# Natural Language Processing

Unlocking the Power of Human Language !

# Introduction


Virtual assistants

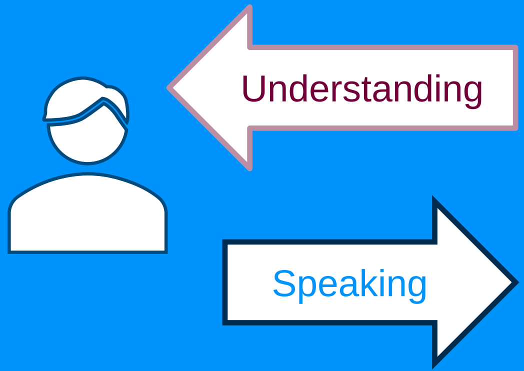

Chatbots

LLM Models

# ALL The Previous Are NLP Examples !

Let's Explore The NLP World

Imitating Human Behavior!

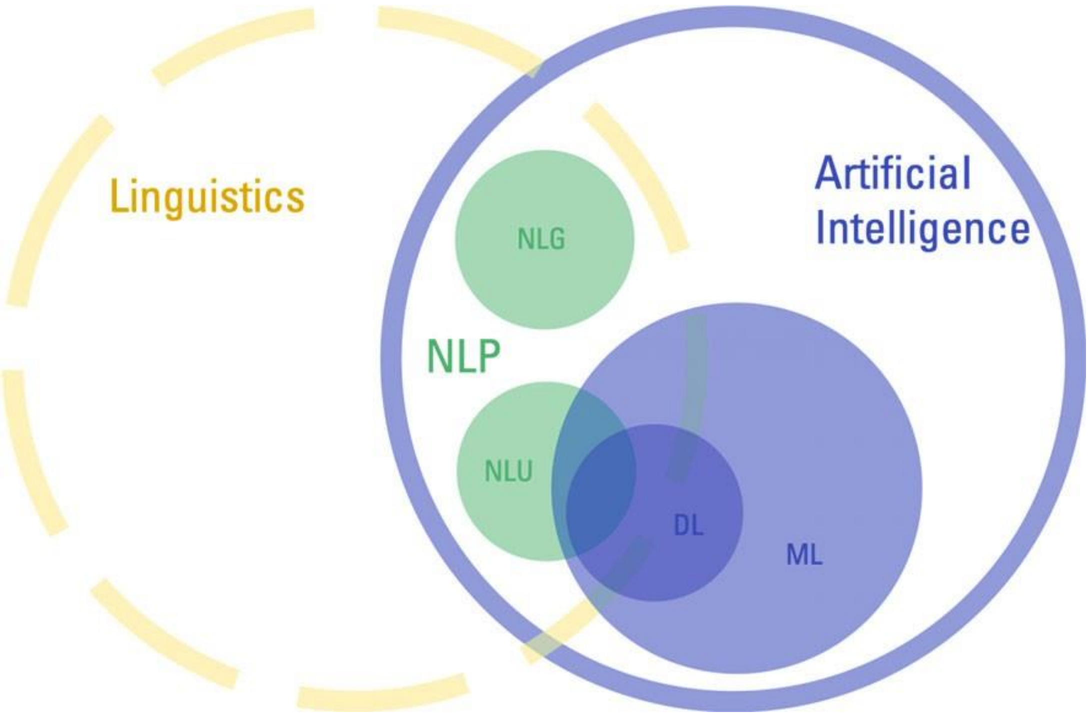



# What IS NLP?

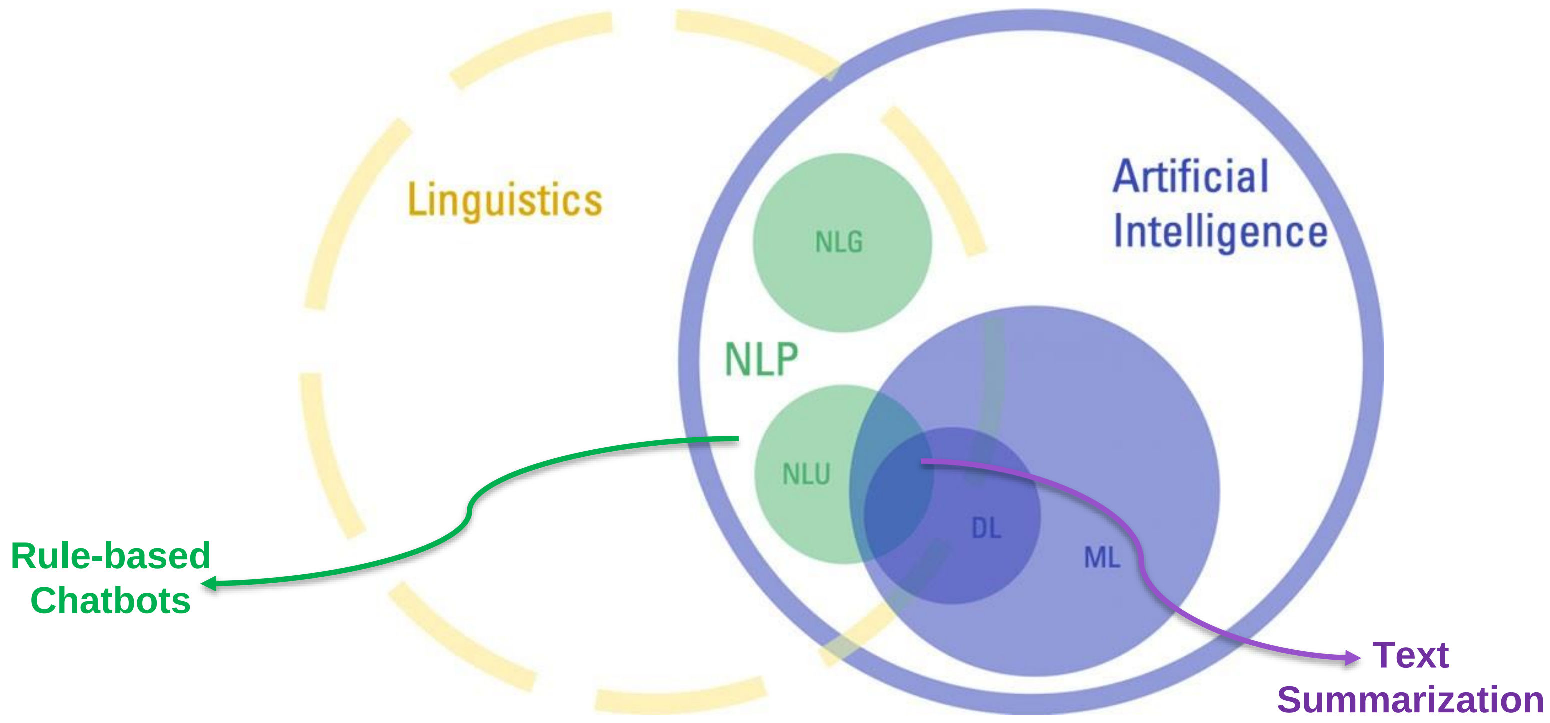
- A subfield of AI that deals with giving computers the ability to understand , process and generate human language.

- Human language can be text or voice.
- It is unstructured data

# Bird's-eye View



# Examples


Linguistics
Artificial Intelligence
NLG
NLP
NLU
DL
ML
Rule-based Chatbots
Text Summarization

# NLP Tasks

## NLP
Natural Language Processing

### NLU
Natural Language Understanding

### NLG
Natural Language Generating

