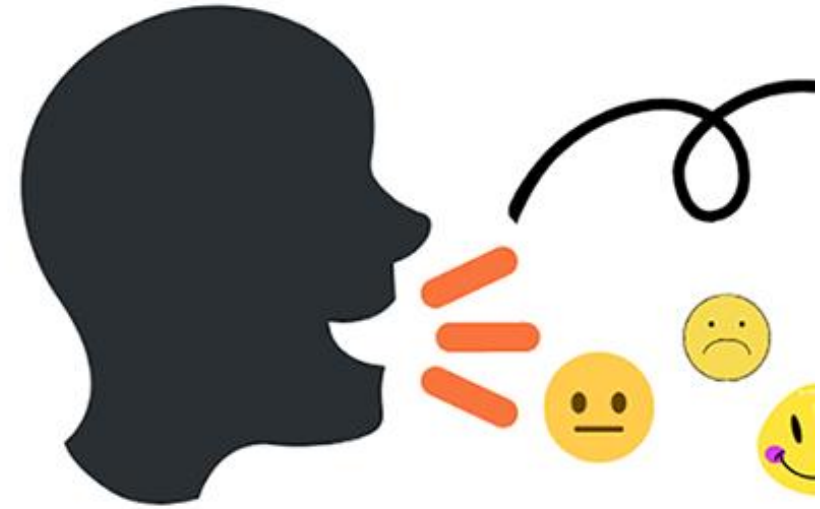
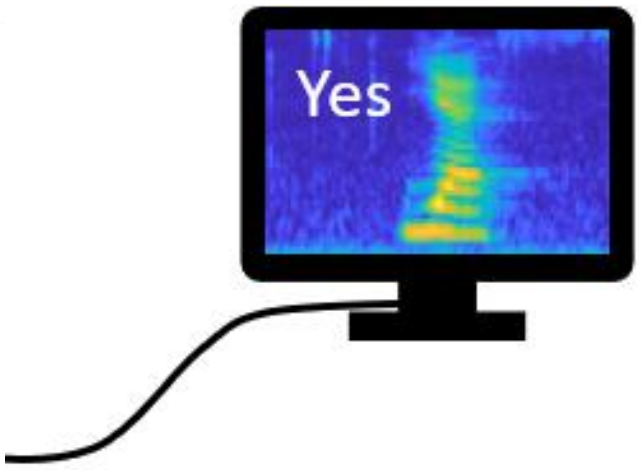
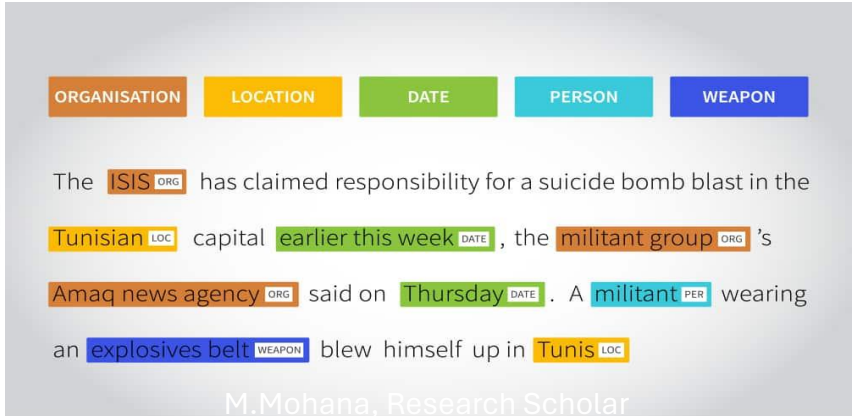


Image Source: Google





NLP Applications

- Translating the languages
- Text processing in various languages
- Automatic Text Summarization
- Analyzing sentiments
- Speech Recognition
- Named Entity Recognition
- Phrase Extraction
- Tense Identification
- Relationship Extraction and so on.

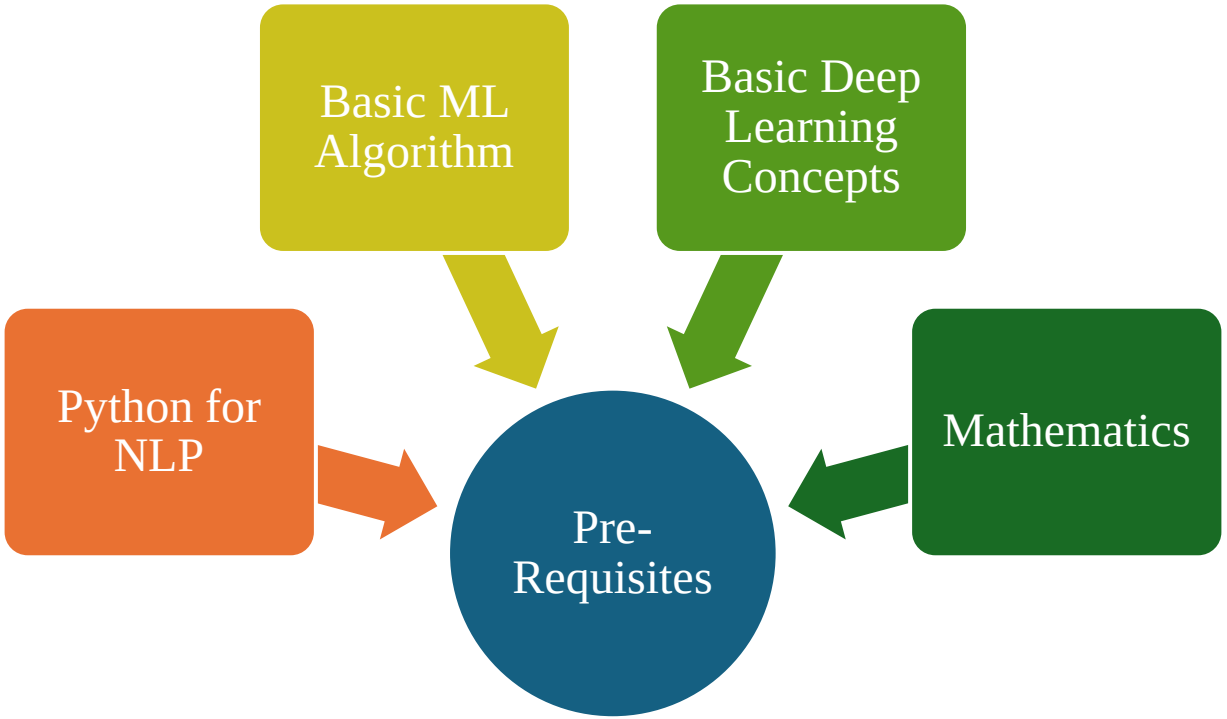
NLP Pipeline

Sentence Segmentation	• Breaks the paragraph into sentence
Word Tokenization	• Break the sentence into separate words or tokens
Stemming	• Normalize words into its base form or root words (Celebrate- celebrates, celebrated, and celebrating) (no meaning sometimes)
Lemmatization	• Similar to Stemming but root word has meaning, used to group different inflected forms of the word, called Lemma
Identifying Stop Words	• “is,and, the, a” – stop words might be filtered out before doing any statistical analysis
Dependency Parsing	• To find that how all the words in the sentence are related to each other
POS tags	• Parts of speech, indicates that how a word functions with its meaning as well as grammatically within the sentences.
Name Entity Recognition	• Process of detecting the named entity such as person name, movie name, organization name or location
Chunking	• Used to collect the individual piece of information and grouping them into bigger pieces of sentences

Phases of NLP

Lexical Analysis	• Scans the source code as a stream of characters and converts lexemes • Whole text into paragraphs, sentences, and words
Syntactic Analysis	• Used to check grammar, and word arrangements, and show the relationship among the words • I play (reject – no meaning), I play cricket (Accept – full meaning)
Semantic Analysis	• Concerned with the meaning representation • Focus on literal meaning of words, phrases, and sentences
Discourse Integration	• Depends upon the sentences that proceed with it and also invokes the meaning of the sentences that follow it
Pragmatic Analysis	• To discover the intended effects by applying a set of rules that characterize cooperative dialogues. • “open the door” – request instead of an order

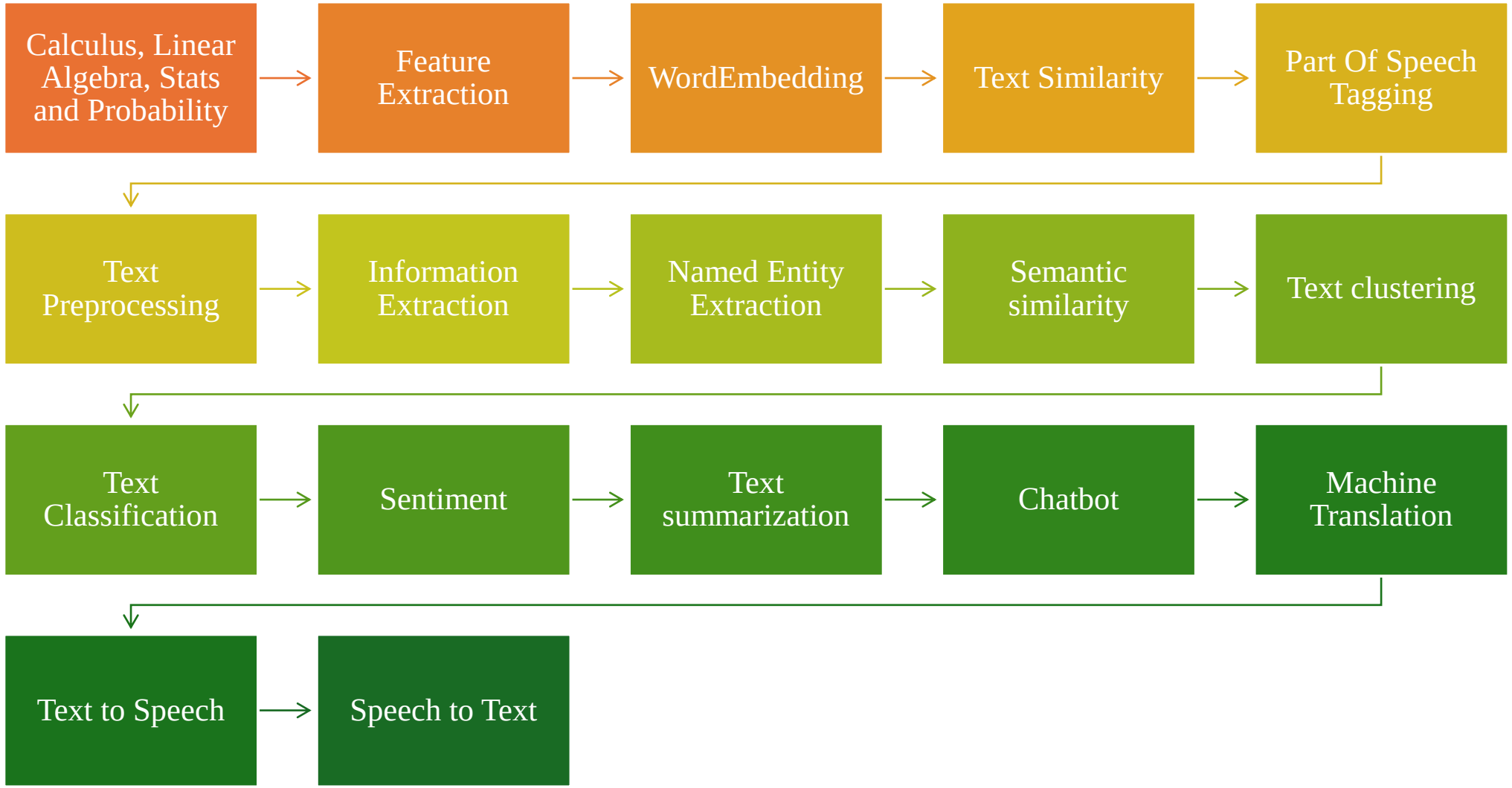
Things We need to know for learning NLP



- Text processing techniques
- Word Embedding
- Deep Learning Network for NLP (CNN, LSTM, GRUs, Encoder and Decoder)
- Attention mechanism, Transfer learning in NLP
- Transformers (BERT, GPT, ALBERT and so on)
- Fine Tuning NLP task
- Large Language Model (LLM)

PyTorch TensorFlow

Roadmap List for NLP

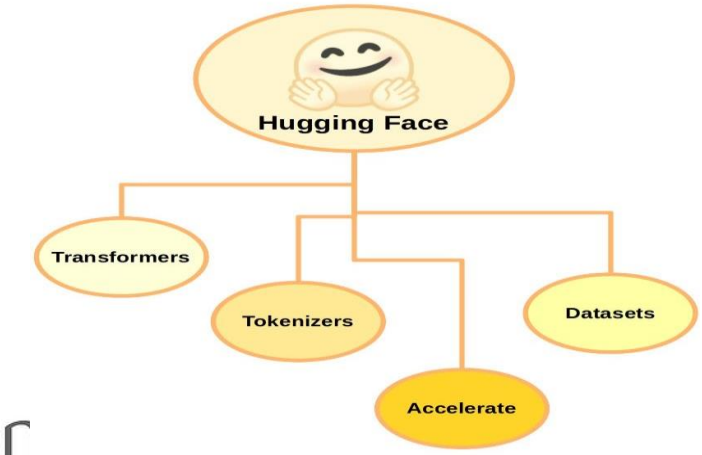


fastText

CoreNLP

spaCy

GENSIM
topic modelling for humans



AllenNLP

PyTorch NLP

python
Natural Language Analyses
with NLTK

Image Source: Google

Open sources and Library available for NLP

Important Reading to know about NLP approaches

Level	Topic	Weblink
Beginning Level	RNN and LSTM	https://arxiv.org/pdf/1808.03314.pdf
	Word2Vec	https://arxiv.org/pdf/1301.3781
	Attention for Translation Task	https://arxiv.org/abs/1409.0473
	Attention is all you Need	https://arxiv.org/abs/1706.03762
	BERT	https://arxiv.org/abs/1810.04805
	GPT-1	https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf
	GPT-2	https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
	Research Papers	https://www.kaggle.com/discussions/general/236973
Advanced Level	RLHF	https://arxiv.org/abs/2203.02155
	LIama2	https://arxiv.org/abs/2307.09288
	PaLM2	https://blog.google/technology/ai/google-palm-2-ai-large-language-model/
	Mistral : Mixture of Experts	https://mistral.ai/news/mixtral-of-experts/
	GPT -4	https://openai.com/research/gpt-4
	Gemini AI	https://blog.google/technology/ai/google-gemini-ai

https://youtu.be/7vHquWmUriE?si=JkiXw-b6i371hyQX

Projects we should try for a better understanding of NLP

Sentiment Analysis
Question Answering System
Named Entity Recognition
Fake News Detection
Topic Modeling
Text Similarity
Text summarization and machine translation
Next word Prediction
LLM applications using RAG
Fine-tune a model for a specific NLP task
Constructing your own LLM, inspired by models like Llame 2

Let’s explore more about it

Text Pre-processing Techniques

Tokenization
Dividing text into smaller units such as words, phrases, or symbols (tokens).

Lowercasing
Converting all text to lowercase.

Stopword Removal
Eliminating common words that do not carry significant meaning.

Punctuation Removal
Removing punctuation marks.

Numeric Token Removal
Eliminating numerical tokens.

Whitespace Removal
Eliminating extra spaces, tabs, or newline characters.

Stemming
Reducing words to their base or root form.

Lemmatization
Converting words to their canonical form based on their part of speech.

Input TextSpell Checking and Correction
Detecting and correcting spelling errors.

Text Normalization
Standardizing text by converting abbreviations or variations to their full forms.

Entity Recognition and Masking
Identifying and masking named entities.

Removing HTML Tags and Special Characters
Eliminating HTML tags and special characters.

Text Pre-processing Techniques Examples

Tokenization

Input Text: "The quick brown fox jumps over the lazy dog."

Output Tokens: ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."]

Lowercasing

Input Text: "The Quick Brown Fox"

Output Text: "the quick brown fox"

Stopword Removal

Input Text: "The quick brown fox jumps over the lazy dog."

Output Text: "quick brown fox jumps lazy dog."

Punctuation Removal

Input Text: "The quick, brown fox jumps over the lazy dog!"

Output Text: "The quick brown fox jumps over the lazy dog"

Numeric Token Removal

Input Text: "There are 5 apples on the table."

Output Text: "There are apples on the table."

Whitespace Removal

Input Text: "The \t quick brown fox"

Output Text: "The quick brown fox".

Stemming

Input Text: "running"

Output Text: "run"

Lemmatization

Input Text: "better"

Output Text: "good"

Input TextSpell Checking and Correction

Input Text: "I havvve a pen."

Output Text: "I have a pen."

Text Normalization

Input Text: "w/o"

Output Text: "without"

Entity Recognition and Masking

Input Text: "John Smith works at Google."

Output Text: "[PERSON] works at [ORGANIZATION]."

Removing HTML Tags and Special Characters

Input Text: " This is bold text. "

Output Text: "This is bold text."

Feature Extraction Techniques

Techniques	Main Feature	Use case
CountVectorizer	Coverts text to matrix of word counts	Text classification, topic modeling
TF-IDF (Term Frequency-Inverse Document Frequency)	Assign weights to words based on importance	Information retrieval, text classification
Word embeddings	Vector representation of words based on semantics and syntax	Text classification, Information Retrieval
Bag of words	Represents text as a vector of word frequencies	Text classification, Sentimental Analysis
Bag of n-grams	Capture frequency of sequences of n words	Text classification, Sentimental Analysis
Hashing Vectorizer	Maps words to fixed-size features space using hashing function	Large-scale text classification, online learning
Latent Dirichlet Allocation (LDA)	Identifies topics in the corpus and assigns probability distribution to each document	Topic modeling, Content analysis
Non-negative matrix factorization (NMF)	Decomposes documents-term matrix into lower-dimensional parts	Topic modeling, Content analysis

https://medium.com/@eskandar.sahel/exploring-feature-extraction-techniques-for-natural-language-processing-46052ee6514

Feature Extraction Techniques

Techniques	Main Feature	Use case
Principal Component Analysis (PCA) , t-SNE	Reduces dimensionality of documents-term matrix	Text visualization, text compression
Part-of-speech (POS) tagging	Assigns parts of speech tag to each word in text	Named entity recognition, text classification
N-grams	sequences of contiguous words or characters, capturing local word dependencies	Captures context and word order information, useful in language modeling, machine translation, and text generation
Named Entity Recognition (NER)	identifies and classifies named entities (e.g., person names, organizations, locations) in text	Information extraction, entity linking, and improving search engine results
Dependency Parsing	Captures syntactic dependencies, useful for machine translation, question answering, and information retrieval	analyzes the grammatical structure of a sentence by identifying relationships between words
Syntax Tree-Based Features	Includes subtree patterns, syntactic paths, and tree kernels for parsing and semantic analysis	Syntax tree-based features capture syntactic structures and relationships in sentences

https://medium.com/@eskandar.sahel/exploring-feature-extraction-techniques-for-natural-language-processing-46052ee6514

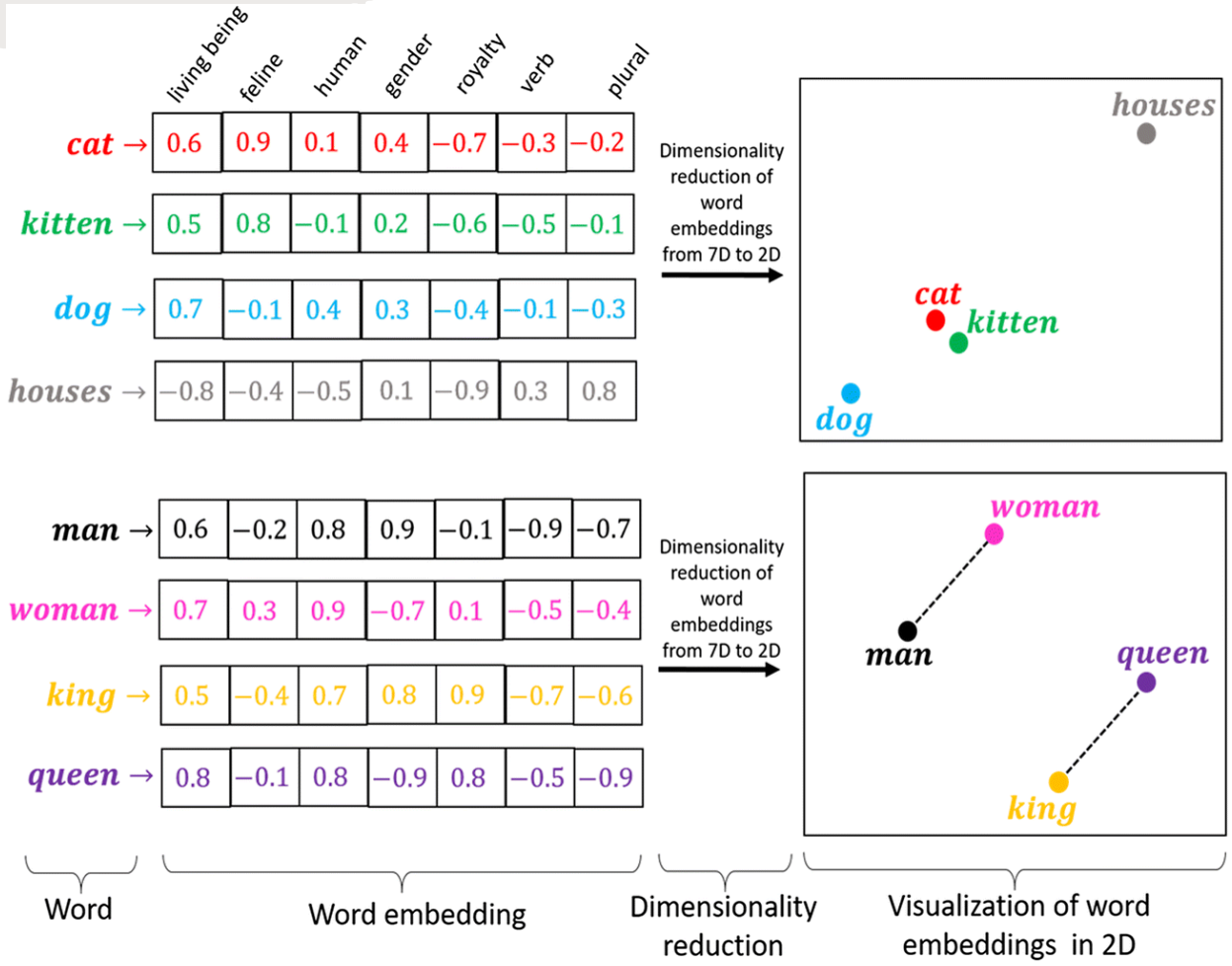
Word Embedding or Word Vector

- Represent words as dense vectors in a continuous vector space, where words with similar meanings are closer to each other in the space.
- Numeric representations of words in a lower-dimensional space
- Try to capture semantic and syntactic information
- Word2Vec, GloVe (Global Vectors for Word Representation), and fastText
- Method of extracting features out of text, to feed into ML model for work with text data

Need for Word Embedding

- To reduce dimensionality
- To use a word to predict the words around it
- Inter-word semantics must be captured

Image Source: Google



https://www.geeksforgeeks.org/word-embeddings-in-nlp/

Word Embedding or Word Vector (Cont.)