# Natural Language Understanding (NLU)

- A subset of NLP, which uses syntactic and semantic analysis of text and speech to determine the meaning of a sentence.

- Syntax: The grammatical structure of a sentence.
- Semantics: Focuses on the meaning of words and sentences.

- NLU focuses on computer reading comprehension

# Natural Language Understanding (NLU)

For Example this sentence :

"Green ideas sleep furiously.“

- Syntax : Its correct as it follows the basic structure of subject-verb-object.

- Semantics: It doesn't make any sense because the meanings of the words don't create a logical concept.

# Natural Language Understanding (NLU)

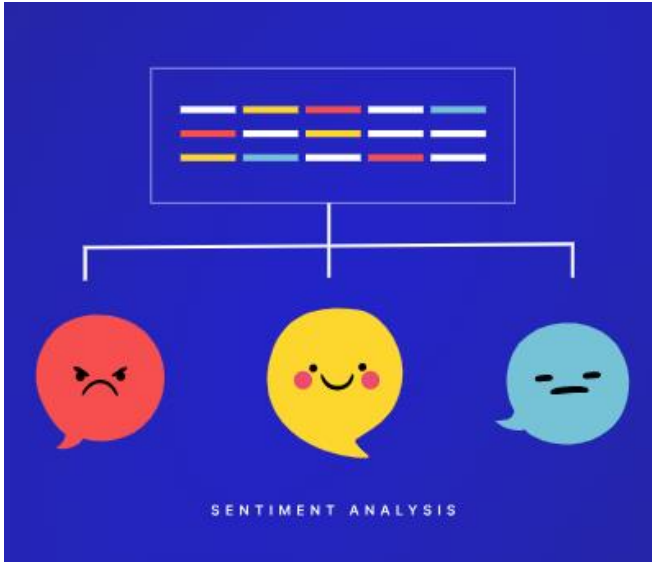
Another example :

1. Alice is swimming against the current.
2. The current version of the report is in the folder.

- Words can have multiple meanings .
- NLU make it possible to understand the intended meaning depending on the context .

# NLU Use Cases


Spam Detection

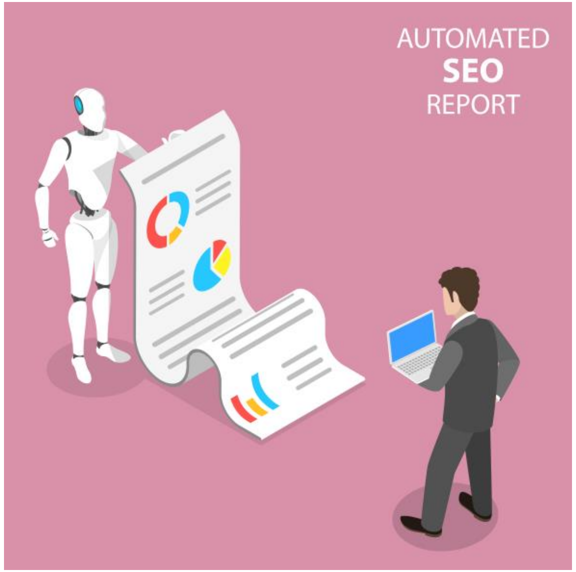



Sentiment Analysis

# Natural Language Generating (NLG)

- NLG is the ability of a computer program to generate human-like text.
- NLG systems take data as input, which can be structured or unstructured, and use it to create natural language outputs.
- NLU May be an initial step for NLG

# NLG Use Cases




**Automated Reporting**


**Text Translation**

# NLP Lifecycle

Problem Definition and Data Collection → Data Preprocessing and Feature Engineering → Model Selection and Training → Model Evaluation and Tuning → Deployment and Monitoring

**Problems :**
Sentiment Analysis ,
Text Classification,
Topic Clustering ...etc.
**Data :**
Text / Voice

Data Cleaning
Tokenization
Stemming
Lemmatization
Spell Correction

RNNs
BERT
GPT
ALBERT

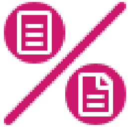
# Common NLP Problems

Machine Translation

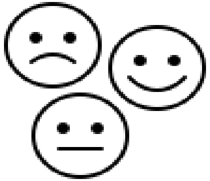
Speech Recognition

Text Summarization

Question Answering



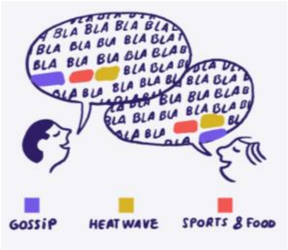
Text Classification



Sentiment Analysis

INFORMATION RETRIEVAL

Text Similarity

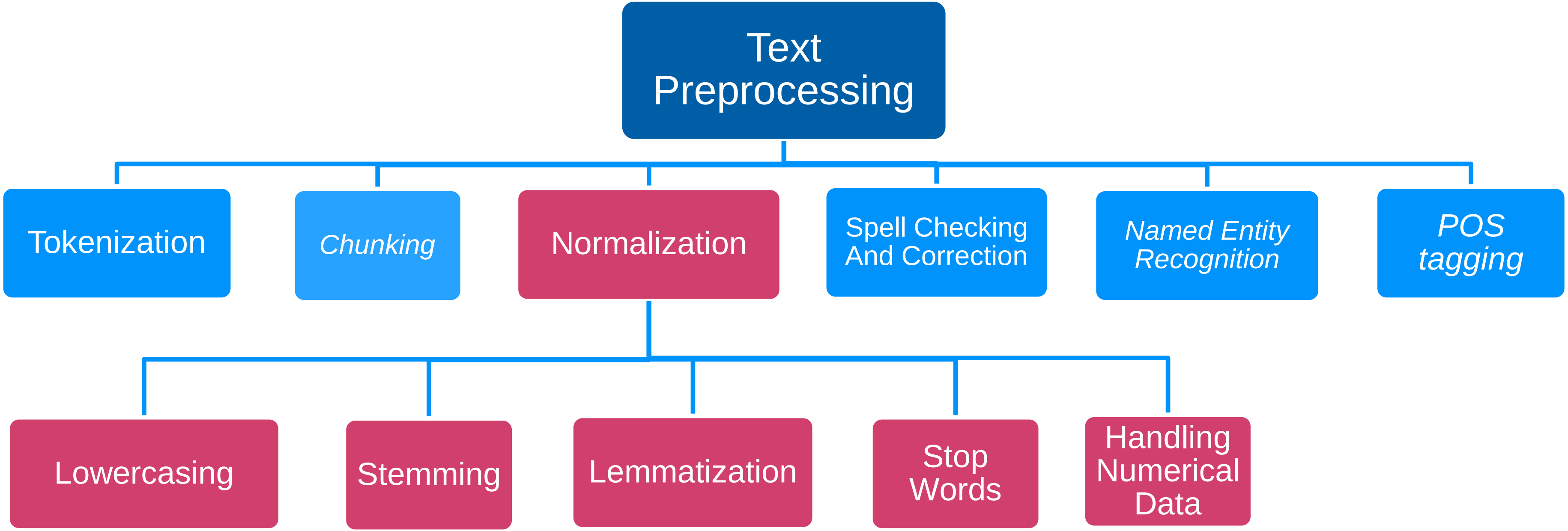



Topic Modeling

Chatbots and Virtual Assistants

# Text Preprocessing

# Text Preprocessing Steps



# Tokenization

- The process of breaking down a text into individual units called **tokens.**
- A token is a word, a phrase, a character or other meaningful elements.