Approaches for Text Representation

- **Traditional Approach**
  - Compiling a list of distinct terms and giving each one a unique integer value, or id. After that, insert each word's distinct id into the sentence.
  - Every vocabulary word is handled as a feature in this instance.
  - Large vocabulary will result in an extremely large feature size.
  - One-Hot Encoding, Bag of Words (BoW), CountVectorizer, TF-IDF
- **Neural Approach**
  - Word2Vec, Continuous Bag of Words (CBOW), Skip-Gram
- **Pretrained Techniques**
  - Representation of words that are learned from large corpora and are made available for reuse in various NLP tasks
  - Capture semantic relationships between words, to understand similarities and relationships between different words in a meaningful way
  - GloVe (Global Vectors for Word Representation), FastText, BERT (Bidirectional Encoder Representations from Transformers)

- Types of Word Embedding
  - Frequency-based Embedding
    - Count Vector
    - TF-IDF Vector
    - Co-Occurrence Vector
  - Prediction-based Embedding
    - CBOW
    - Skip – Gram model

$\mathrm{tf}(t,d)$

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

X

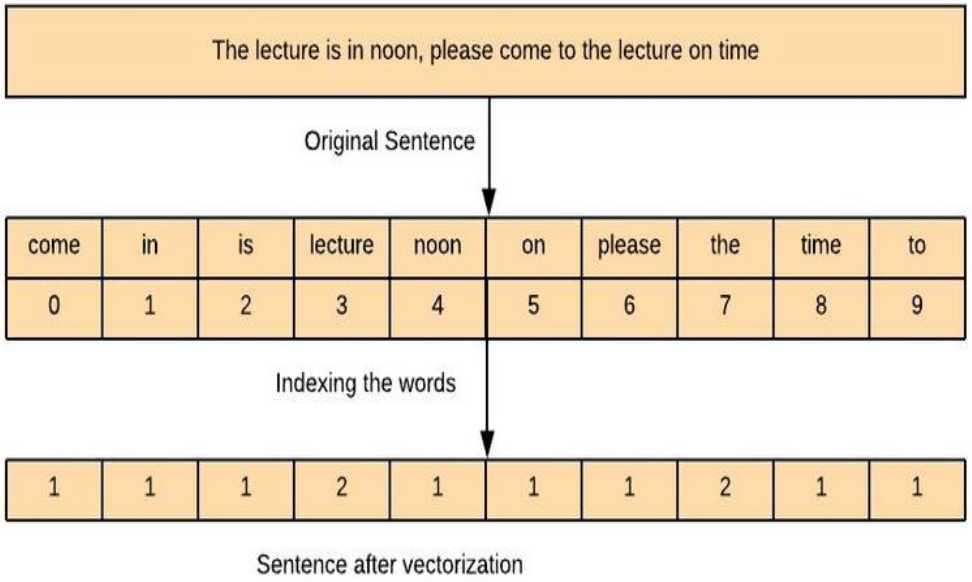
$\mathrm{idf}(t,D)$

	blue	bright	can	see	shining	sky	sun	today
	0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

- TF-IDF: Multiply TF and IDF scores, use to rank importance of words within documents
  - Most important word for each document is highlighted

$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$

	blue	bright	can	see	shining	sky	sun	today
1	**0.301**	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	**0.201**
3	0	0.0417	0	0	0	**0.100**	0.0417	0
4	0	0.0209	**0.100**	**0.100**	**0.100**	0	0.0417	0



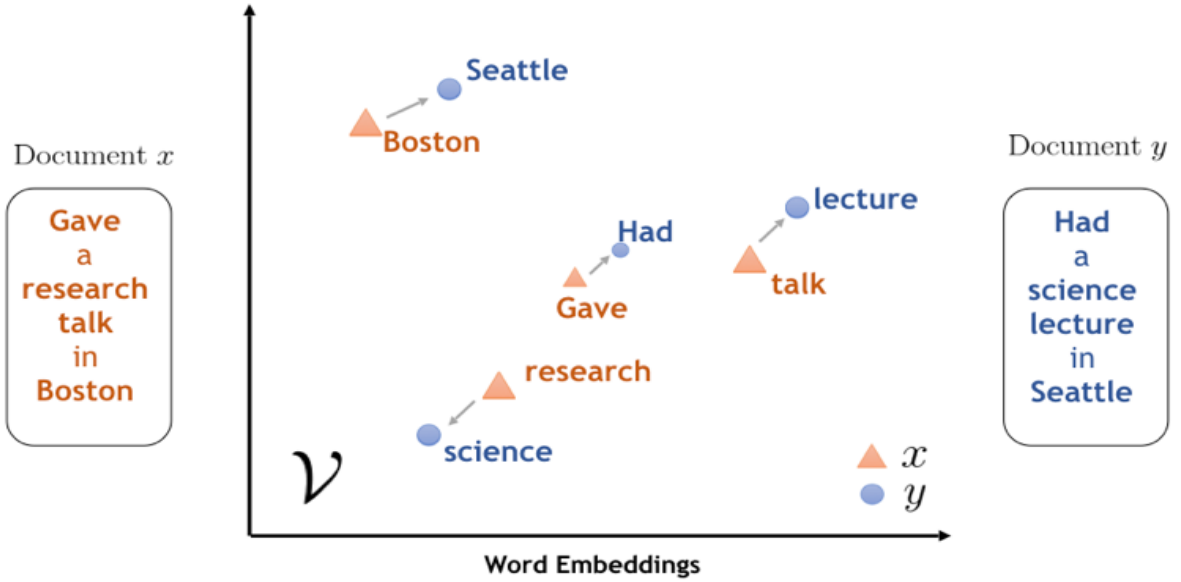


Image Source: Google

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Text Similarity

- Refers to the measure of how similar two or more pieces of text are in terms of their semantic or syntactic content.
- Finding similarities between documents is utilized in a variety of fields, book and article recommendations, detection of plagiarism, legal documents, etc.
- Two texts define the same notion and are semantically comparable, or identical when say they are similar.

1. Global Warming is here.
Ocean temperature is rising.

2. I'm reading a book.
The book is about NLP.

3. Text similarity in NLP is easy.
I like data science.

4. This place is great.
This is great news.

As a human, this is similar

1. Global Warming is here.
Ocean temperature is rising.

2. I'm reading a book.
The book is about NLP.

3. Text similarity in NLP is easy.
I like data science.

4. This place is great.
This is great news.

In NLP, this is similar, How ?

Text Similarity (Cont.)

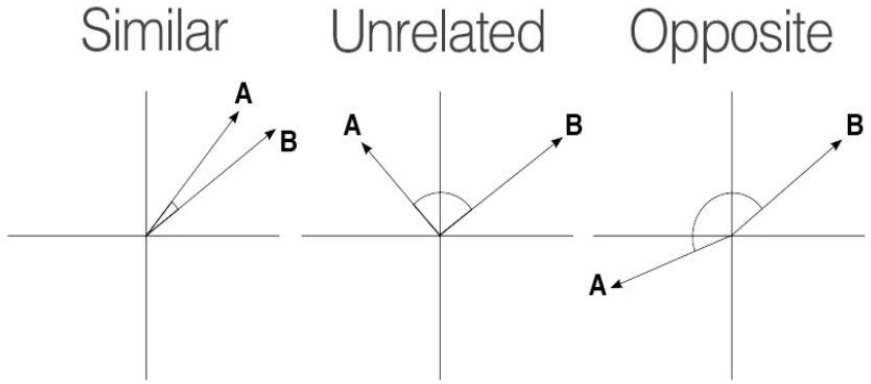
- Semantic similarity is about the meaning closeness, and lexical similarity is about the closeness of the word set.

Let's check the following two phrases as an example:

The dog bites the man

The man bites the dog

- According to the lexical similarity, those two phrases are very close and almost identical because they have the same word set.
- For semantic similarity, they are completely different because they have different meanings despite the similarity of the word set.
- Calculating text similarity depends on converting text to a vector of features, and then the algorithm selects a proper feature representation, like TF-IDF.



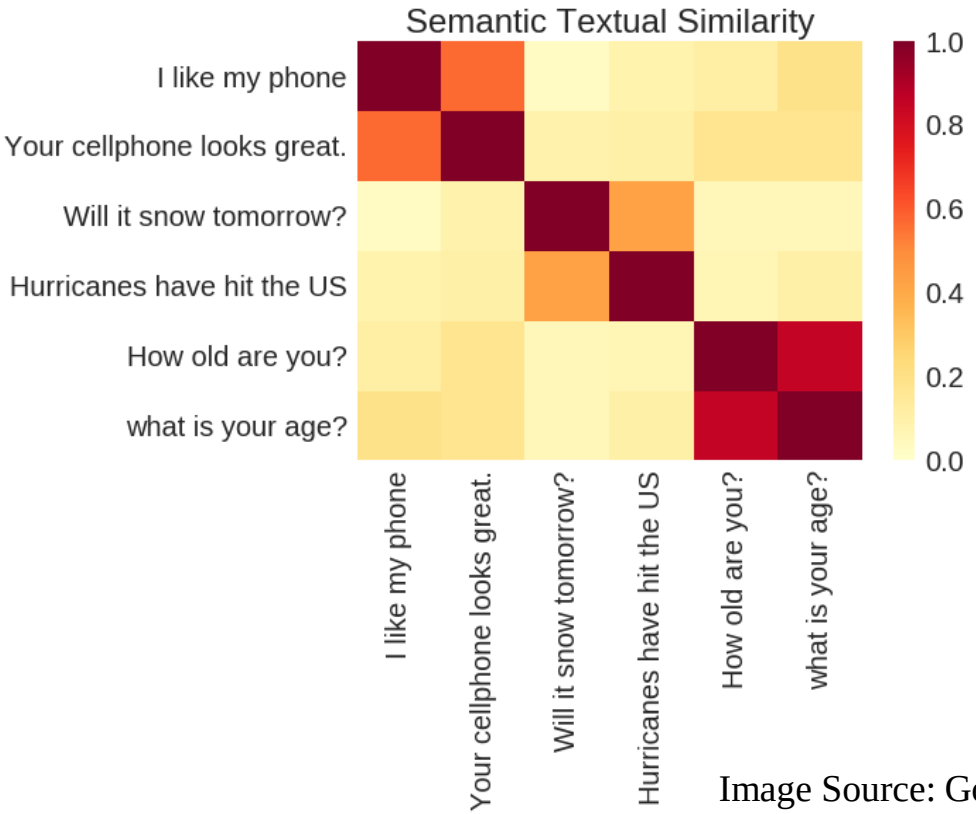


Image Source: Google

https://www.baeldung.com/cs/semantic-similarity-of-two-phrases#:~:text=2.-,Text%20Similarity,be%20lexical%20or%20in%20meanin

Techniques are commonly used to compute text similarity

Cosine Similarity

- Measures the cosine of the angle between two vectors representing the text.
- Used with word embeddings or TF-IDF vectors to compute similarity.

Jaccard Similarity

- Measures the similarity between two sets by dividing the size of their intersection by the size of their union.

Edit Distance

- Calculates the minimum number of operations (insertions, deletions, substitutions) required to transform one text into another.
- Measuring similarity between texts with similar structures but potentially different words.

Word Embeddings

- Used to compute similarity between texts by averaging or combining word vectors to represent the entire text.

BERT and Similar Models

- Pre-trained language models like BERT (Bidirectional Encoder Representations from Transformers) can be fine-tuned for text similarity tasks.

Semantic Similarity Metrics

- Metrics use semantic information to measure text similarity.
- They may leverage knowledge graphs, ontology-based approaches, or semantic vector spaces to compute similarity.

Sequence Alignment Algorithms

- Techniques like **Needleman-Wunsch or Smith-Waterman algorithms**, commonly used in bioinformatics for sequence alignment,
- Can also be adapted to measure text similarity by aligning sequences of words or characters.

Part-of-Speech Tagging

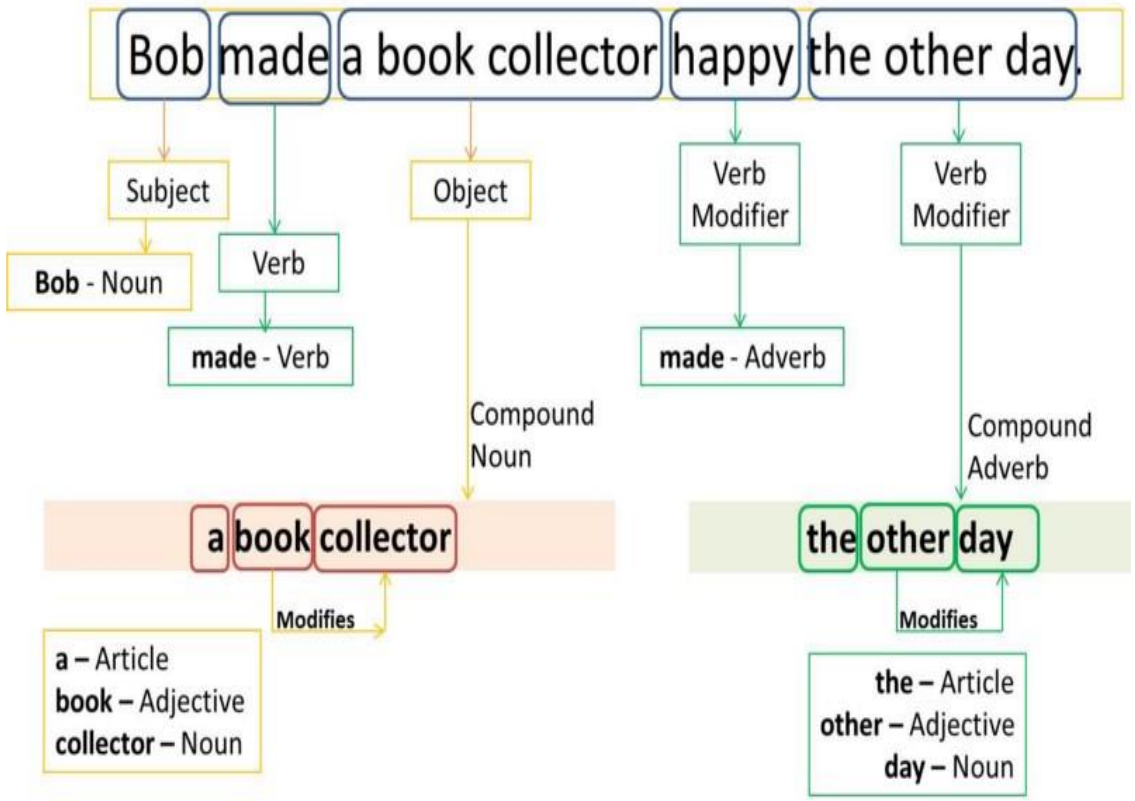


Image Source: Google

- Label each word in a sentence with its corresponding part of speech, such as noun (NN), verb (VB), adjective (JJ), adverb (RB), pronoun (PRP), preposition (IN), conjunction (CC), interjection (UH), Determiner (DT), etc.,
- Machine translation, sentiment analysis, information retrieval, and NER, POS tagging are essential.
- Works well for clearing out ambiguity in terms with numerous meanings and revealing a sentence's grammatical structure. For example, "lead" can be a noun (the metal) or a verb (to guide).
- There is a hierarchy of tasks in NLP, at the bottom are sentence and word segmentation.
- POS tagging builds on top of that, and phrase chunking builds on top of POS tags.
- The, a, and an are not always considered POS but are often included in POS tagging libraries.

Several approaches to POS tagging

Rule-Based Tagging

- This approach uses handcrafted rules based on linguistic knowledge to assign POS tags to words.
- For example, a rule might specify that words ending in "-ing" are typically gerunds (verbs).

Probabilistic Tagging

- This approach uses statistical models to assign POS tags based on the probability of a word occurring with a particular tag.
- Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) are commonly used probabilistic models for POS tagging.

Deep Learning Tagging

- With the rise of deep learning, neural network-based approaches have become popular for POS tagging.
- Models like Bidirectional LSTMs (Long Short-Term Memory networks) or Transformers can learn complex patterns and dependencies in text to predict POS tags.

Hybrid Approaches

- Some systems combine rule-based and probabilistic or deep-learning techniques to improve tagging accuracy.

Challenges in Part-of-Speech Tagging

Ambiguity

For example, the word "bank" can be a noun (e.g., "riverbank") or a verb (e.g., "to bank on something").

Out-of-Vocabulary (OOV) Words

not present in their training data, leading to difficulties in assigning appropriate tags to these out-of-vocabulary words.

Domain-Specific Terminology

trained on general text may struggle with domain-specific terminology and jargon.

Word Sense Disambiguation

correct meaning of a word with multiple meanings in a particular context. For example, "bat" as a flying mammal and "bat" as a sports equipment requires contextual understanding.

Syntax and Grammar Complexity

complex syntax or grammar structures, such as passive voice, nested clauses, or ellipses, can pose challenges for POS taggers.

Lack of Context

POS tagging models may struggle with lack of context, especially in short or incomplete sentences.

Cross-Lingual Challenges

POS tagging across multiple languages introduces additional challenges due to differences in grammar, word order, and linguistic features.

Information Extraction

- Process of automatically extracting structured information from unstructured or semi-structured text data that can be easily analysed, searched, and visualized
- Goal of information extraction is to identify specific pieces of information, such as entities (e.g., names of people, organizations, locations) and their relationships (e.g., who works for which company), from textual data.
- Spark NLP – used for identifying specific entities from large volumes of text data, converting them into a structured format for further analysis
- Involves identifying specific entities, relationships, and events of interest in text data, such as named entities like people, organizations, dates, and locations
- Main applications like Search engines, chatbots, recommendation systems, and fraud detection, among others
- 'TextMatcher', and 'BigTextMatcher' are annotators that used to match and extract text patterns from a document
- 'BigTextMatcher'- is designed for large corpora and ' TextMatcher'- works by defining a set of rules that specify the patterns to match and how to match them

Several techniques and approaches are used in information extraction

Named Entity Recognition (NER)

- Involves identifying and classifying entities mentioned in text into predefined categories such as person names, organization names, locations, dates, and more, using machine learning models such as Conditional Random Fields (CRFs), SVMs, or deep learning models like Bidirectional LSTMs or Transformers.

Relation Extraction

- For example, in the sentence "John works at XYZ Corp," the relation extraction task would identify that "John" is an employee of "XYZ Corp." Relation extraction can be done using pattern-based approaches, rule-based systems, or machine learning models.

Event Extraction

- Event extraction focuses on identifying events mentioned in text and extracting relevant information such as event types, participants, time, location, and other attributes.

Template-Based Extraction

- For example, a template for extracting product information might include patterns like "Product: [product name], Price: [price], Category: [category].“
- Template-based extraction is useful for extracting structured data from semi-structured or unstructured text.

Machine Learning-Based Extraction

- Machine learning models, including supervised, semi-supervised, and unsupervised learning techniques, are widely used for information extraction tasks.
- These models learn patterns and relationships from annotated training data and apply them to new text for extraction.

Albert Einstein PER Albert Einstein was born in Ulm LOC in Germany LOC on March 14, 1879. Six weeks later the family moved to Munich LOC , where he later on began his schooling at the Luitpold Gymnasium ORG . In 1896 he entered the Swiss Federal Polytechnic School ORG in Zurich LOC to be trained as a teacher in physics and mathematics.

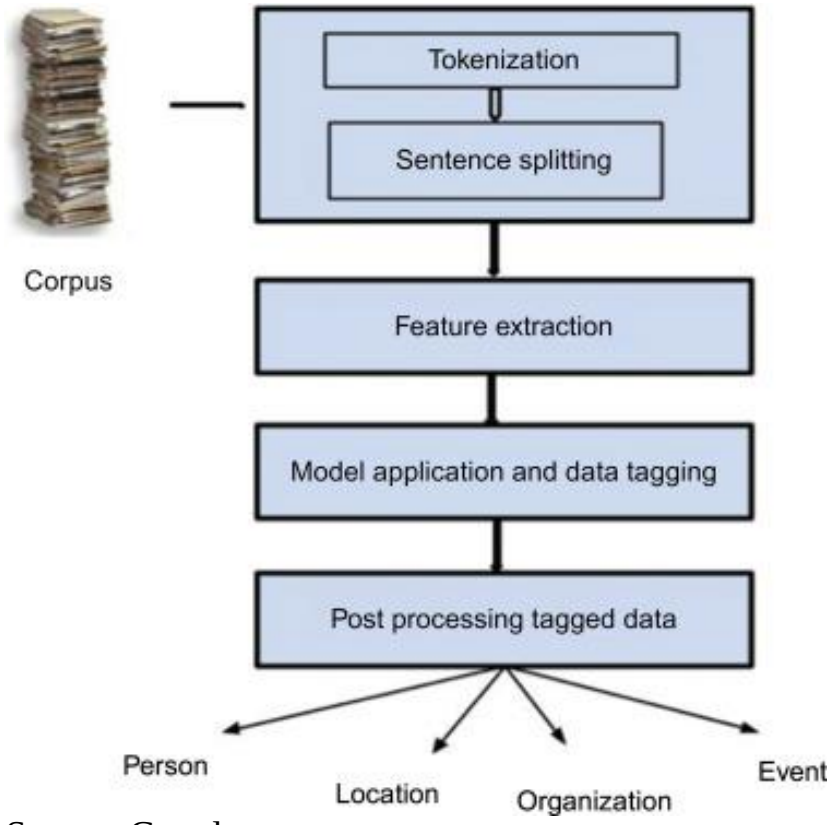


Image Source: Google

Named Entity Extraction

- Involves identifying and classifying named entities in text into predefined categories such as names of persons, organizations, locations, dates, numerical expressions, and other entities of interest.
- Serves as a bridge between unstructured text and structured data,
- enabling machines to sift through vast amounts of textual information and extract nuggets of valuable data in categorial form

Fall into three broad categories:

- Rule-based approaches – a set of rules for the grammar of a language
- Machine learning approaches - machine learning model on a labeled dataset using algorithms like conditional random fields and maximum entropy
- Hybrid approaches – a rule-based system to quickly identify easy-to-recognize entities and a machine learning system to identify more complex entities

Key steps and techniques involved in Named Entity Extraction

Tokenization and Part-of-Speech (POS) Tagging:

1. Input text is tokenized into words or tokens, and each token is assigned a part-of-speech tag (e.g., noun, verb, adjective) using POS tagging techniques.

Named Entity Recognition:

1. NER algorithms then identify and label tokens that correspond to named entities in the text. Commonly used techniques for NER include:
    1. Rule-based approaches: Using handcrafted rules and patterns to match and classify named entities based on linguistic features (e.g., capitalization, word position, context).
    2. Machine learning models: Training supervised learning models such as Conditional Random Fields (CRFs), Support Vector Machines (SVMs), or deep learning models like Bidirectional LSTMs (Bi-LSTMs) or Transformers (e.g., BERT, RoBERTa) on annotated NER datasets to predict named entity labels for tokens.

Post-processing and Entity Classification:

1. After identifying named entities, post-processing steps may involve resolving entity boundaries, handling overlapping entities, and classifying entities into specific categories (e.g., person, organization, location).

Evaluation and Validation:

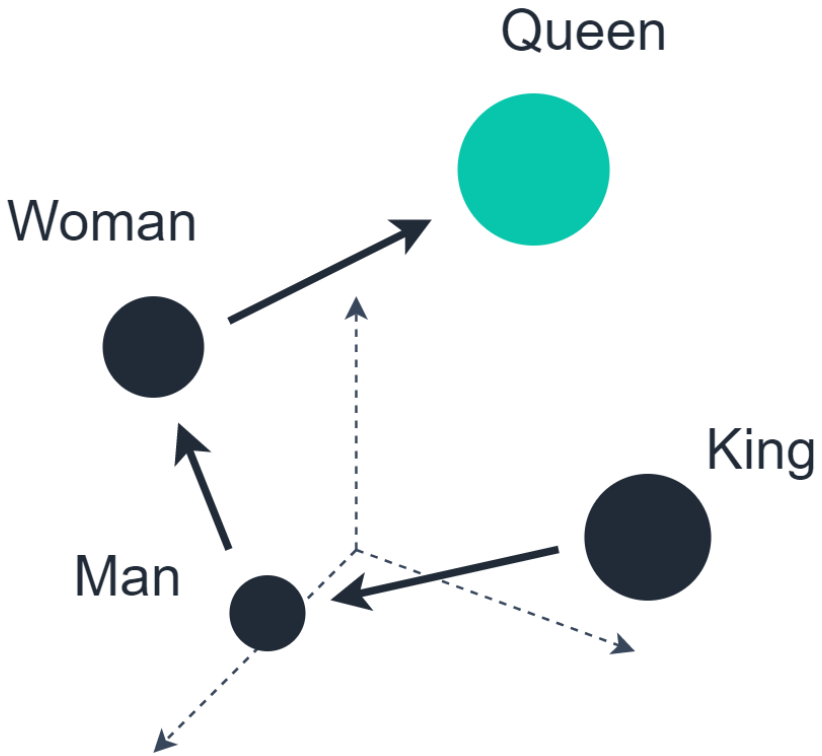
1. NER systems are evaluated using metrics such as precision, recall, and F1-score, comparing the model's predictions against manually annotated ground truth data.

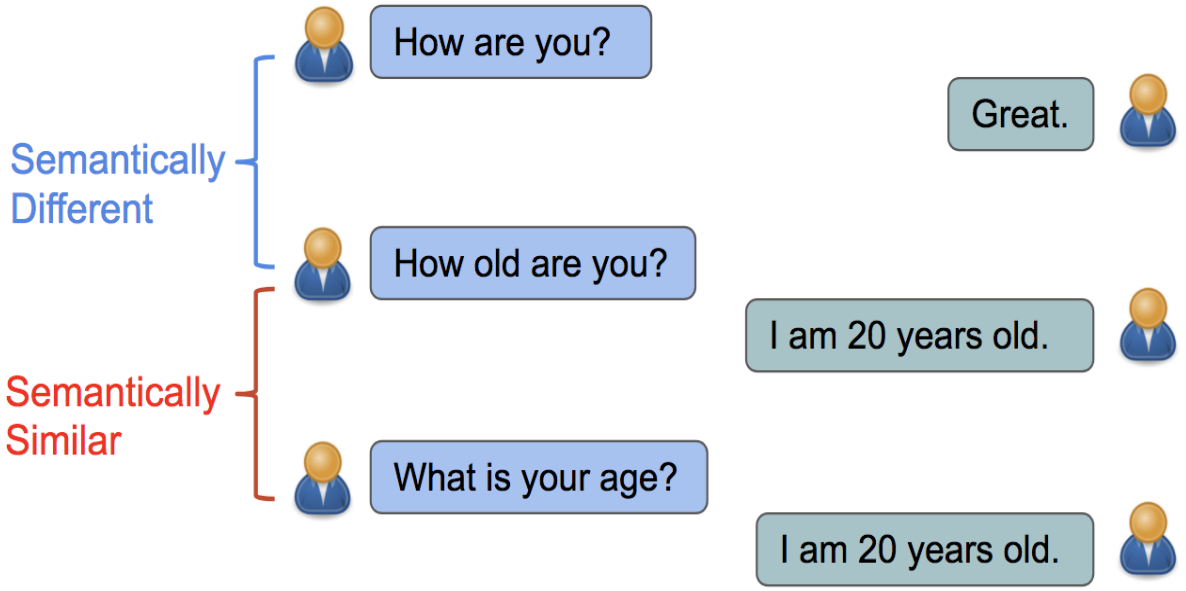
Named Entity Linking (NEL):

1. In some cases, NER is followed by Named Entity Linking, where identified named entities are linked to corresponding entries in knowledge bases or ontologies to enrich their semantic representation.

Semantic similarity

- Refers to the measure of how similar two pieces of text are in terms of their meaning or semantics.
- Measures how close or how different the two pieces of word or text are in terms of their meaning and context.
- NLP applications such as information retrieval, question answering, text summarization, and recommendation systems.





What is Semantic Similarity?

- Semantic Similarity refers to the degree of similarity between the words.
- Focus is on the structure and lexical resemblance of words and phrases.
- Semantic similarity delves into the understanding and meaning of the content.

There are certain approaches for measuring semantic similarity in natural language processing

- Word Embedding - (skip-gram, cbow), GloVe, and Fasttext
- Word2Vec – (Continuous Bag of Words, Skip-gram)
- Dov2Vec – An extension of word2vec
- SBERT – Transformer-based model in which the encoder part captures the meaning of words in a sentence.
- InferSent -It uses bi-directional LSTM to encode sentences and infer semantics.
- USE (universal sentence encoder) – It's a model trained by Google that generates fixed-size embeddings for sentences that can be used for any NLP task.

https://www.geeksforgeeks.org/different-techniques-for-sentence-semantic-similarity-in-nlp/

Types of Semantic Similarity

Knowledge-Based Similarity	• To determine the semantic similarity between concepts • Represents each concept by a node in an ontology graph, also called the topological method because the graph is used as a representation for the corpus concepts.
Statistical-Based Similarity	• Calculates the semantic similarity based on learning features' vectors from the corpus. • count or TF-IDF in LSA, weights of Wikipedia concepts in ESA, synonyms in PMI, and co-occurring words of a set of predefined words in HAL.
String-Based Similarity	• Manhattan Distance, Euclidean Distance, Cosine Similarity, Jaccard Index, and Sorensen-Dice Index.
Language Model-Based Similarity	• Similarity measurement between two English phrases, with the assumption that they are syntactically correct.

Types of Semantic Similarity (Cont.)

M.Mohana, Research Scholar

Word-Level Semantic Similarity:

- **Word Embeddings:**
  Utilizes vector representations of words in a continuous vector space to measure similarity based on the cosine similarity or other distance metrics.
- **WordNet-based Measures:**
  Exploits WordNet, a lexical database of English, to compute similarity between words based on their synsets (groups of synonymous words) and hypernym/hyponym relationships.
- **Distributional Similarity:**
  Measures similarity between words based on their distributional properties in a corpus, capturing how often words co-occur with each other.
- **Lexical Similarity Measures:**
  Include techniques like Jaccard similarity, cosine similarity, or TF-IDF similarity, which compute similarity based on word overlap or statistical properties of words.

Sentence-Level Semantic Similarity:

- **Vector Space Models:**
  Extend word embeddings to sentences by aggregating or combining word vectors to represent the entire sentence. Similarity between sentences is then computed using techniques like cosine similarity.
- **Siamese Networks:**
  Deep learning architectures designed to learn sentence embeddings by comparing pairs of sentences and learning a similarity metric.
- **BERT-based Models:**
  Utilize pre-trained language models like BERT to compute sentence embeddings and measure similarity using techniques such as fine-tuning for semantic similarity tasks.

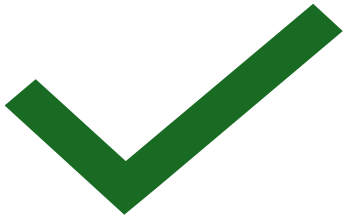
Types of Semantic Similarity (Cont.)

Document-Level Semantic Similarity

Topic Modeling: Techniques like Latent Semantic Analysis (LSA) or Latent Dirichlet Allocation (LDA) can be used to model topics in documents and compute similarity based on topic distributions.

Doc2Vec: An extension of Word2Vec that learns document embeddings, enabling similarity computation at the document level.

Graph-Based Models: Represent documents as nodes in a graph and compute similarity based on graph-based algorithms like Personalized PageRank or graph neural networks.



Semantic Textual Similarity (STS)

STS Benchmarks: Datasets like the STS Benchmark provide pairs of text with human-annotated similarity scores, allowing the evaluation of semantic similarity models.

Embedding-Based Approaches: Utilize pre-trained word or sentence embeddings to compute semantic similarity between texts, often fine-tuned on STS datasets for improved performance.

Common techniques and methods used for measuring semantic similarity

Word Embeddings

- **Word2Vec, GloVe, and FastText** represent words as dense vectors in a continuous vector space
- Similarity measured using **cosine similarity, Euclidean distance**, or other distance metrics between their corresponding word vectors

Sentence Embeddings

- **USE, BERT, and Sentence-BERT (SBERT)** generate embeddings for entire sentences or phrases.
- Similarity is computed using the **same distance metrics applied to sentence embeddings**

WordNet and ConceptNet

- WordNet - **Lexical database** that organizes words into semantic hierarchies and provides information about their relationships (e.g., synonyms, hypernyms, hyponyms).
- **ConceptNet is a knowledge graph** that captures commonsense knowledge and semantic relationships between concepts

Distributional Semantics

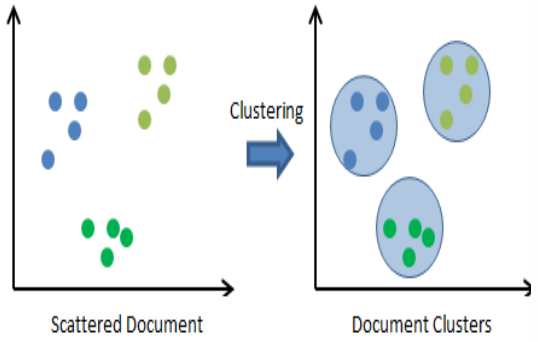
- Models capture semantic relationships between words based on their **distributional patterns in a large corpus of text**
- **Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA)** extract semantic information from word co-occurrence statistics

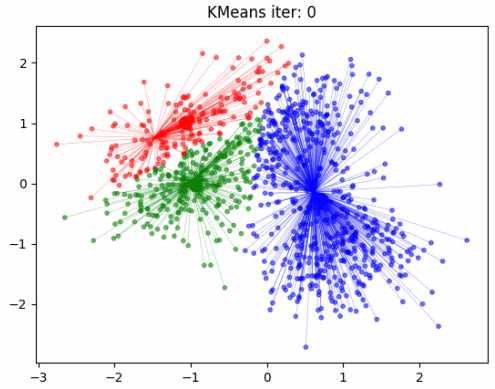
Deep Learning Models

- **Siamese networks and Triplet networks**
- Transformer-based models **like BERT, RoBERTa, and XLNet** leverage pre-trained language representations to measure semantic similarity

Metric Learning

- Similarity metric directly from data by optimizing a loss function that encourages similar pairs to have low distances and dissimilar pairs to have high distances.
- Pearson correlation, **Spearman correlation, or Mean Squared Error (MSE)** can be used to assess the performance of similarity models





Text Clustering

- Process of grouping similar documents or text data into clusters based on their semantic similarity or other relevant features
- By clustering text, identify trends, discover hidden patterns, extract valuable insights, and streamline large volumes of unstructured text data
- Unsupervised learning technique used for various tasks such as document categorization, topic modeling, and information retrieval

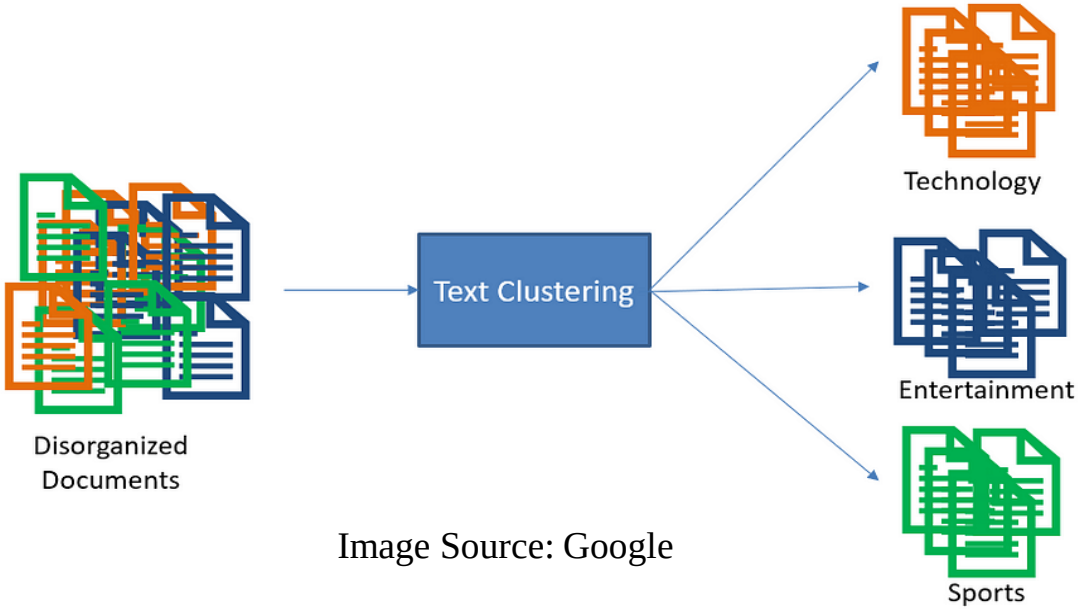


Image Source: Google

Algorithm Selection (K-means, hierarchical clustering, DBSCAN, and agglomerative)

Feature Extraction (TfidfVectorizer, TF-IDF, BoW, Word2Vec or GloVe, LDA)

Similarity Measures (Cosine similarity, Jaccard similarity or Euclidean distance)

Pre-processing (Basic text pre-process)

Evaluation (t-SNE, or PCA)

Text Classification

- Ask for automatically categorizing text documents into predefined classes or categories based on their content.

There are three text classification approaches:

- Rule-based System: texts are separated into an organized group using a set of handicraft linguistic rules.
- For example, words like Donald Trump and Boris Johnson would be categorized into politics. People like LeBron James and Ronaldo would be categorized into sports.
- Machine System: learns to make a classification based on past observations from the data sets. User data is prelabeled as train and test data. (Bag of words)
- Hybrid System: combines a rule-based and machine-based approach, approach usage of the rule-based system to create a tag and use machine learning to train the system and create a rule.

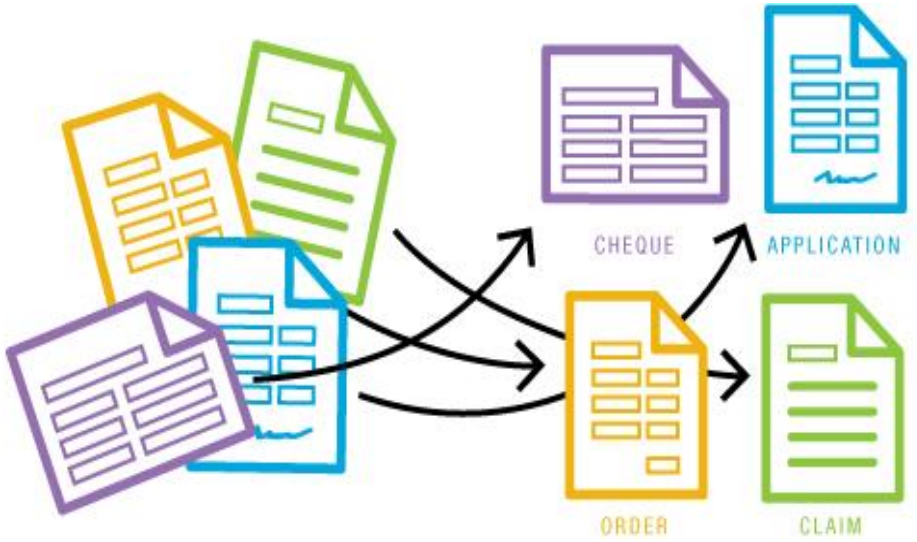
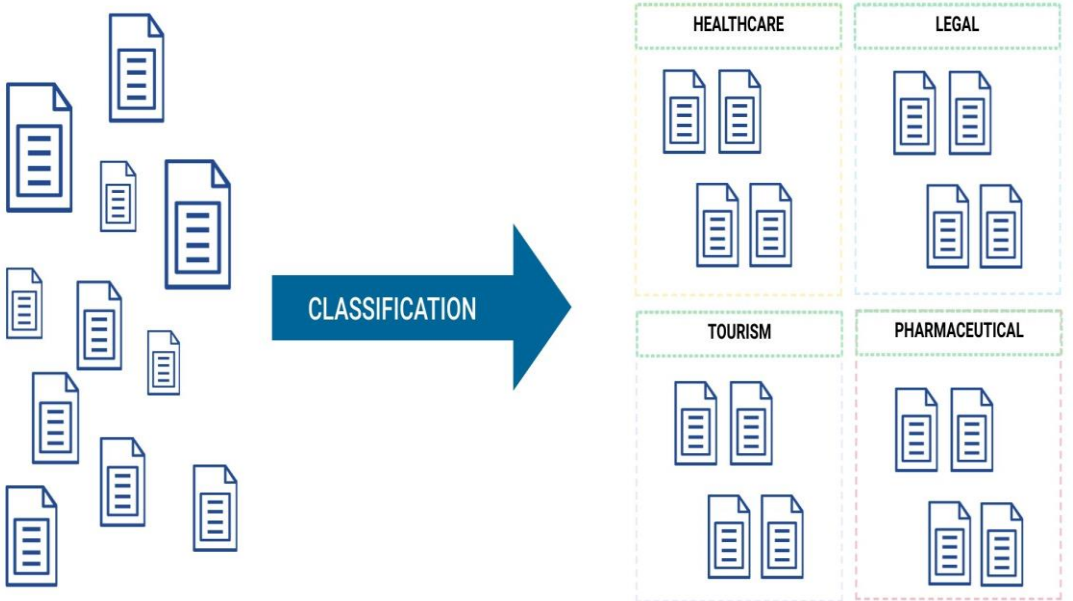


Image Source: Google



Application of Text Classification

Sentiment Analysis

Text sentiment, categorizing content as positive, negative, or neutral.