[“We”, “love”, “NLP” ,”!”]

# Tokenization

## The primary goal of tokenization:

- To represent text in a manner that's meaningful for machines without losing its context.

### Types of Tokenization

**Sentence Tokenization**

**"I love natural language processing. It's fascinating!“**
["I love natural language processing." ,"It's fascinating!"]

**Word Tokenization**

**"Natural language processing”**
["Natural", "language", "processing”]

**WordPiece Tokenization**

**"unhappiness“**
["un", "happiness"]

**Subword Tokenization**

**"unhappiness“**
["un", "hap", "pi", "ness"]

# Tokenization

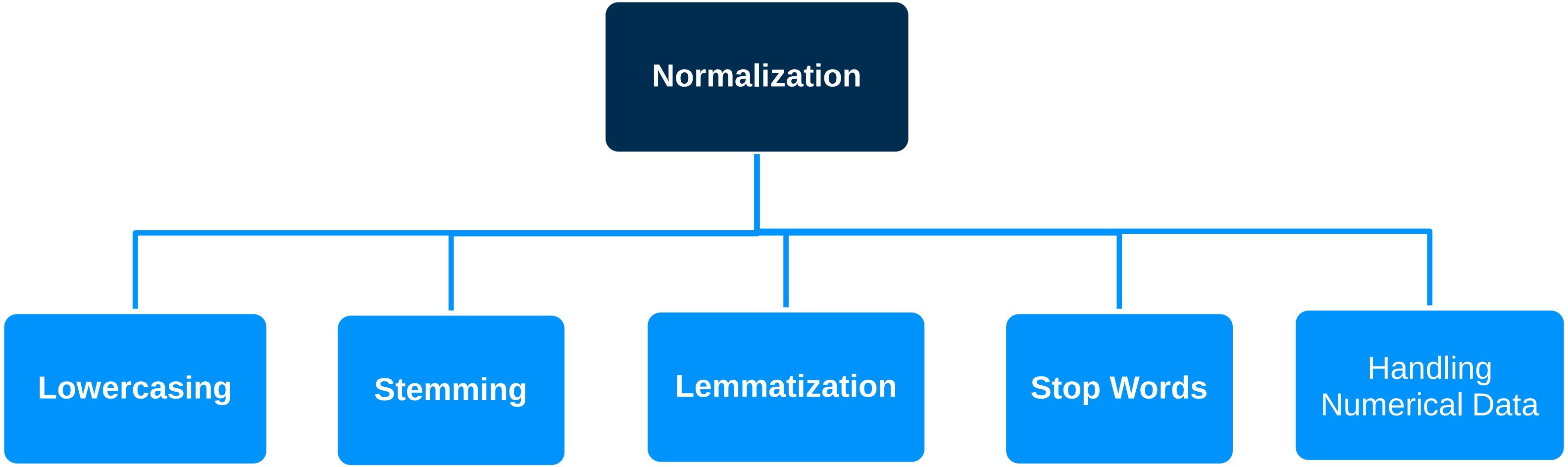
Challenges :

- Dealing with punctuation that could be part of a token or separate tokens. ( “U.K.”)
- Handling exceptional cases: like dates, amounts, and abbreviations.
- Splitting compound words. Breakfast
- Adjusting to the context, the same sequence of characters might need to be tokenized differently depending on its use.

# Normalization

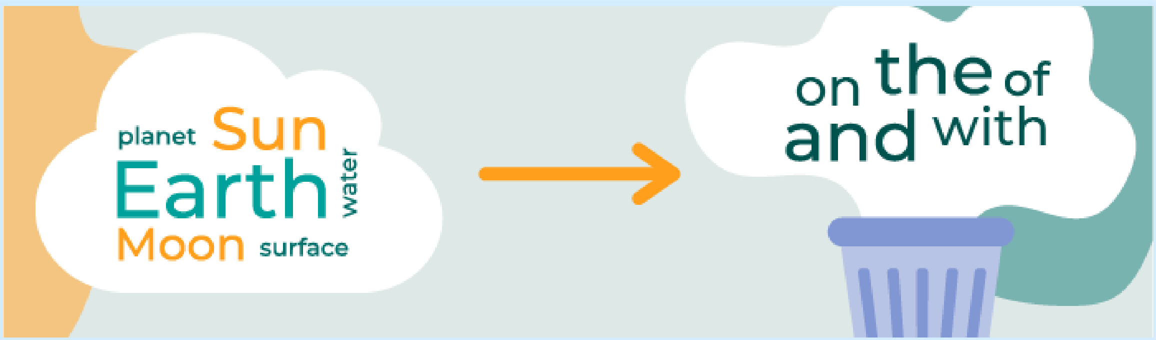
The process of transforming text into a standard, consistent format to enhance its analysis, understanding, and processing.