Spam Detection

Identify and filter out spam emails or messages by analyzing their content and characteristics, enhancing communication security.

Topic Labeling

Automatically assigning topics or categories to documents, making content organization and retrieval more efficient.

Language Identification

Detect the language in which a text is written, which is useful for multilingual content processing and translation.

News Categorization

News articles to be categorized into sections like politics, technology, sports, etc., improving content organization for readers.

Product Classification

E-commerce platforms use NLP for product categorization, ensuring items are correctly labeled and presented to customers.

Customer Feedback Analysis

Customer reviews and feedback to extract insights, understand customer satisfaction and areas of improvement.

Medical Document Classification

Categorizing medical records, research papers, and patient notes, assisting in efficient data retrieval for healthcare professionals.

Legal Document Categorization

Law firms use NLP to classify legal documents, making managing and retrieving information from large databases easier.

Social Media Monitoring

NLP tracks and classifies social media posts, tweets, and comments, allowing brands to monitor their online presence and engage with users.

Fraud Detection

Classify financial texts to detect fraudulent activities and identify potential risks.

Resume Screening

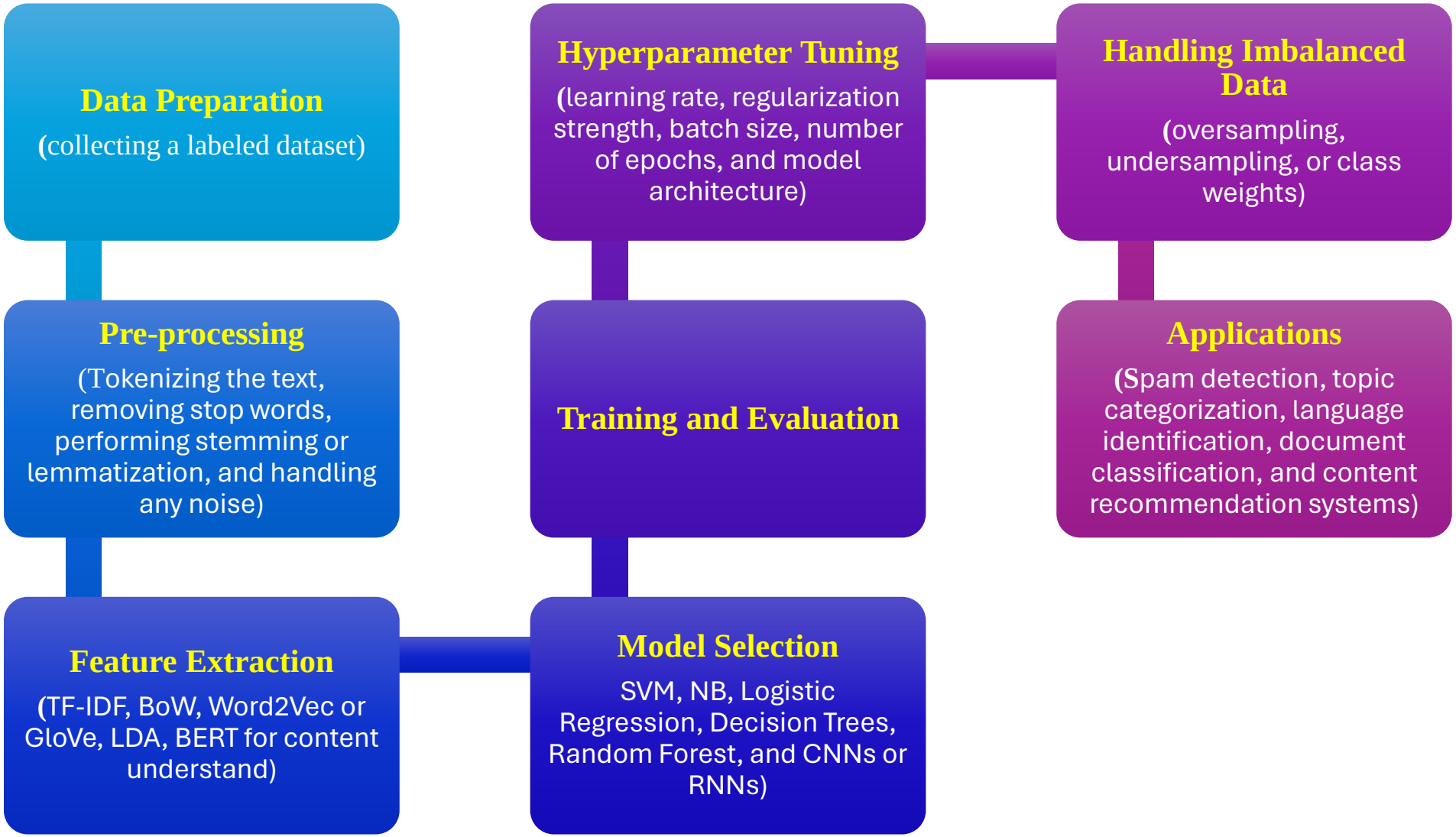
Resume screening by categorizing job applications based on skills, experience, and qualifications.

Content Recommendation

Recommending relevant articles, blogs, or products to users based on their interests.

https://www.analyticsvidhya.com/blog/2020/12/understanding-text-classification-in-nlp-with-movie-review-example-example/

Steps Involved in Text Classification



Sentiment Analysis

- Process of determining the sentiment or emotional tone expressed in a piece of text
- analyzing the text to identify whether the sentiment is positive, negative, or neutral
- Sentiment analysis, also known as opinion mining, is an important business intelligence tool
- applications to understand public opinion, customer feedback, social media trends, and more

Types of Sentiment Analysis

- Binary Sentiment Analysis: Positive and negative sentiments.
- Multi-class Sentiment Analysis: Positive, negative, neutral, and sometimes additional categories like very positive or very negative.
- Aspect-Based Sentiment Analysis: Analyzes specific aspects or aspects of a product, service, or topic to determine sentiment.

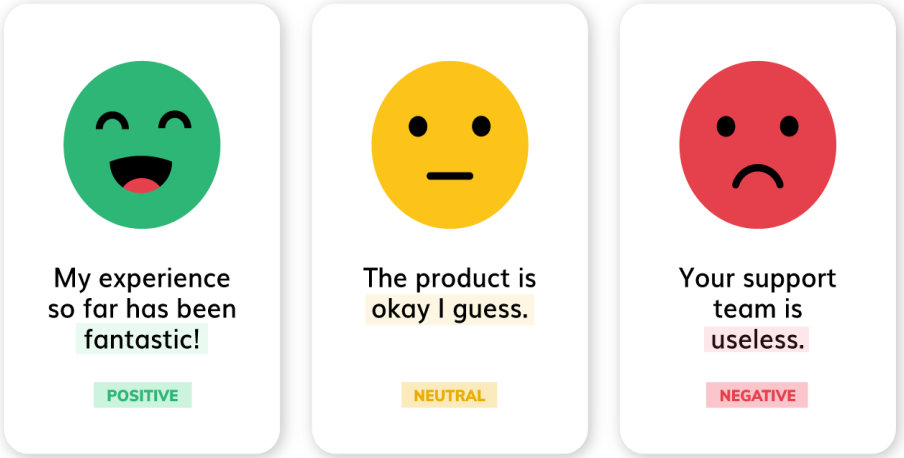
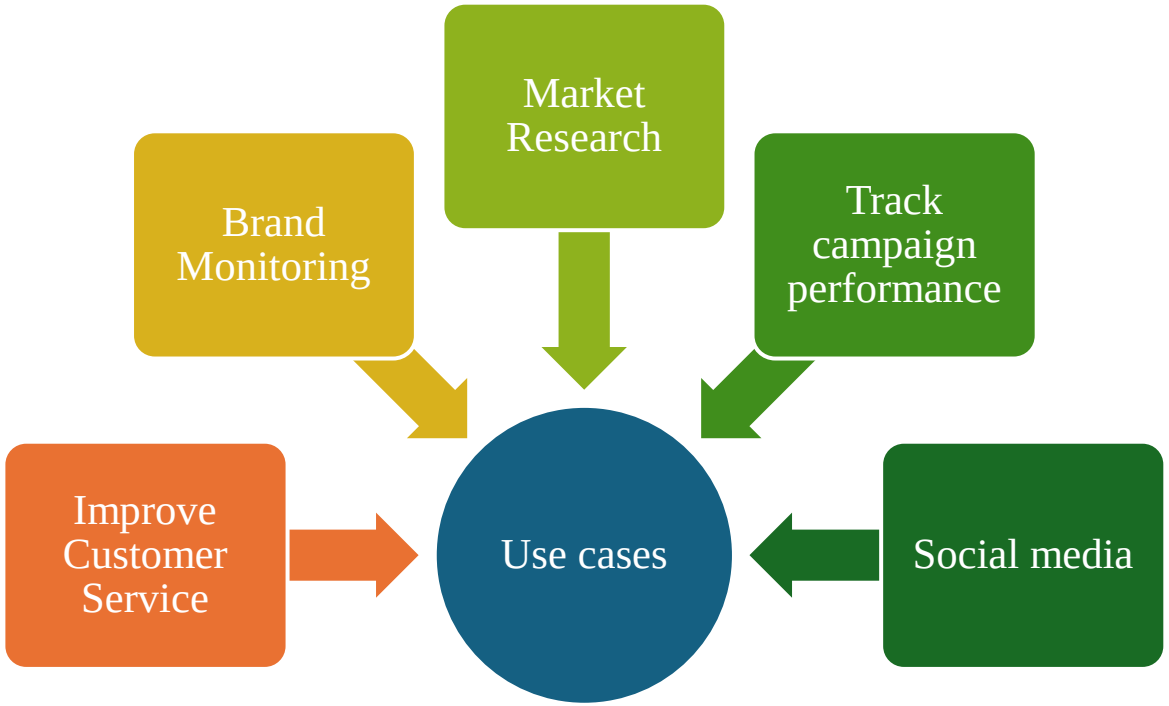


Image Source: Google

Why Sentiment Analysis is important?

I'm amazed by the speed of the processor but disappointed that it heats up quickly.

- Marketers might dismiss the discouraging part of the review and be positively biased towards the processor's performance.
- However, accurate sentiment analysis tools sort and classify text to pick up emotions objectively.

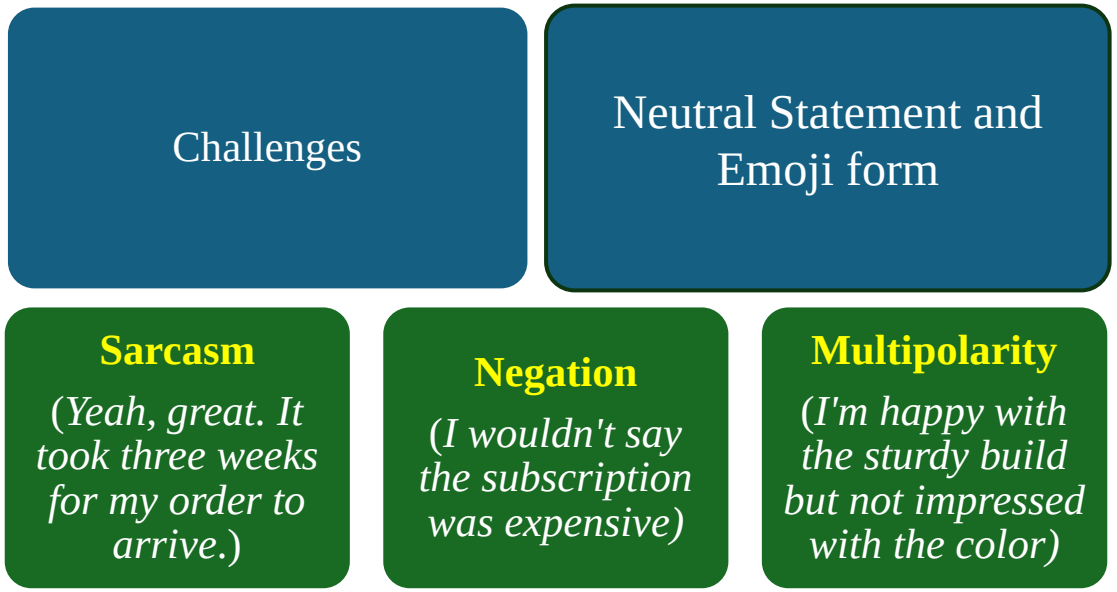


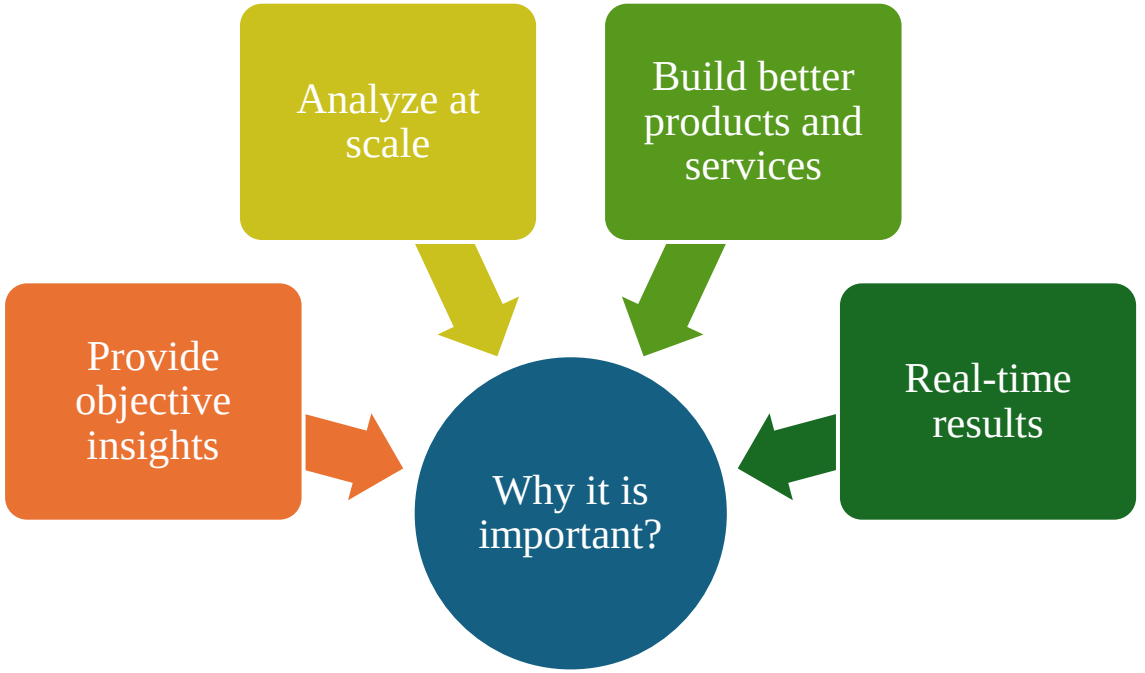
How does it work?

- **Pre-processing** (Tokenization, Lemmatization, Stop-word removal)
- **Keyword Analysis** (extracted keywords and give them a sentiment score)

Why Sentiment Analysis is Important in Business?

- Companies have large volumes of text data like emails, customer support chat transcripts, social media comments, and reviews.
- Sentiment analysis tools can scan this text to automatically determine the author's attitude towards a topic.
- Companies use the insights from sentiment analysis to improve customer service and increase brand reputation.





https://aws.amazon.com/what-is/sentiment-analysis/#:~:text=Sentiment%20analysis%20is%20an%20application,before%20providing%20the%20final%20result.

Sentiment Analysis Vs Semantic Analysis

Sentiment Analysis

- focuses on determining the emotional tone expressed in a piece of text.
- classify the sentiment as positive, negative, or neutral, especially valuable in understanding customer opinions, reviews, and social media comments.
- used to identify the prevailing sentiment and gauge public or individual reactions to products, services, or events.

Semantic Analysis

- Aims to comprehend the meaning and context of the text.
- Understand the relationships between words, phrases, and concepts in a given piece of content.
- Semantic analysis considers the underlying meaning, intent, and the way different elements in a sentence relate to each other.
- Crucial for tasks such as question answering, language translation, and content summarization, where a deeper understanding of context and semantics is required.

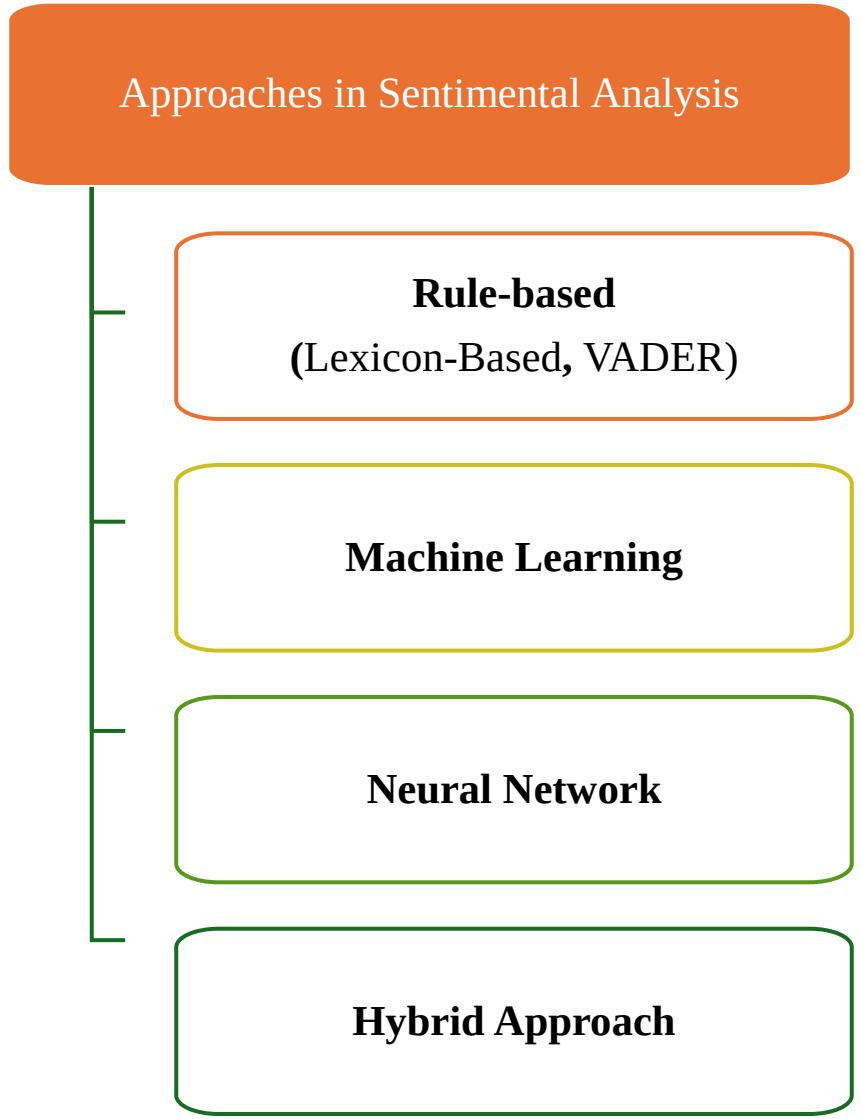
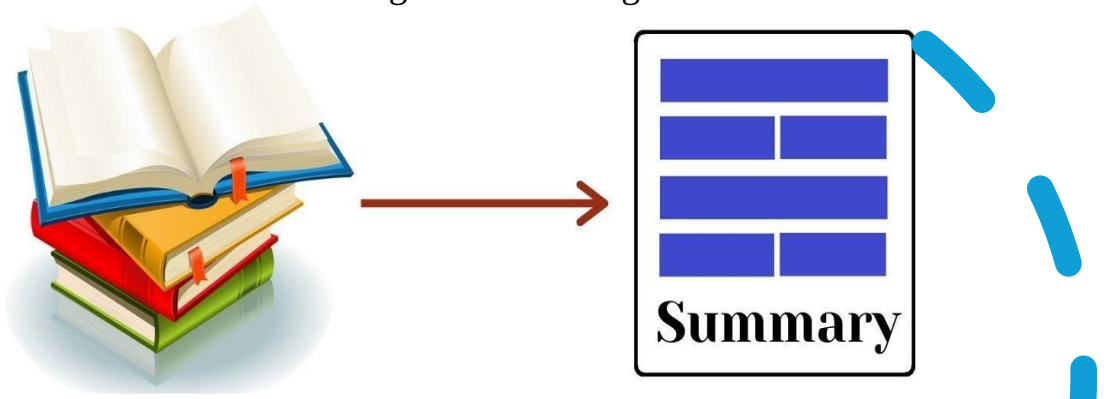
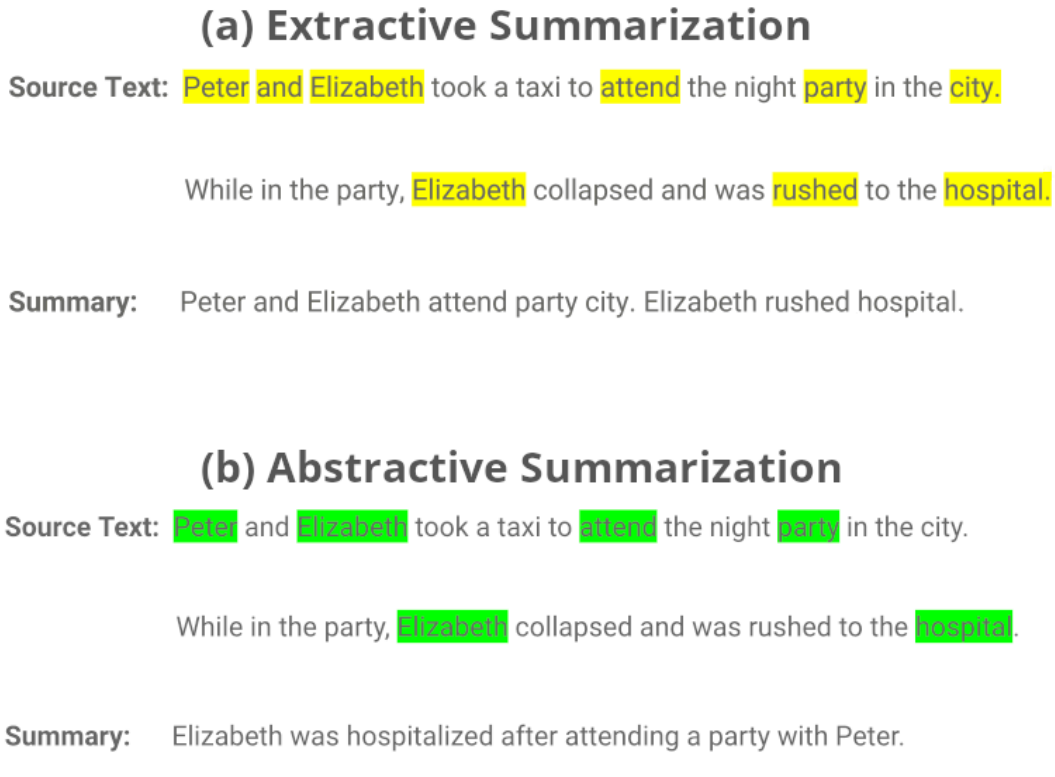


Image Source: Google





Text summarization

- Process of creating a concise and coherent summary of a longer text while retaining its key information and main points
- Task in information retrieval and document analysis, helping users quickly grasp the essential content of a document without reading the entire text

Types of Text Summarization

Extractive Summarization:

- Involves selecting and combining important sentences or phrases directly from the original text to create a summary.
- It doesn't generate new sentences but rather extracts existing ones.