- It helps reduce variability, noise, and redundancy in the text corpus.



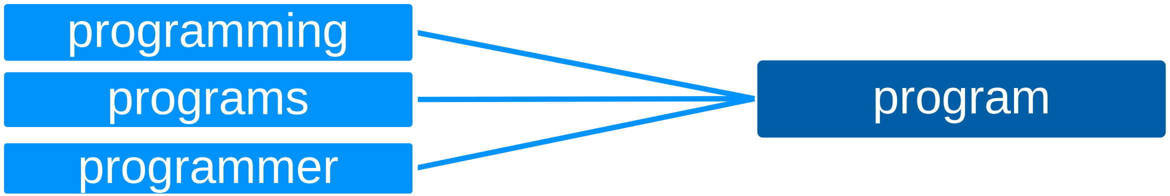
# Stop Words

- Stop words are commonly occurring words in a language that do not carry significant meaning or contribute to the understanding of the text.
- They are typically **removed** to improve the **efficiency** and **accuracy**.



# Stemming

- Stemming is a technique used to simplifies a word to its stem (base or root) by removing the suffixes or prefixes from words




- It helps in enhancing text analysis and language understanding.

- **Importance of stemming :**
  - Normalization
  - Vocabulary Reduction
  - Improving Information Retrieval

# Stemming

## Challenges:

- **Under-stemming**

  **can’t get the root of did as do**

- **Over-stemming**

  **universe → univers**

- **Language challenges**

  For example, an Italian stemmer is more complicated than an English stemmer because there is a higher number of verb inflections.

# Lemmatization

- A technique used to reduce inflected words to their root word by identifying an inflected word's **"lemma"** (dictionary form) based on its intended meaning and context.

- It ensures that the resulting lemma is a valid word in the language.

**Programming** → **Program**

**Better** → **Good**

**Mice** → **Mouse**

# Lemmatization

## Characteristics of Lemmatization

### Accuracy:

- Unlike stemming ,it does not merely cut words off.
- Analysis of words is conducted based on the word's POS (Part-of-Speech) to take context into consideration.
- It leads to real dictionary words being produced.

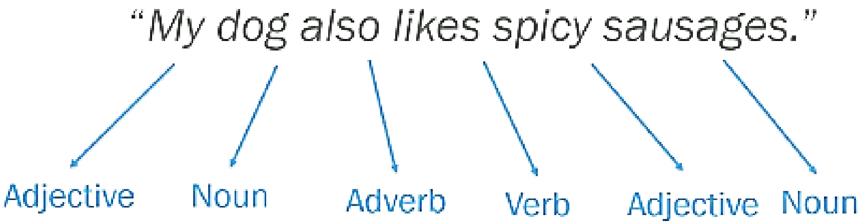
### Time-consuming:

- Compared to stemming, lemmatization is a slow .
- This is because lemmatization involves performing morphological analysis and deriving the meaning of words from a dictionary.

# Part-of-Speech Tagging

- An NLP task where the goal is to assign a grammatical category (such as noun, verb, adjective, etc.) to each word in a given text.

- This provides a deeper understanding of the structure and function of each word in a sentence.



# Text Representation

# Text Representation (Vectorization)

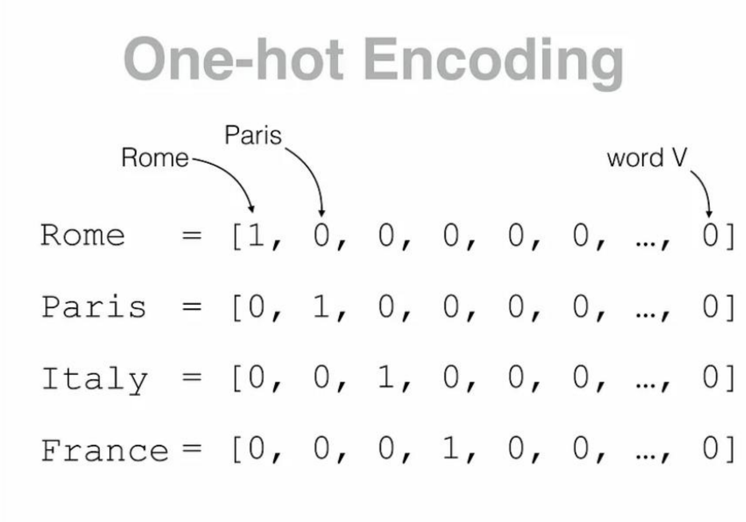
- A preprocessing step that aims to converting raw text data into a format that computers can understand and work with.

**Common techniques for text representation:**

| | | | |
|---|---|---|---|
| Bag-of-Words | TF-IDF | N-grams | Word Embeddings |

# One-hot Encoding

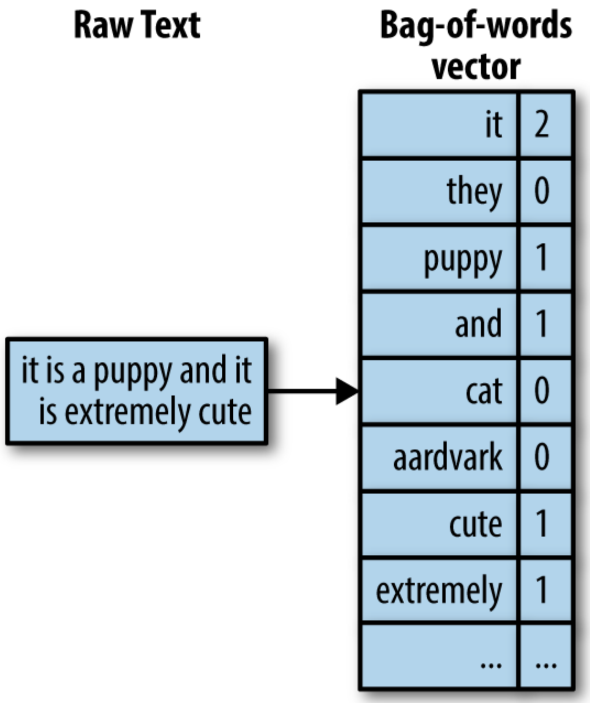
This does not provide much information about word meaning and it does not reveal any existing relationship between words.



# Bag-of-Words (BoW)

# Bag-of-Words (BoW)

- The Bag-of-Words (BoW) model is a simplified approach to representing text data.
- It simplifies the text by representing it as a numerical vector. It focuses on word frequency rather than word order
- The complexity of BoW depends on the tokenization level.



# BoW Sentence Similarity Example

« The dog started to run after the cat, but the cat jumped over the fence. »
« The cat's food was eaten by the dog in a few seconds. »
« The cat attacked the bird the other day. »

↓

```
[ « dog », « start », « run », « cat », « cat », « jump », « fence » ]
[ « cat », « food », « eat », « dog », « second » ]
[« cat », « attack », « bird », « day » ]
```

Lowercasing
Stemming
Lemmatization
Stop Words

↓

Bag-Of-Words representation

| | Dog | Start | Run | Cat | Jump | Fence | Food | Eat | Second | attack | Bird | Day |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | **2** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

# How does BOW work?

1. It uses **One-hot encoding** method to generate a feature vector for each unique word

| | satisf ied | prod uct | amaz ing | happ y | best | bad | servi ce | unha ppy | not |
|---|---|---|---|---|---|---|---|---|---|
| satisf ied | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| prod uct | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| amaz ing | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| happ y | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| best | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| bad | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| servi ce | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| unha ppy | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| not | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

1. Then generate a **sentence feature vector** by summing the words vectors that this sentence include

| satisf ied | prod uct | amaz ing | happ y | best | bad | servi ce | unha ppy | not |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 2 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 2 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

# TF-IDF

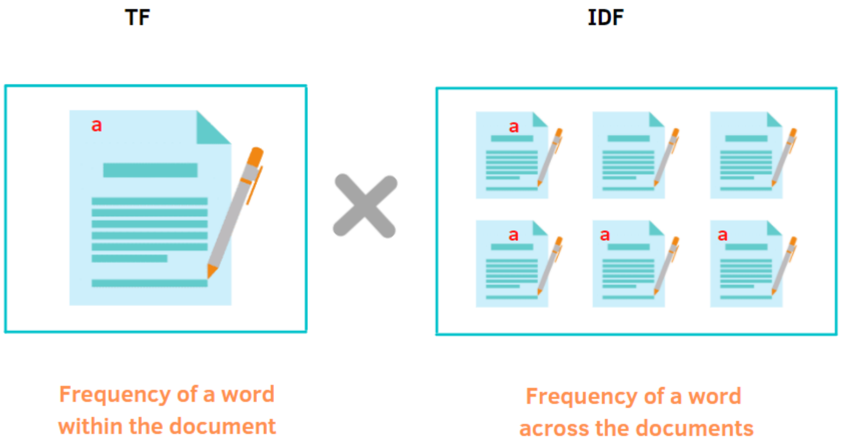
# Term Frequency-Inverse Document Frequency (TF-IDF)

- A statistical measure that evaluates how relevant a word is to a document in a collection of documents.

- Term Frequency (TF): is a measure of how frequently a term appears in a document

- Inverse Document Frequency (IDF) : is a measure of how important a term is.

# Term Frequency-Inverse Document Frequency (TF-IDF)



# Term Frequency-Inverse Document Frequency (TF-IDF)

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

**TF-IDF**

Term $x$ within document $y$

$tf_{x,y}$ = frequency of $x$ in $y$

$df_x$ = number of documents containing $x$

$N$ = total number of documents

# BoW Vs TF-IDF

## BOW

- Assigns equal importance to all words.