Abstractive Summarization:

- Generates a summary by understanding the text's meaning and creating new sentences that convey the main ideas.
- Often involves natural language generation techniques.

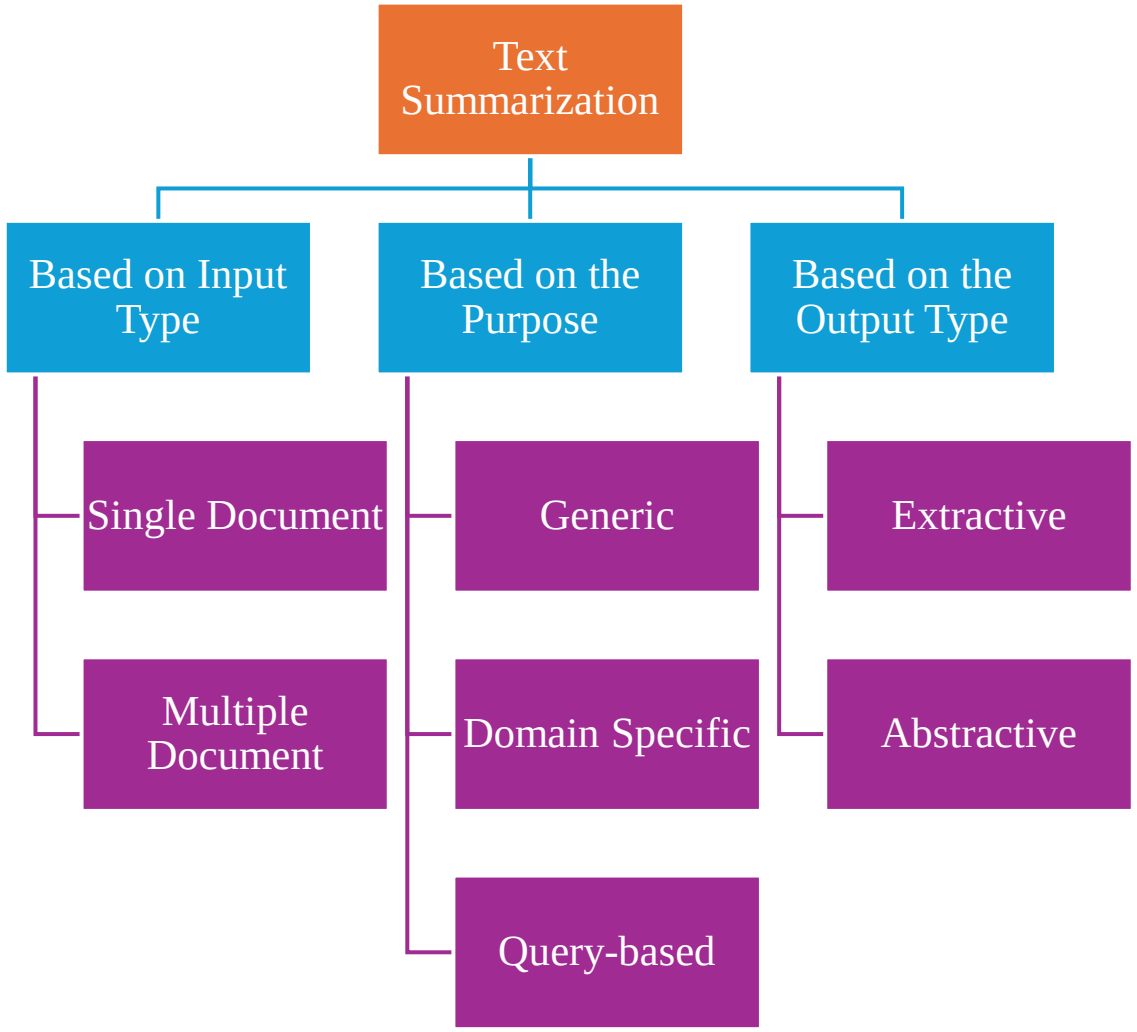
Text summarization (Cont.)

Extractive Summarization Techniques:

- **Graph-based Methods**: Use graph algorithms (e.g., TextRank, PageRank) to identify important sentences based on their connections in the text graph.
- **Machine Learning Models**: Supervised or unsupervised machine learning algorithms (e.g., Support Vector Machines, Clustering) to rank sentences based on features like sentence length, word frequency, and semantic similarity.

Abstractive Summarization Techniques:

- **Deep Learning Models**: Recurrent Neural Networks and transformer models like BERT or GPT generate summaries by understanding the context, semantics, and relationships within the text.



Why automatic text summarization?

- Summaries reduce reading time.
- When researching documents, summaries make the selection process easier.
- Automatic summarization improves the effectiveness of indexing.
- Automatic summarization algorithms are less biased than human summarization.
- Personalized summaries are useful in question-answering systems as they provide personalized information.
- Using automatic or semi-automatic summarization systems enables commercial abstract services to increase the number of text documents they can process.
- Summaries make information more accessible to a broader audience, including individuals with limited time, attention span, or reading abilities.
- Can process text in multiple languages, enabling cross-language summarization and facilitating information access across diverse linguistic contexts
- Summarization aids decision-making by presenting important information in a condensed and digestible format.
- Summaries allow readers to skim through the main ideas and concepts of a document without delving into every detail.
- Summaries act as a quick reference point for finding relevant information within documents.

Chatbot

- Artificial intelligence (AI) program designed to simulate conversations with users in natural language
- NLP transforms text into structured data that the computer can understand
- customer service, information retrieval, task automation, entertainment, and more
- Standard bots don't use AI, so their interactions usually feel less natural and human.

Types of Chatbots:

Rule-Based Chatbots:

- Follow predefined rules and patterns to respond to user inputs.
- They are relatively simple and have limited capabilities but can handle specific tasks effectively.

Machine Learning-Based Chatbots:

- Use machine learning algorithms to learn from data and improve their conversational abilities over time.
- Understand context handle more complex conversations, and adapt to user preferences.

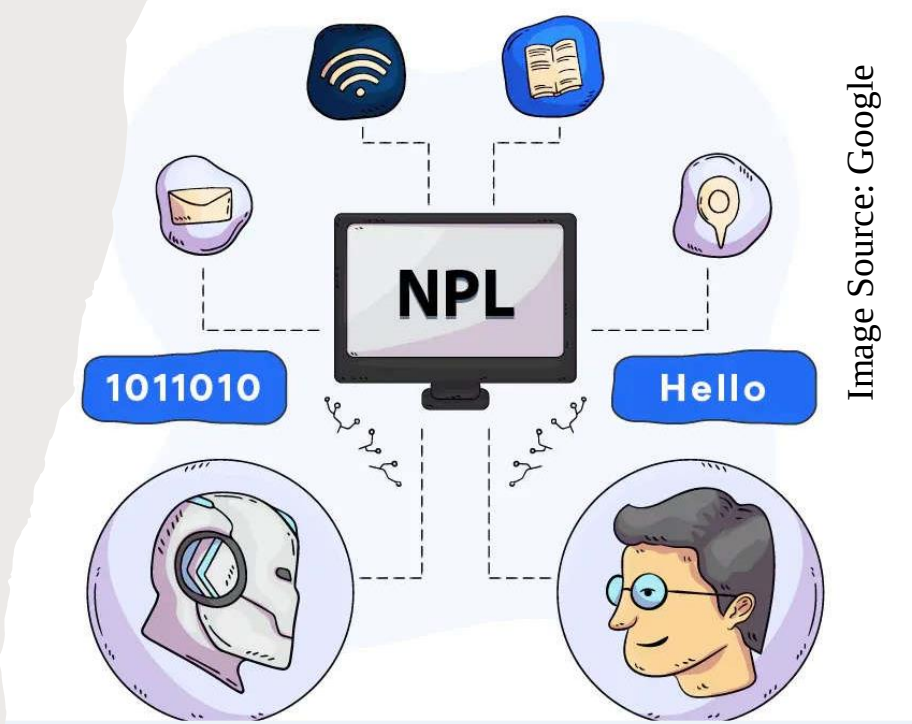
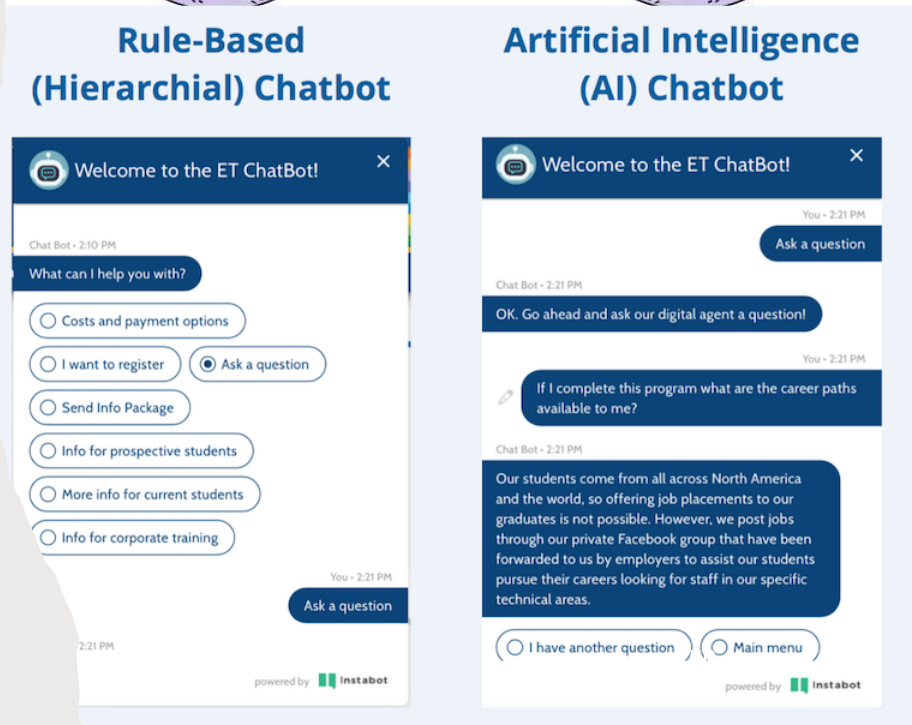
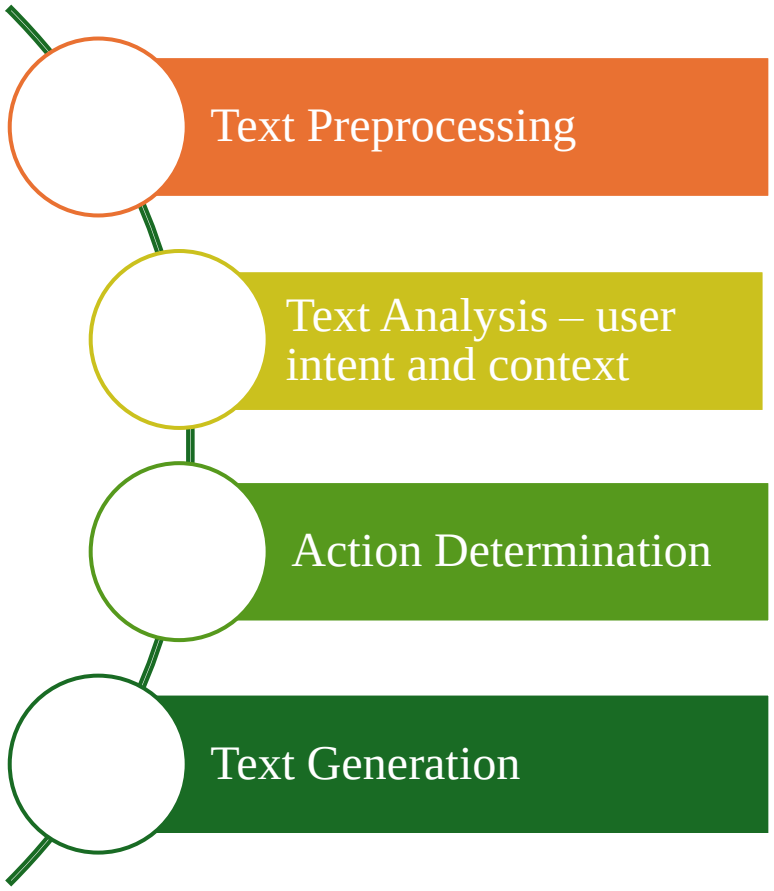


Image Source: Google

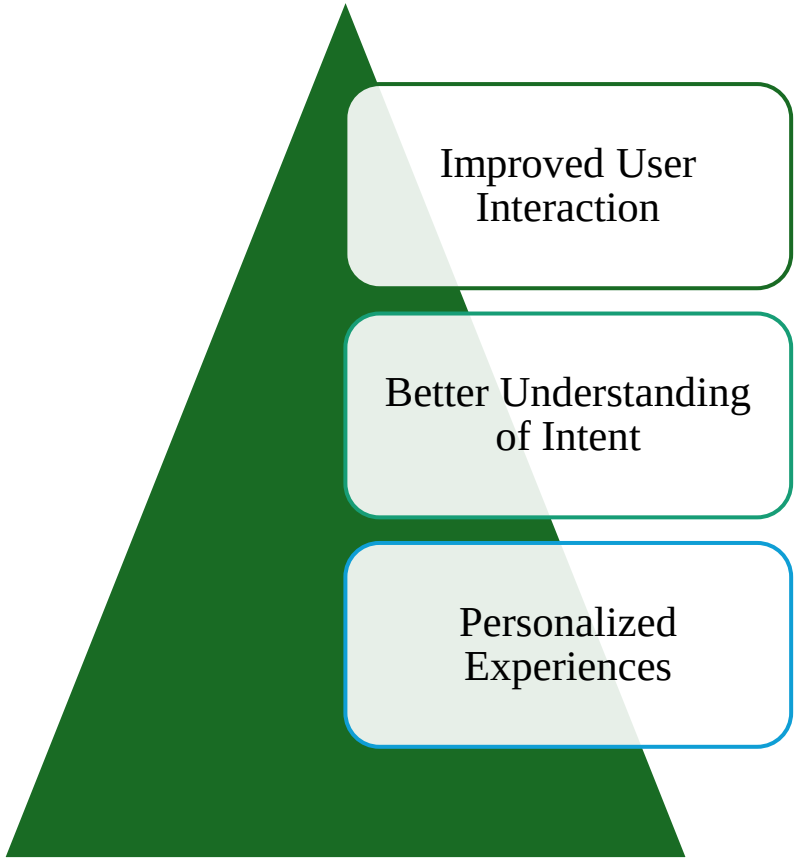


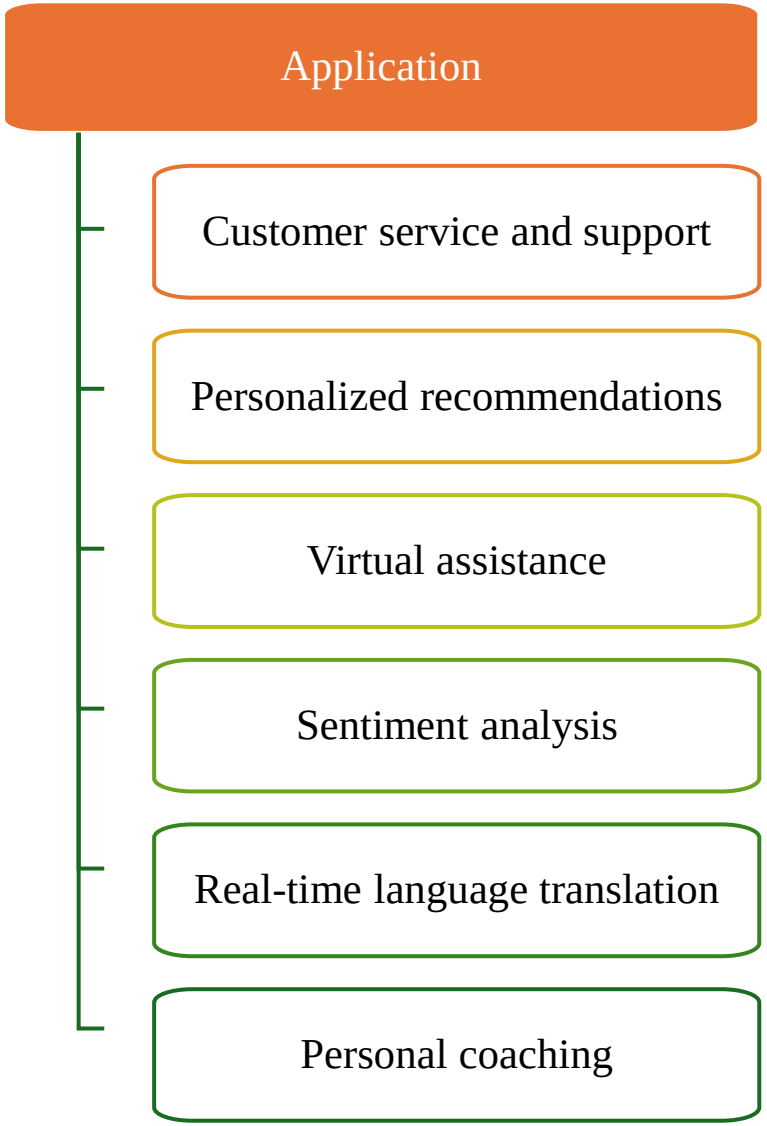
Chatbot (Cont.)

How Does an NLP Chatbot Work?



Why NLP is a Must for Chatbot





https://doubletick.io/blog/nlp-chatbots

Components of a Chatbot

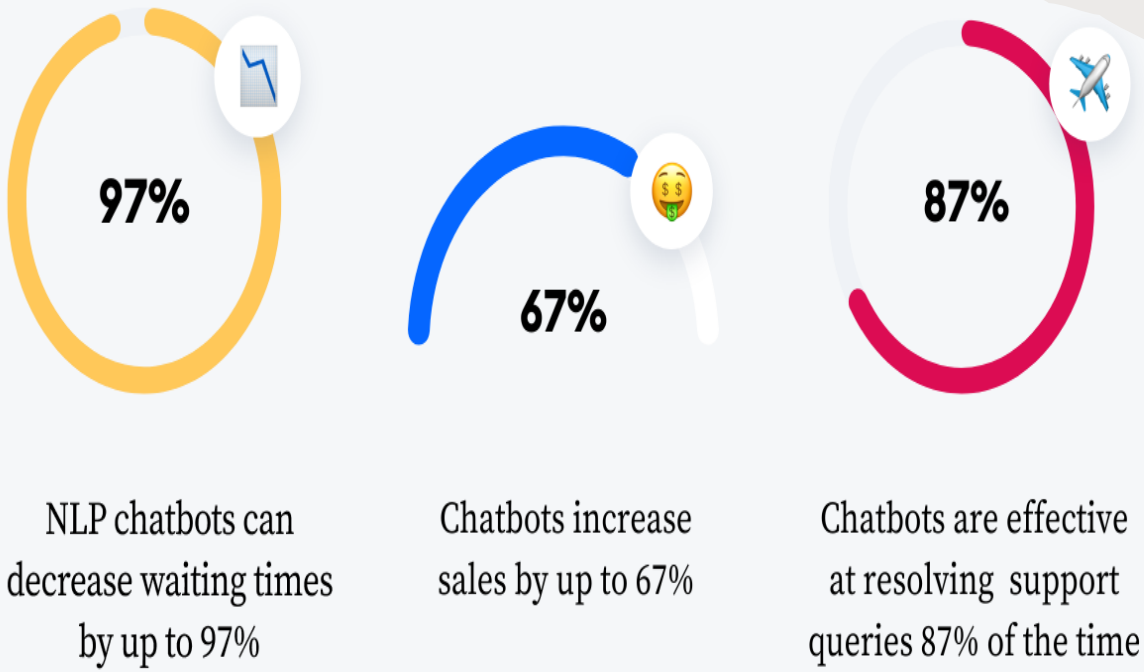


Image Source: Google

NLP Engine

- Processes user inputs to understand intents, extract entities, and generate responses.
- Includes tasks like tokenization, part-of-speech tagging, named entity recognition, and sentiment analysis.

Dialog Management

- Manages the flow of conversation, handles context, maintains state, and determines appropriate responses based on user inputs and system knowledge.

Backend Integration

- Connects with external systems, databases, APIs, or services to fetch information, perform actions, and fulfill user requests.

User Interface

- Provides the interface for users to interact with the chatbot, which can be a text-based interface (e.g., messaging platforms, websites) or voice-based interface (e.g., virtual assistants).

Development Approaches and Functionality

- **Code-Based Development**: Involves writing code using programming languages (e.g., Python, Java) and NLP libraries (e.g., NLTK, spaCy) to build and train chatbots from scratch.
- **Chatbot Platforms**: Use pre-built chatbot development platforms (e.g., Dialogflow, Microsoft Bot Framework, IBM Watson Assistant) that offer tools, APIs, and frameworks for designing, training, and deploying chatbots with minimal coding.
- **Custom Development**: Combines code-based development with chatbot platforms to create highly customized solutions tailored to specific use cases and requirements.

Functionality

- **Information Retrieval**: Provides information, answers questions, and offers recommendations based on user queries.
- **Task Automation**: Performs automated tasks, such as scheduling appointments, making reservations, placing orders, and handling routine customer inquiries.
- **Conversational Engagement**: Engages users in meaningful conversations, maintains user interest, and provides personalized experiences.
- **Feedback and Learning**: Collects user feedback, learns from interactions, and improves conversational abilities and performance over time.

Difference between NLP, NLG, NLU, and NLI

NLP (Natural Language Processing)	NLG (Natural Language Generation)	NLU (Natural Language Understanding)	NLI (Natural Language Interface)
Enabling computers to understand, interpret, and generate human language in a way that is meaningful	Generation of human-like text or speech from structured data or non-linguistic input	Enabling computers to understand and interpret human language input	Interface or system that allows users to interact with computers or systems using natural language input
Text preprocessing, tokenization, part-of-speech tagging, named entity recognition, syntactic analysis	Converting data or information into natural language text, generating summaries, creating stories, composing emails	Intent classification, entity recognition, sentiment analysis, context understanding, and discourse analysis	Encompasses both NLU and NLG capabilities to interpret user queries or commands, extract meaning, generate appropriate responses
	Template-based generation, Rule-based generation, statistical approaches, and neural network-based generation	chatbots, virtual assistants, voice recognition systems, information retrieval, and sentiment analysis	chatbots, voice assistants, search engines, and smart home devices

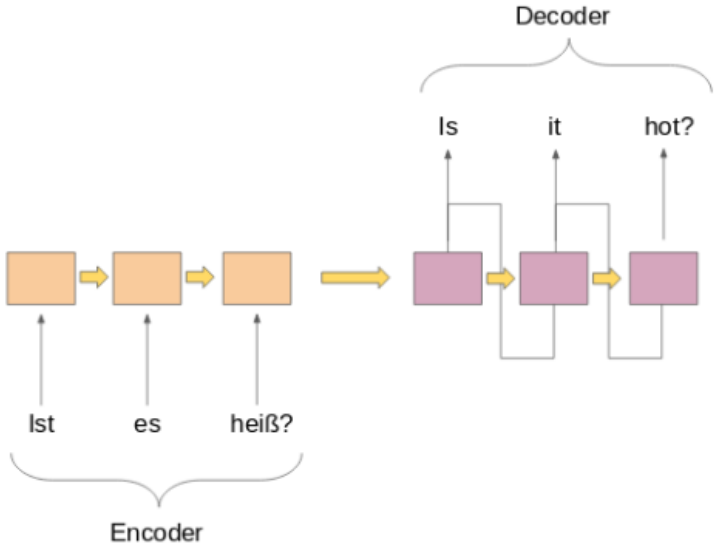


Image Source: Google

Machine Translation

- Automated process of translating text or speech from one language to another using computational algorithms and techniques
- Machine translation software in the source language and let the tool automatically transfer the text to the selected target language.

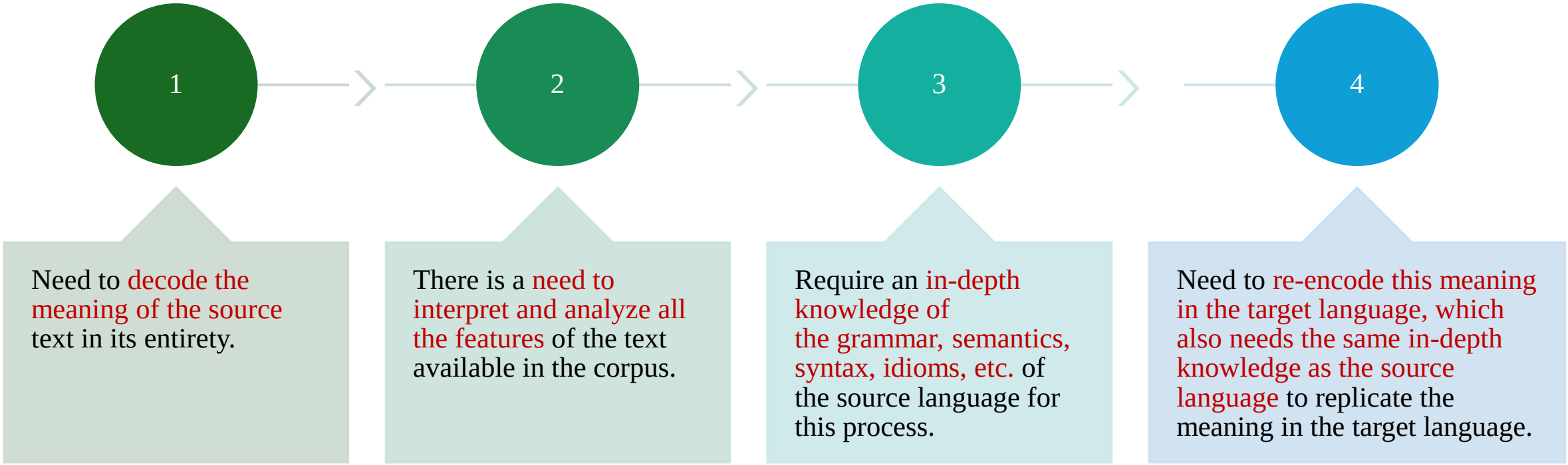
Automated Translation vs Machine Translation

- Automated translation refers to any triggers built into a traditional computer-assisted translation tool (CAT tool) or cloud translation management system (TMS) to execute manual or repetitive tasks related to translation
- Machine translation is about converting text from one natural language to another using software.
- In other words, there's no human input involved as in traditional translation.

Process of Machine Translation

- The basic requirement in the complex cognitive process of machine translation is to understand the meaning of a text in the original (source) language and then restore it to the target (sink) language.

The primary steps in the machine translation process are



https://www.scaler.com/topics/nlp/machine-translation-in-nlp/

Machine Translation (Cont.)

Types of Machine Translation:

Rule-based Translation: (1950-1980)

- Relies on linguistic rules and dictionaries to translate text.
- Requires a substantial amount of linguistic knowledge and often lacks flexibility.

Statistical Machine Translation (SMT): (1990-2014)

- SMT uses statistical models that learn from large bilingual corpora to generate translations.
- Works well for common language pairs but may struggle with less common languages or complex sentence structures.

Neural Machine Translation (NMT): (2014 to till)

- NMT is the latest paradigm in machine translation, using deep learning models such as sequence-to-sequence models with attention mechanisms.
- NMT has shown significant improvements in translation quality, especially for long and complex sentences.

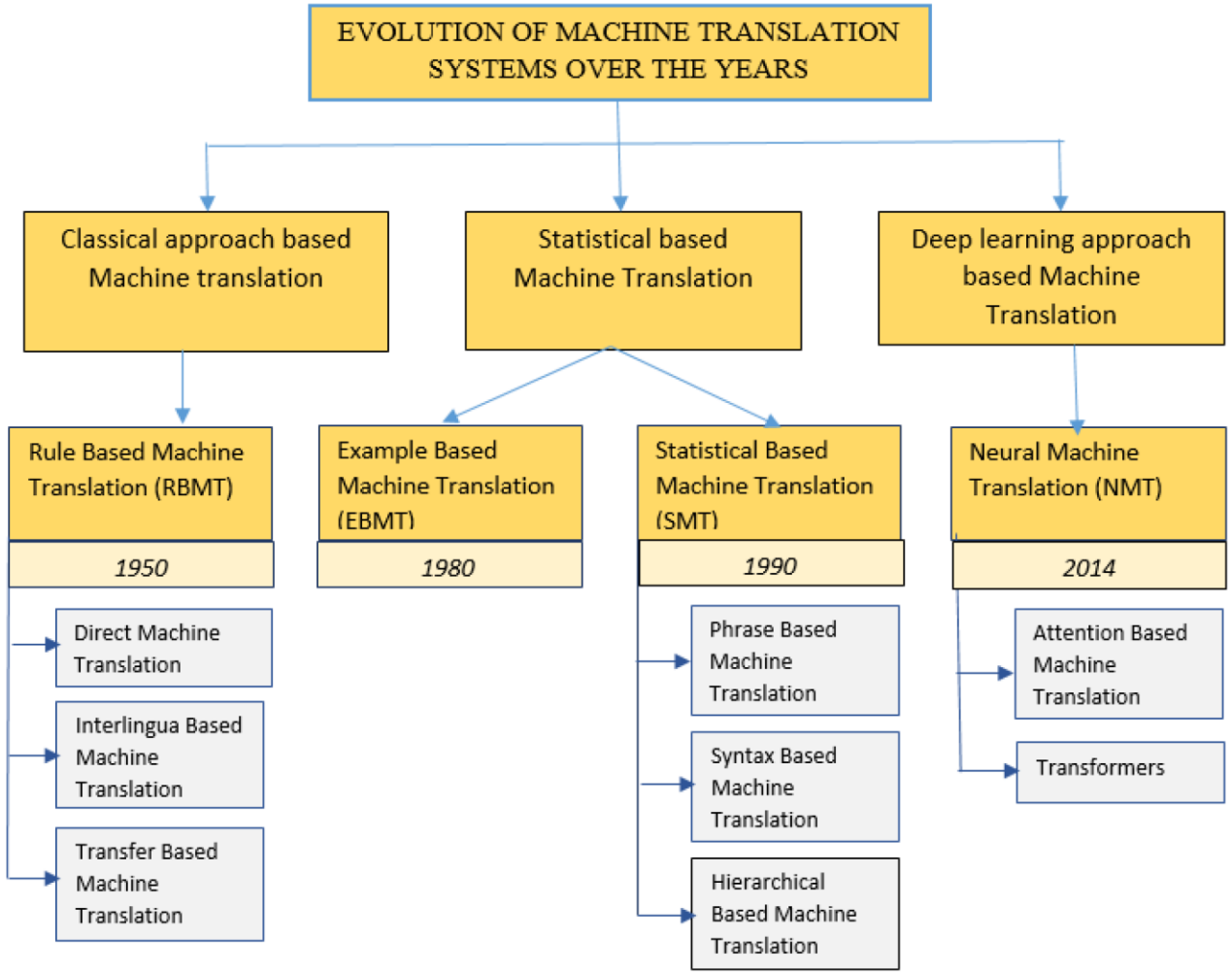


Image Source: Google

Machine Translation (Cont.)

Neural Machine Translation (NMT) Process:

Encoder-Decoder Architecture:

- NMT models typically use an encoder-decoder architecture.
- The encoder processes the input text and converts it into a fixed-dimensional vector representation (encoding).
- The decoder then generates the translated text based on this encoding.

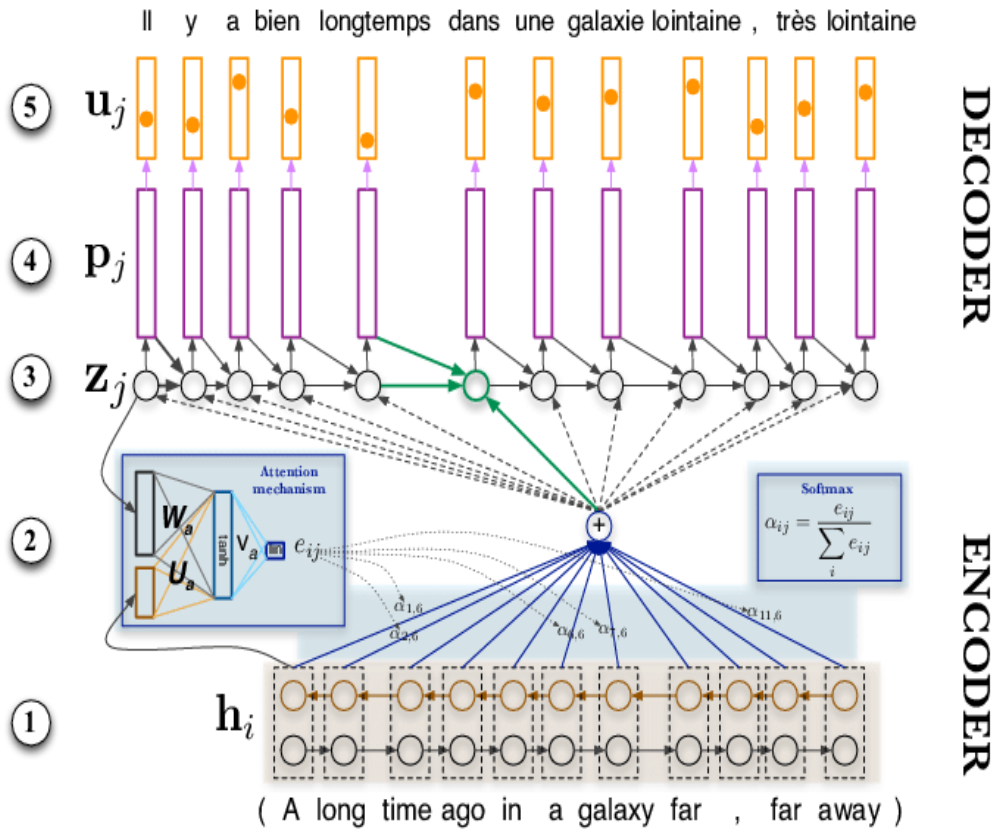
Attention Mechanism:

- Attention mechanisms in NMT allow the model to focus on relevant parts of the input sentence during translation, improving the accuracy of translations, especially for long sentences.

Training Data:

- NMT models require large amounts of parallel data (source-target language pairs) for training.
- The quality and diversity of the training data significantly impact the model's performance.

Image Source: Google



Challenges in Machine Translation

Ambiguity: Many words and phrases have multiple meanings or interpretations, leading to translation ambiguity.

Idioms and Cultural Nuances: Idiomatic expressions, cultural references, and context-specific language nuances can be challenging for machine translation systems.

Rare Languages: Limited availability of training data and resources for less common languages can hinder accurate translations.

Domain-specific Translation: Translating specialized or domain-specific content (e.g., medical or legal texts) accurately requires domain knowledge and specialized training data.

Text to Speech

- Text-to-speech (TTS) in NLP refers to the conversion of written text into spoken speech, a Type of speech synthesis that transforms written text into spoken words using computer algorithms
- Enables machines to communicate with humans in a natural-sounding voice by processing text into synthesized speech
- TTS - typically uses a combination of linguistic rules and statistical models to generate synthetic speech
- Speech synthesis refers to the process of using a computer to produce artificial human speech.
- Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database.
- TTS systems can then use NLP understanding to generate more accurate synthetic speech reflecting the input text's intended meaning.
- Elevenlabs, amazon's Polly, and Deepgram's aura represent some of the cutting-edge AI & ML developments.

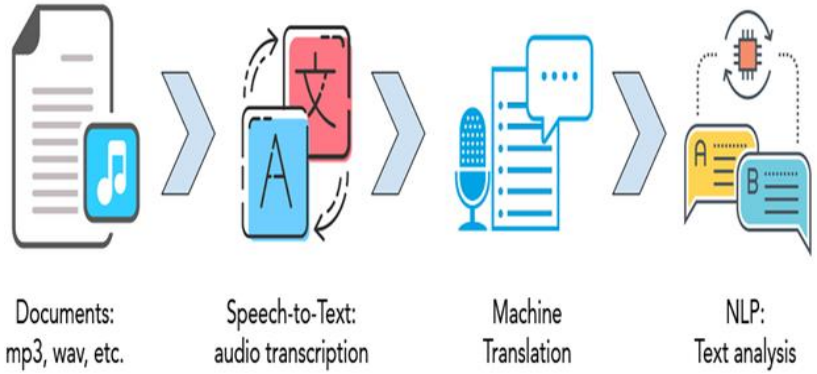


Image Source: Google

Why do we need AI for TTS?

Necessity of AI in realistic voices

Machine learning's role

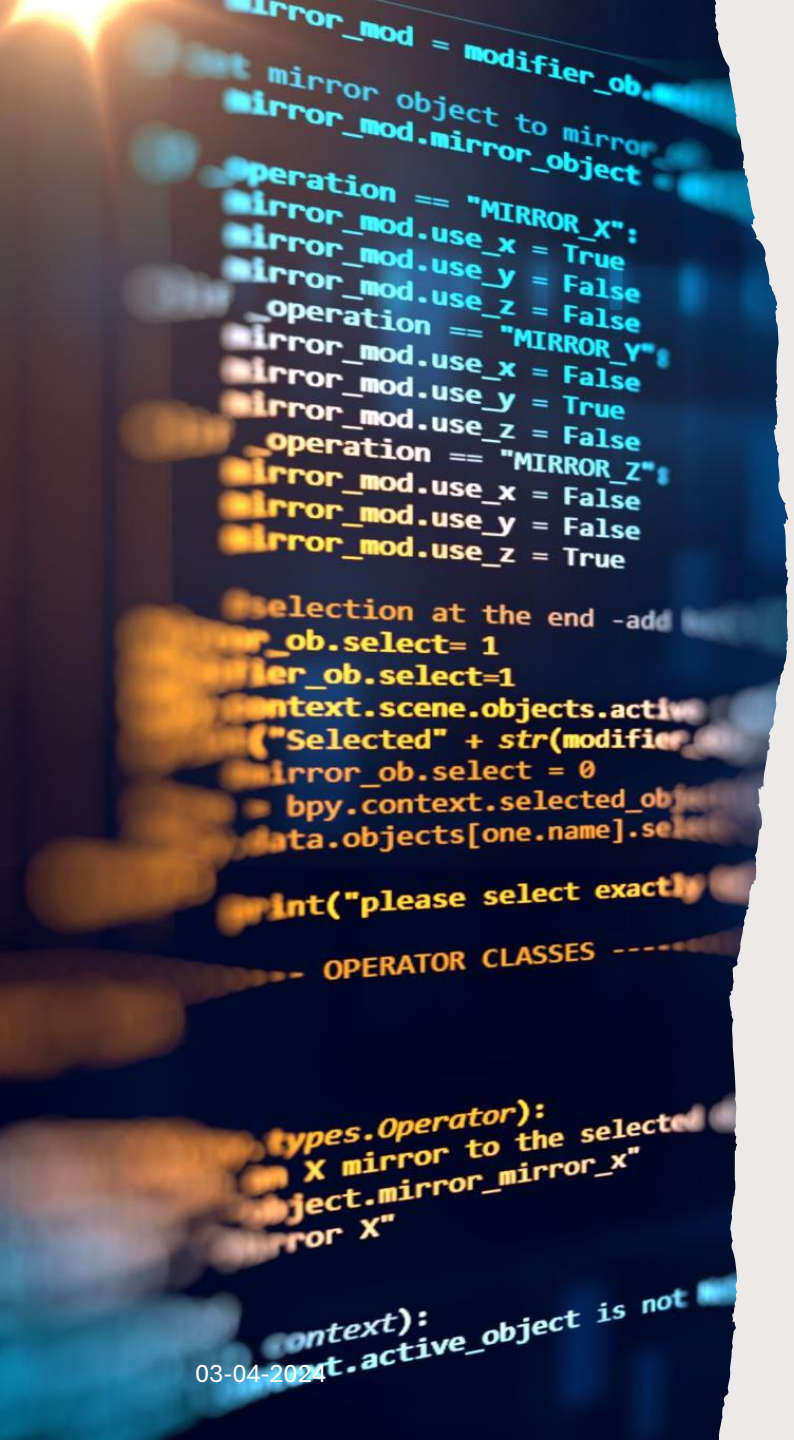
The spectrogram's function

Waveforms in speech synthesis

Training AI models

Overcoming speech challenges

https://deepgram.com/ai-glossary/text-to-speech-models



Text to Speech (Cont.)

- A text-to-speech system (or “engine”) is composed of two parts: a **front-end** and a **back-end**.
- The front end has two major tasks.
  - First, it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often called text normalization, pre-processing, or tokenization.
  - The front end then assigns phonetic transcriptions to each word and divides and marks the text into prosodic units, like phrases, clauses, and sentences.
- The back-end — often referred to as the synthesizer — then converts the symbolic linguistic representation into sound.