## TF-IDF

- Gives higher weight to rare words

## Both

- Both methods ignore word order and context, resulting in a loss of semantic information.

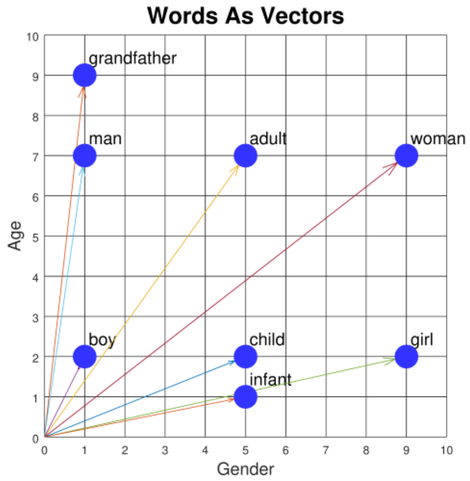
# Word Embedding

# Word Embedding

- A type of word representation where every term in the corpus is represented by **a numerical vector.**
- It allows words with **similar meanings** to have a **similar representation**.

**Word Coordinates**

| | Gender | Age |
|---|---|---|
| grandfather | [ 1, | 9 ] |
| man | [ 1, | 7 ] |
| adult | [ 5, | 7 ] |
| woman | [ 9, | 7 ] |
| boy | [ 1, | 2 ] |
| child | [ 5, | 2 ] |
| girl | [ 9, | 2 ] |
| infant | [ 5, | 1 ] |

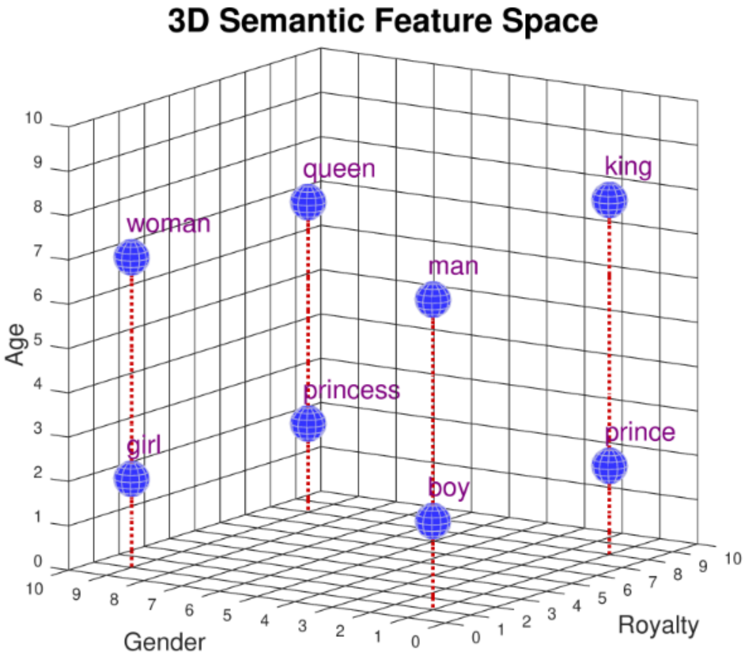


# Word Embedding

3 Dimensions vector representation

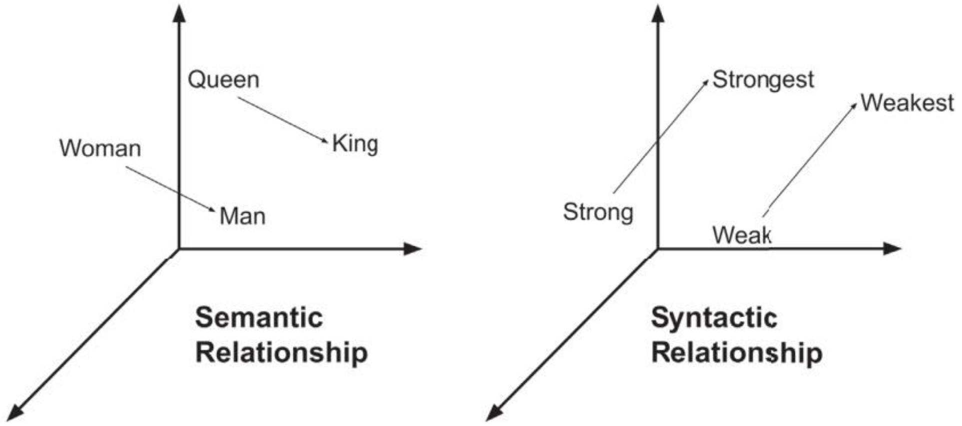
**Word Coordinates**

| | Gender | Age | Royalty |
|---|---|---|---|
| man | [ 1, | 7, | 1 ] |
| woman | [ 9, | 7, | 1 ] |
| boy | [ 1, | 2, | 1 ] |
| girl | [ 9, | 2, | 1 ] |
| king | [ 1, | 8, | 8 ] |
| queen | [ 9, | 7, | 8 ] |
| prince | [ 1, | 2, | 8 ] |
| princess | [ 9, | 2, | 8 ] |



# Word Embedding

word embeddings can capture both semantic and syntactic relationships between words.