There are different ways to perform speech synthesis:

- Concatenative TTS, (Domain-Specific Synthesis, Unit Selection Synthesis, Diphone Synthesis)
- Formant Synthesis,
- Parametric TTS, and
- Hybrid approaches.

Text to Speech basic process

Text Input: Process begins with a piece of written text as input, text can be in various languages and formats, such as plain text, web pages, or documents.

Text Analysis : Text undergoes linguistic analysis to understand its structure, including sentence segmentation, part-of-speech tagging, and syntactic parsing.

Analysis helps in generating more natural-sounding speech

Speech Synthesis: Synthesized speech generation involves converting the analyzed text into spoken words.

The step includes prosody (intonation, rhythm, and stress) modeling to mimic human-like speech patterns.

Audio Output : Final output is an audio file or real-time speech output that can be played through speakers, headphones, or integrated into applications.

Text to Speech Techniques

Techniques and Models:

Concatenative TTS:

- Uses a database of recorded speech segments (phonemes, diphones, or larger units) to piece together and synthesize the desired speech.
- offers good quality but may lack flexibility and naturalness.

Parametric TTS:

- Parametric TTS generates speech using mathematical models that simulate human vocal tract physiology.
- These models can be trained on linguistic and acoustic data to produce more natural and expressive speech.

Neural TTS:

- Neural TTS, powered by deep learning techniques such as sequence-to-sequence models and WaveNet, has gained popularity for its ability to generate highly natural and expressive speech, including variations in tone, emotion, and speaking style.

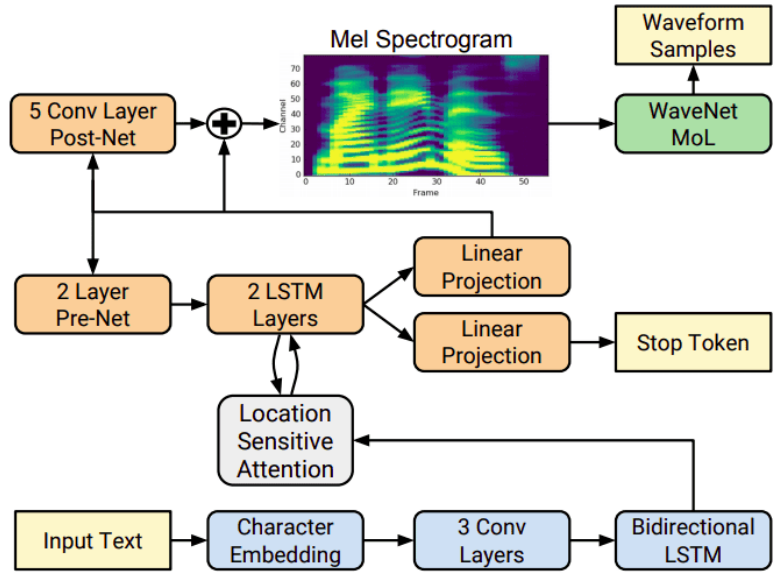
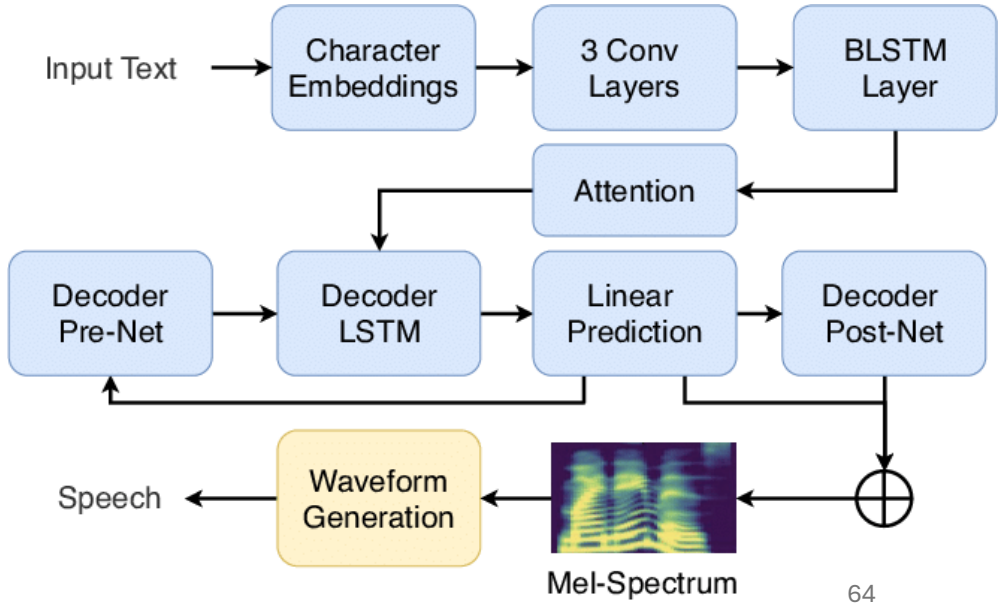


Image Source: Google



Commonly used tools and libraries for text-to-speech in NLP

- **Google Text-to-Speech (TTS)**: Provides high-quality and natural-sounding speech synthesis with various voices and languages.
- **Amazon Polly:** Offers lifelike speech synthesis with customizable voice styles and expressive capabilities.
- **Microsoft Azure Text-to-Speech:** Provides customizable speech synthesis with neural network-based models for natural-sounding output.
- **Open-source TTS frameworks:** Mozilla TTS, Tacotron, and WaveNet that allow customization and development of TTS systems.

Challenges in TTS

Naturalness:

- Achieving natural-sounding speech with proper intonation, rhythm, and emphasis is a key challenge. Neural TTS models have significantly improved naturalness compared to earlier techniques.

Multilingual Support:

- TTS systems need to support multiple languages and accents, requiring robust linguistic resources and models for each language.

Emotional Speech:

- Some TTS systems can generate speech with emotional variations (e.g., happy, sad, angry), enhancing user engagement and interaction.

Real-Time Processing:

- Real-time TTS, where speech is generated dynamically as text is inputted, requires efficient algorithms and computational resources for seamless performance.

Speech to Text

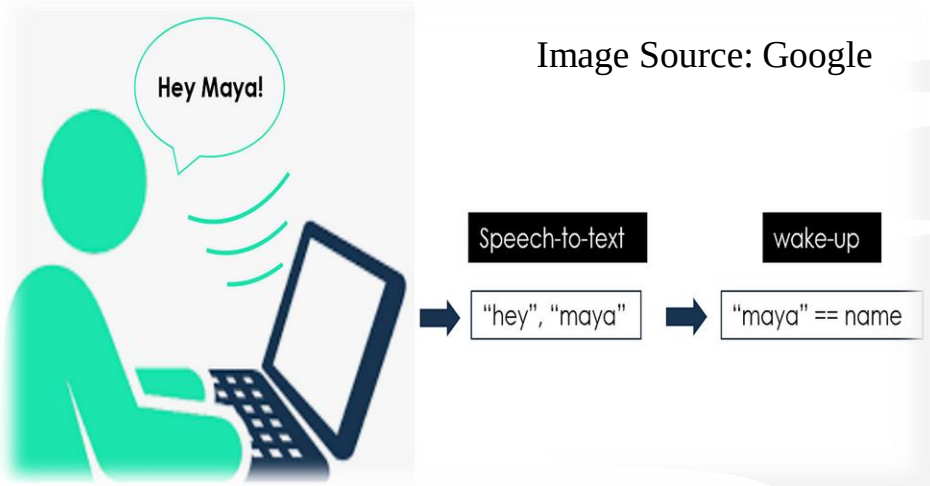
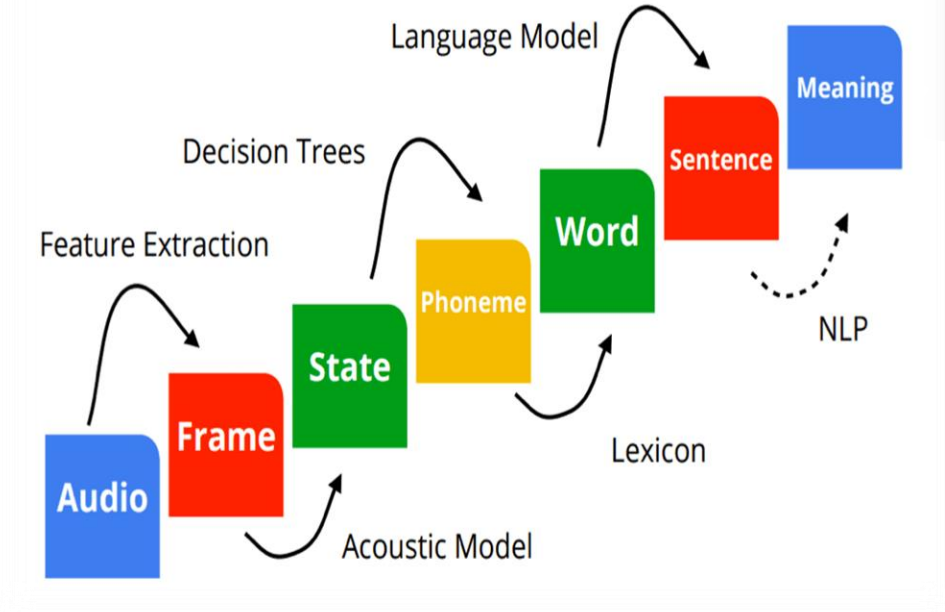


Image Source: Google

- Speech-to-text (STT) in NLP refers to the process of converting spoken language into written text.
- Automatic caption generation for videos, dictation to generate reports, creating transcriptions of audio recordings, Voice assistants, and accessibility tools for individuals with disabilities.



To perform this task, modern models use transformers-based deep learning models, and out of those:

- **Wav2Vec 2.0,** created and shared by a Facebook researcher on wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations
- **HuBERT,** also proposed by Facebook on HuBERT: Self-Supervised Speech Representation Learning
- Main differences between Wav2Vec 2.0 and HuBERT - how they process the audio input and the loss function to measure the performance of the outputs and backpropagate the errors during training.

https://www.johnsnowlabs.com/converting-speech-to-text-with-spark-nlp-and-python/

How does speech-to-text work?

When sounds come out of someone's mouth to create words, it also makes a **series of vibrations**.

Speech to text technology works by picking up on these vibrations and **translating them into a digital language through an analog to digital converter.**

The analog-to-digital-converter takes sounds from an audio file, measures the waves in detail, and **filters them to distinguish the relevant sounds.**

The sounds are then **segmented into hundredths or thousandths of seconds and are then matched to phonemes**.

A phoneme is a unit of sound that distinguishes one word from another in any given language. For example, there are approximately 40 phonemes in the English language.

The phonemes are then **run through a network via a mathematical model** that compares them to well-known sentences, words, and phrases.

The text is then **presented as text** or a computer-based demand based on the audio's most likely version.

https://aws.amazon.com/what-is/speech-to-text/

STT systems typically involve several stages

Audio Preprocessing

- Input audio is preprocessed to remove noise, normalize volume levels, and potentially separate multiple speakers if needed.

Feature Extraction

- Preprocessed audio is converted into a format suitable for analysis.
- Mel-frequency cepstral coefficients (MFCCs) or spectrograms to represent the audio's frequency content over time.

Acoustic Model

- Relationship between input audio features and phonemes (basic units of sound in a language).
- Techniques like Hidden Markov Models (HMMs), Gaussian Mixture Models (GMMs), or deep learning models such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs).

Language Model

- Acoustic model is combined with a language model that predicts the most likely sequence of words given the audio input, corrects errors, and improves overall transcription accuracy.

Decoding

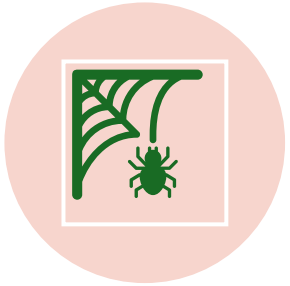
- The combined output from the acoustic and language models is decoded to generate the final text transcription.

Commonly used tools and libraries for speech-to-text in NLP include

Google Cloud Speech-to-Text: Provides accurate transcription through deep learning models.

IBM Watson Speech to Text: Offers real-time transcription with customization options.

Mozilla DeepSpeech: An open-source speech-to-text engine based on deep learning.



CMU Sphinx: A toolkit for speech recognition using HMMs and other algorithms.

Let’s explore some important concepts in deep-learning for NLP

Attention Mechanism

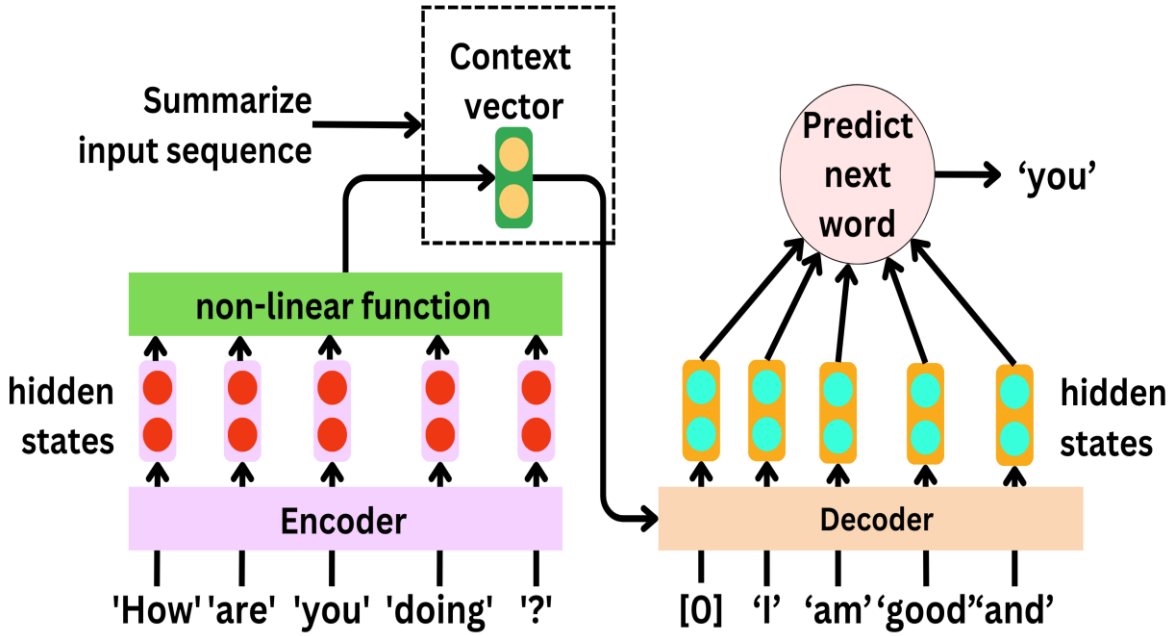


Image Source: Google

Before learning about Attention Mechanism, we should learn about how RNN, LSTM, and GRU are working

- Technique used in neural network models to focus on relevant parts of the input sequence when making predictions or generating output sequences
- developed to increase the performance of encoder decoder(seq2seq) RNN model.
- Solution to the limitation of the Encoder-Decoder model which encodes the input sequence to one fixed length vector from which to decode the output at each time step.
- Allows the model to "pay attention" to certain parts of the data and to give them more weight when making predictions.
- preserve the context of every word in a sentence by assigning an attention weight relative to all other words.
- even if the sentence is large, the model can preserve the contextual importance of each word.

https://medium.com/analytics-vidhya/https-medium-com-understanding-attention-mechanism-natural-language-processing-9744ab6aed6a

How attention mechanisms work in NLP

Contextual Relevance

- In NLP tasks, especially those dealing with sequences like sentences or documents, different parts of the input sequence may contribute differently to the output. For example, in machine translation, certain words in the input sentence may have more influence on the translation of specific words in the output sentence.

Attention Weights

- Attention mechanisms assign weights to each element in the input sequence, indicating how relevant that element is to the current step in processing. These weights are often computed using neural network layers, such as softmax layers, based on the similarity between the current state of the model and each element in the input sequence.

Attention Scores

- To calculate the attention weights, attention mechanisms typically compute attention scores. These scores measure the similarity or relevance between the current state of the model (e.g., the decoder state in machine translation) and each element in the input sequence. Common methods for computing attention scores include dot product attention, additive attention, and multiplicative attention.

Soft Attention

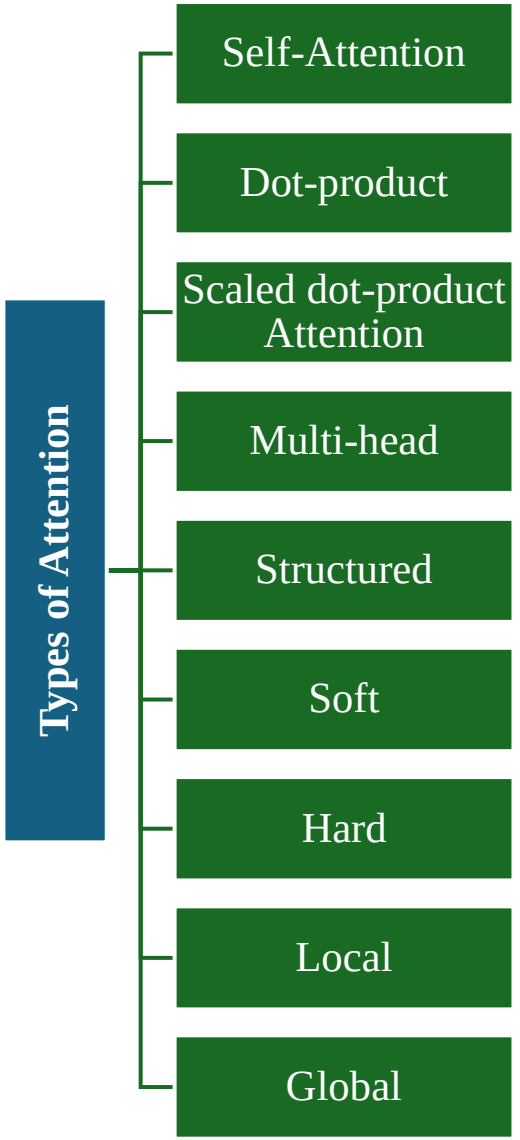
- After computing the attention scores, a softmax function is often applied to convert the scores into attention weights. Soft attention allows the model to consider the entire input sequence but with varying degrees of importance for each element.

Context Vector

- Once the attention weights are determined, a context vector is computed as a weighted sum of the input elements, where the weights are given by the attention weights. This context vector represents the focused information from the input sequence that is relevant to the current step in the model's processing.

Type of Attention Mechanism (Cont.)

- Self-attention, also known as intra-attention, is a specific type of attention mechanism where the input sequence is compared to itself.
- Multi-head attention, the attention mechanism is split into multiple heads, each computing attention independently.
- Soft attention computes attention weights for all elements in the input sequence, typically using a softmax function to normalize these weights.
- Hard attention selects a single element from the input sequence at each step of processing. This selection process is often based on learned probabilities or heuristics.
- Structured Attention allows the attention weights to be learned using a structured prediction model, such as a conditional random field.
- Scaled dot-product Attention, a variant of dot-product attention that scales the dot product by the square root of the key dimension.
- Dot-product Attention computes the attention weights as the dot product of the query and key vectors.
- Local attention restricts the attention mechanism to a specific region or window of the input sequence.
- Global attention considers the entire input sequence when computing attention weights.



https://www.scaler.com/topics/deep-learning/attention-mechanism-deep-learning/

Difference Between Global vs Local Attention

Global Attention

- Model considers the entire input sequence when computing attention weights.
- requires computing attention weights for all elements in the input sequence, leading to higher computational complexity, especially for long sequences.
- allows the model to align input and output sequences more flexibly, as it can attend to any part of the input sequence based on the relevance to the current processing step.
- may struggle with long sequences due to the need to compute attention weights for all elements, leading to memory and computational challenges.

Local Attention

- Restricts the attention mechanism to a specific region or window of the input sequence.
- reduces the computational burden by focusing on a smaller subset of input elements. This can be advantageous for handling long sequences more efficiently, as the attention mechanism only needs to consider a limited number of elements at each step.
- enforces a more constrained alignment between input and output sequences, as it only considers a specific region or window of the input sequence.
- can handle long sequences more efficiently by focusing on relevant parts of the input sequence, mitigating some of the issues associated with processing lengthy data.

Transformers

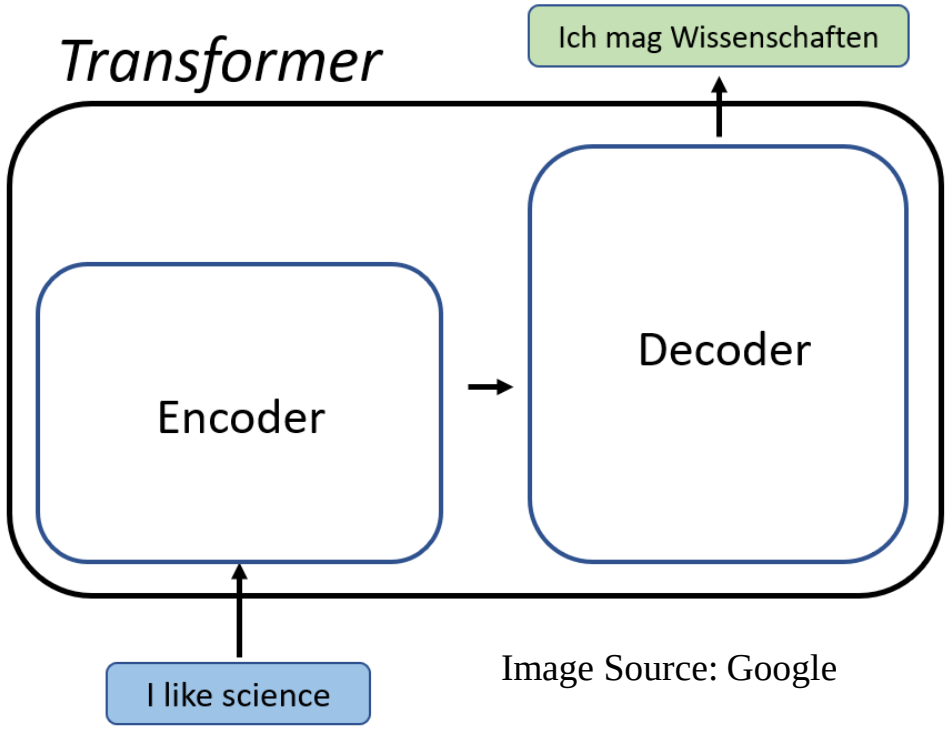


Image Source: Google

- Transformer is a revolutionary deep learning model introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017.
- Profound impact on NLP tasks, especially in areas like machine translation, text generation, and language understanding.
- First encoding the input sentence into a sequence of vectors. This encoding is done using a self-attention mechanism, which allows the model to learn the relationships between the words in the sentence.
- Once the input sentence has been encoded, the model decodes it into a sequence of output tokens. This decoding is also done using a self-attention mechanism.
- The attention mechanism is what allows transformer models to learn long-range dependencies between words in a sentence.
- The attention mechanism works by focusing on the most relevant words in the input sentence when decoding the output tokens.

Transformers (Cont.)

Self-Attention Mechanism: allows the model to weigh the importance of different words in a sentence based on their context within the sentence itself.

This enables the model to capture dependencies and long-range relationships between words more effectively than traditional recurrent or convolutional architectures.

Encoder-Decoder Architecture: Encoder processes the input sequence (e.g., source language sentence) using self-attention layers and feed-forward layers, producing a representation that captures the semantic and contextual information of the input.

The decoder then uses self-attention layers and encoder-decoder attention layers to generate the output sequence (e.g., target language translation).

Positional Encoding: Provides the model with information about the position of each word in the sequence, allowing it to differentiate between words based on their position.

Multi-Head Attention: Allows the model to focus on different parts of the input or output sequence simultaneously, enhancing its ability to capture diverse relationships and patterns.

Feed-forward Networks: Networks process the outputs of the attention layers, adding non-linearity and further modeling capabilities to the architecture.

Layer Normalization and Residual Connections: To stabilize training and improve gradient flow, the Transformer uses layer normalization and residual connections after each sub-layer in the encoder and decoder.

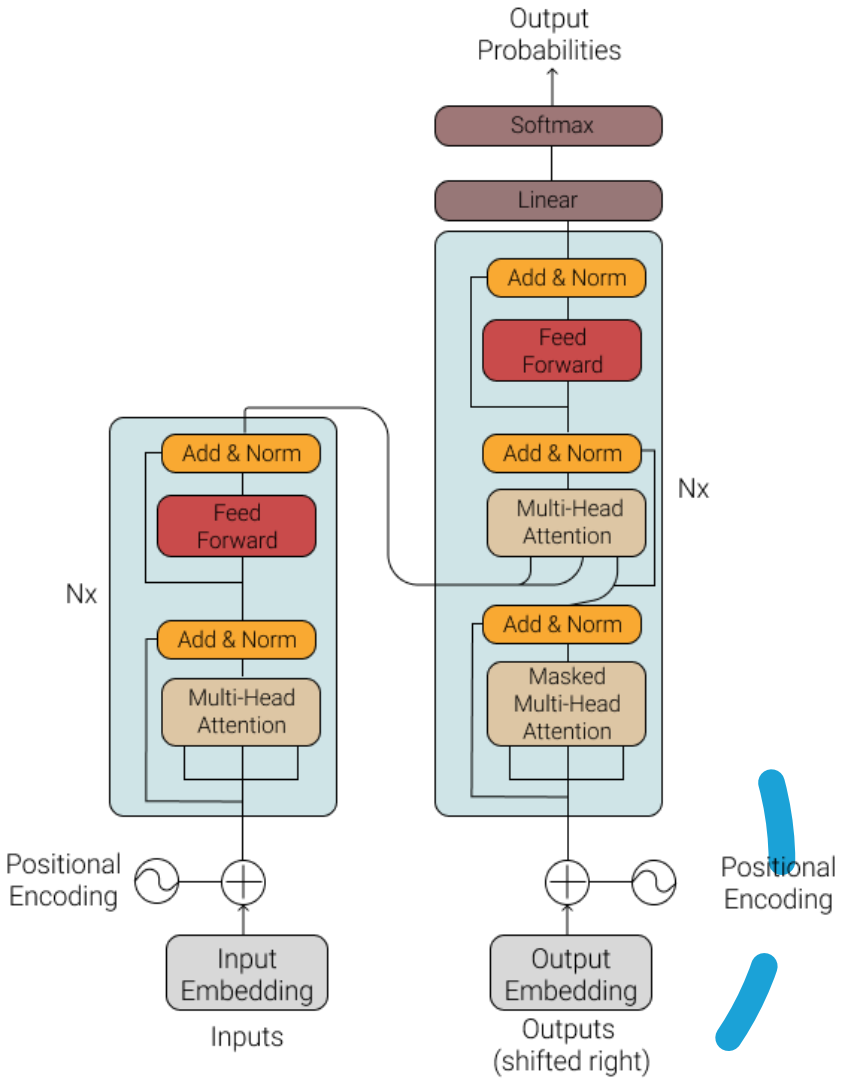
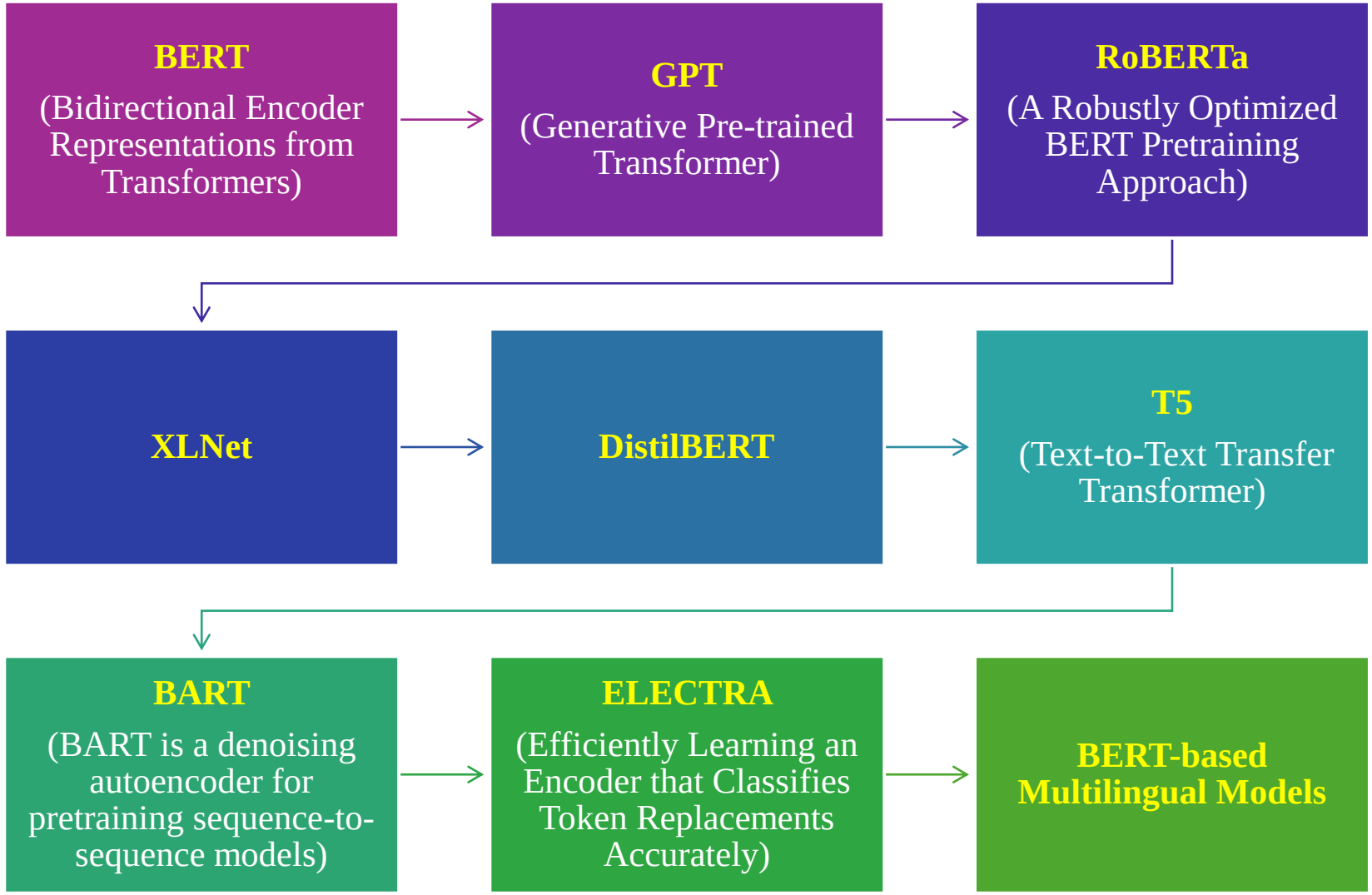


Image Source: Google

Transformer-based models and architectures



Large Language Model (LLM)

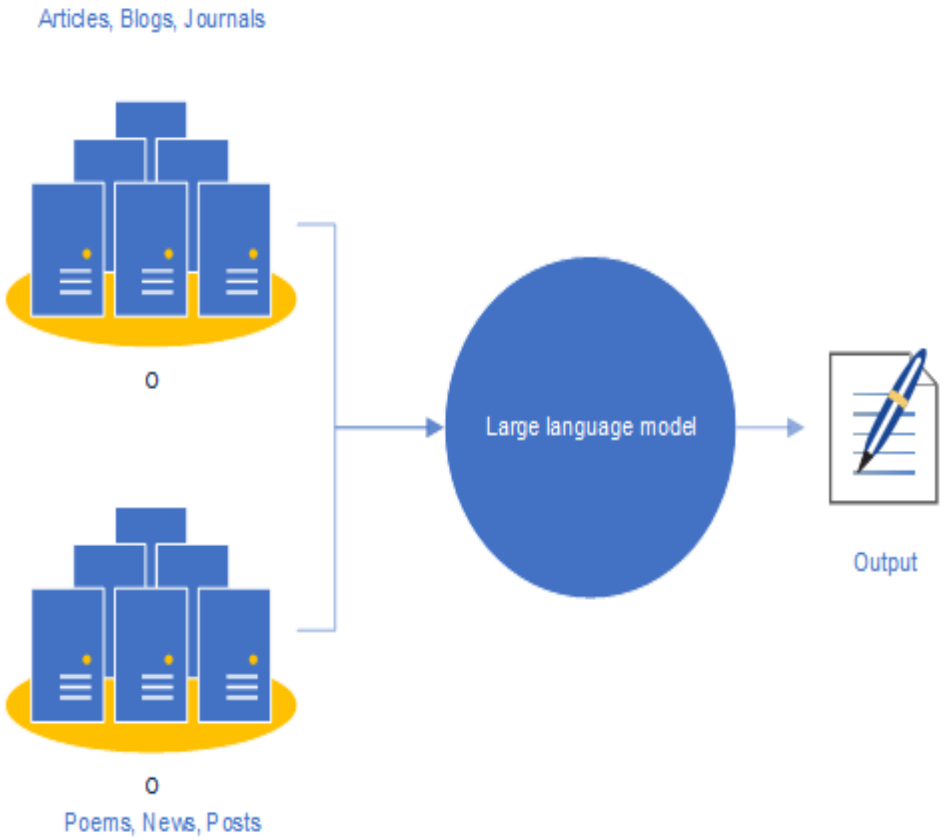


Image Source: Google

- Type of AI model that has been trained on a vast amount of text data to understand and generate human-like text
- Large language models use transformer models and are trained using massive datasets — hence, large.
- This enables them to recognize, translate, predict, or generate text or other content.
- GPT – 3.5, including NLP understanding, text generation, code generation, and more
- Works by processing input text, analyzing patterns, and generating responses based on learned information from the training data.
- Many LLMs are trained on data that has been gathered from the Internet – thousands or millions of gigabytes' worth of text as programmers may use a more curated data set.
- LLMs are then further trained via tuning for particular tasks such as interpreting questions and generating responses or translating text from one language to another.

How a Large Language Model (LLM) works ?

Pre-training

LLMs undergo pre-training on large corpora of text data.

Model learns to predict the next word in a sequence of text given the preceding context

Transformer Architecture

LLMs are often built using the Transformer architecture or its variants.

Tokenization

Text data is tokenized into smaller units, such as words or subwords, before being fed into the LLM.

Breaks down the text into meaningful components that the model can process.

Contextual Representations

LLMs learn to generate contextual representations of words and phrases based on their surrounding context.

Model captures these contextual variations.

Attention Mechanisms

LLMs leverage attention mechanisms to weigh the importance of different words or tokens in a sequence.

Fine-tuning

After pre-training, LLMs can be fine-tuned on specific tasks or domains by further training on task-specific data.

Generation

LLMs can generate human-like text by sampling from their learned language distributions.

Text completion, summarization, question answering, and even creative writing.

Transfer Learning

LLMs leverage transfer learning principles, where knowledge learned during pre-training is transferred and adapted to new tasks with relatively little additional training data.

Large language models use cases

- **Information retrieval:** Think of Bing or Google. Whenever a user uses their search feature, the user relies on a large language model to produce information in response to a query.
- **Sentiment analysis**
- **Text generation:** Large language models are behind generative AI, like ChatGPT, and can generate text based on inputs.
- **Code generation:** Like text generation, code generation is an application of generative AI.
- **Chatbots and conversational AI:** Large language models enable customer service chatbots or conversational AI to engage with customers, interpret the meaning of their queries or responses, and offer responses in turn.
- **Tech:** Large language models are used anywhere from enabling search engines to respond to queries, to assisting developers with writing code.
- **Healthcare and Science:** Large language models can understand proteins, molecules, DNA, and RNA.
- **Customer Service:** LLMs are used across industries for customer service purposes such as chatbots or conversational AI.
- **Legal:** From searching through massive textual datasets to generating legalese, large language models can assist lawyers, paralegals, and legal staff.
- **Banking:** LLMs can support credit card companies in detecting fraud.

Limitations and challenges of large language models

Hallucinations

- When a LLM produces a false output, or that does not match the user's intent.

Security

- Large language models present important security risks when not managed or surveilled properly.
- Can leak people's private information, participate in phishing scams, and produce spam.

Bias

- Data used to train language models will affect the outputs a given model produces.

Consent

- Large language models are trained on trillions of datasets — some of which might not have been obtained consensually.
- When scraping data from the internet, large language models have been known to ignore copyright licenses, plagiarize written content, and repurpose proprietary content without getting permission from the original owners or artists.

Scaling

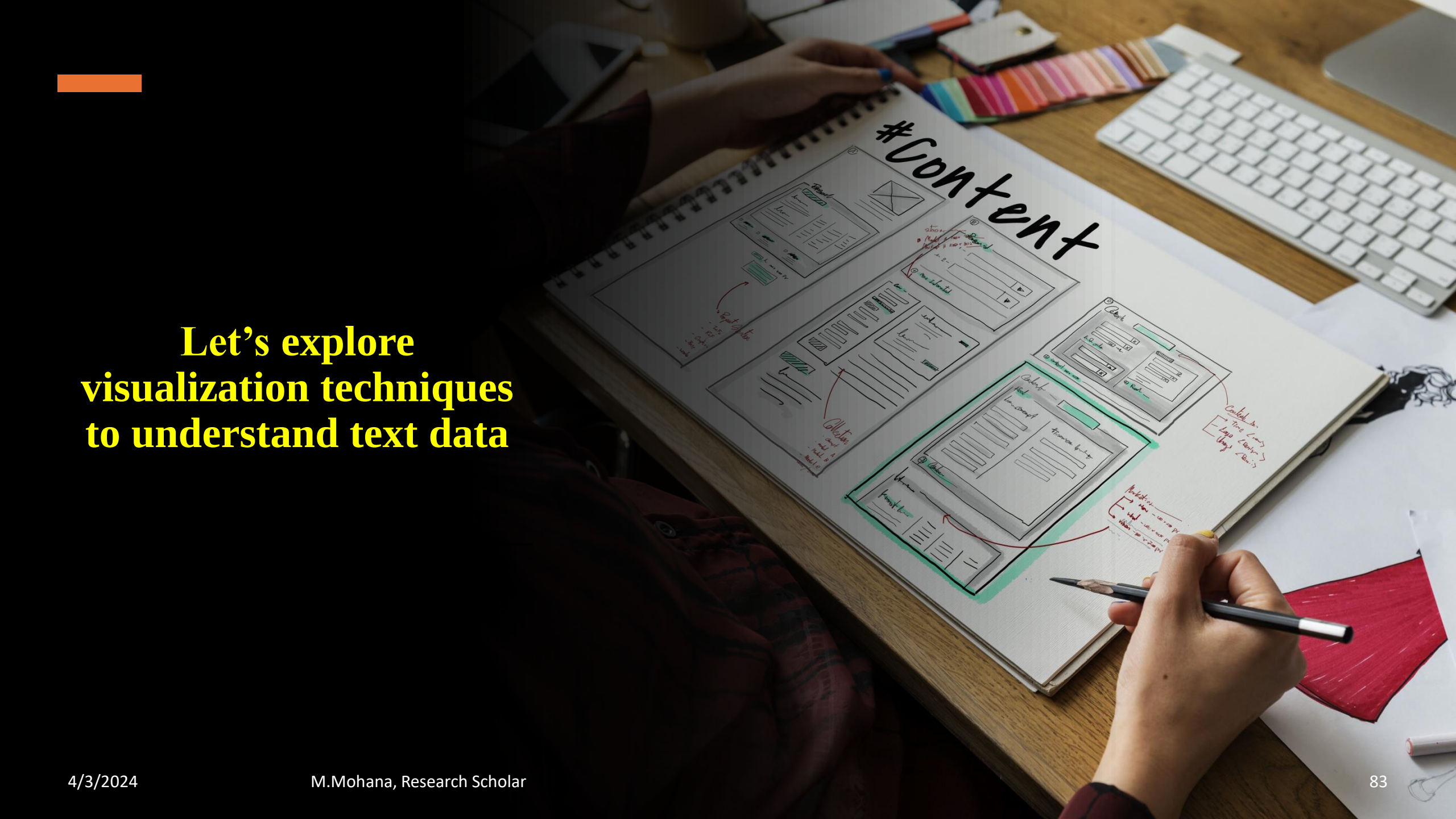
- Can be difficult and time- and resource-consuming to scale and maintain large language models.

Deployment

- Deploying large language models requires deep learning, a transformer model, distributed software and hardware, and overall technical expertise.

https://www.elastic.co/what-is/large-language-models

Let’s explore visualization techniques to understand text data



Visualization in NLP

plotly

Image Source: Google

https://www.numpyninja.com/post/nlp-text-data-visualization

Process of representing and presenting linguistic data, text corpora, language models, or NLP algorithms in a visual format.

Goal of visualization in NLP is to make complex linguistic patterns, semantic relationships, and textual information more accessible and interpretable to users

When choosing a visualization technique, consider the nature of the text data, the insights to convey, and user preferences to understand textual information visually.

Interactive visualizations are useful for exploring and interacting with large text datasets or complex textual structures.

Visualization in NLP (Cont.)

- **Word clouds** display words from a text corpus, with font size indicating word frequency.
- **Bar charts** display word frequencies or other textual data categories.
- **Histograms** show the distribution of word lengths, sentence lengths, or other text-related metrics.
- **Scatter plots** can represent relationships between words or text features.
- **Tree maps** show hierarchical data structures, such as folder structures or topic hierarchies in text.
- **Heatmaps** represent text-related metrics using color gradients.
- **Network graphs** depict relationships between entities in text, such as co-occurrence networks or social networks based on mentions.
- Visualizing topics and their word distributions using bar charts, word clouds, or interactive **topic models**.
- **Time series plots** show changes in text-related metrics over time, such as word frequencies in social media posts or news articles.
- Mapping text data **geographically**, such as sentiment analysis results across regions or locations mentioned in text.
- Adding **annotations, labels, or tooltips** to visualizations to provide context and insights about specific text elements.

Explainable Artificial Intelligence (XAI) techniques in NLP

Attention Mechanisms: attention maps show which words or phrases contribute most to the model's output, providing insights into the model's decision-making process.

Feature Importance: identify the most important features or words in a text that influence the model's predictions (feature permutation importance or SHAP (SHapley Additive exPlanations).

LIME (Local Interpretable Model-agnostic Explanations): used to explain why a specific text input led to a particular prediction from a machine learning model.

ELI5 (Explain Like I'm 5): explain the predictions of models such as text classifiers by highlighting important words or phrases in the input text that contribute to the predicted class.

Integrated Gradients: assigns importance scores to each word or feature in the input text based on how they contribute to the model's output

Saliency Maps: show which words or tokens have the highest impact on the model's decision, aiding in interpretability.

Rule-based Explanations: creates human-readable rules that describe how the model makes predictions based on specific linguistic patterns or features in the text.

Thank You..!