Man – King + Women -> A vector that's similar to queen

# Why Do We Use Word Embedding Instead of TF-IDF and BOW?

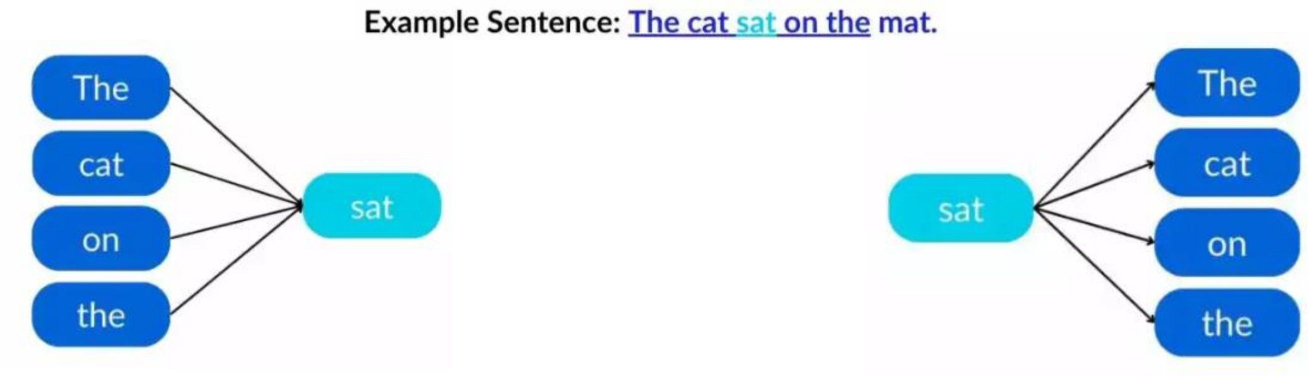
- **Loss of semantic information:**
  - Both BOW and TF-IDF focus on frequency and ignore word order and context .
- **Eliminating the sparse representation:**
  - The resulting vectors in BOW and TF-IDF techniques are sparse( have a lot of 0’s) ,which is bad to memory and algorithm.

# Word Embedding

Word Embedding Approaches

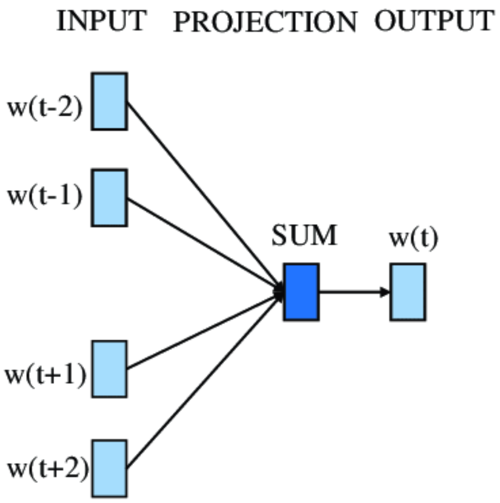
**CBOW**
**Continuous Bag-of-Words**

**Skip-grams**



# CBOW

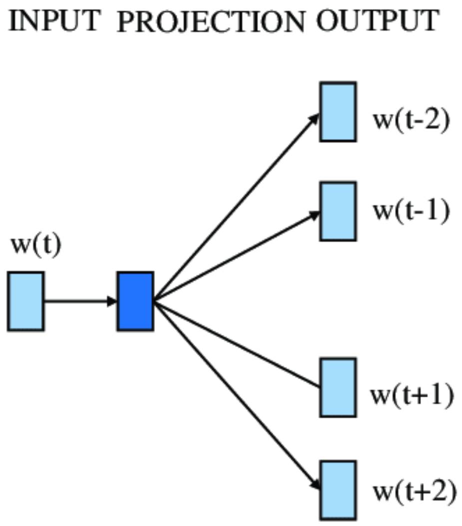
- In CBOW approach, the neural network model will try to predict the **center word** given **context words**.




(a) CBOW

# Skip-gram

- It aims to predict surrounding context words based on a given target word.




(b) Skip-gram

# CBOW Vs Skip-grams

| | CBOW | Skip-grams |
|---|---|---|
| **Convergence Time** | Hours | Days |
| **Learning Relationships Between Words** | Better at **syntactic relationships** | Better at understanding the **semantic relationships**. |
| **Sensitivity to Overfitting Frequent Words** | High | Less |
| **Amount of Documents Required** | More | Less |

# Thanks

Tamara Abu-hawele

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik** and illustrations by **Stories**

Please keep this slide for attribution