

# Natural Language Processing (NLP)

## **Linguistic Basics**

1. Lemmatization
2. Stemming
3. Part of Speech Tagging

## **Text Identification/ Extraction**

4. Stop Words
5. Pattern Matching
6. Sentence Segmentation
7. Named Entity Recognition

## **Text Representation**

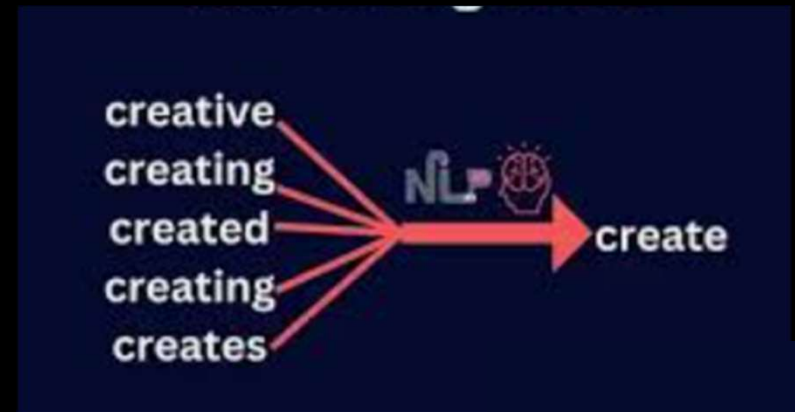
8. Tokenization
9. Word Embedding
10. Bag-of-words/TF-IDF

# NLP - Stemming

Obtain the base or root form of words by removing letters from the word's end

```
from nltk.stem.porter import *  
  
p_stemmer = PorterStemmer()  
words = ["runner", "running", "ran"]  
for word in words:  
    print(word+' --> '+p_stemmer.stem(word))
```

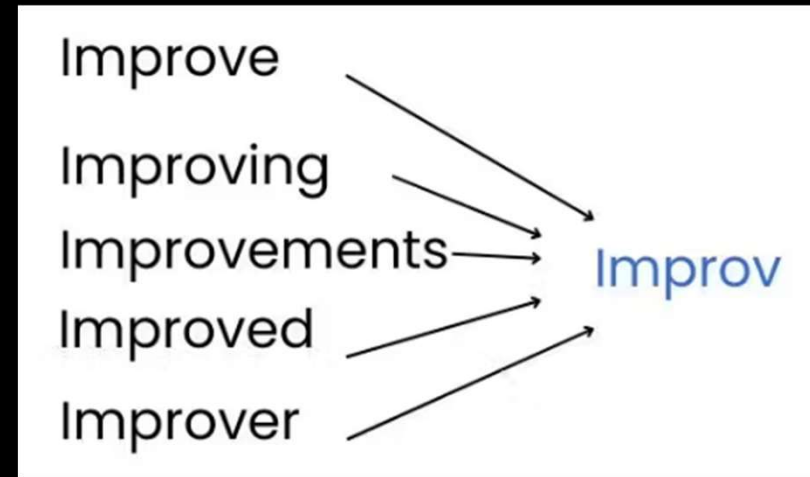
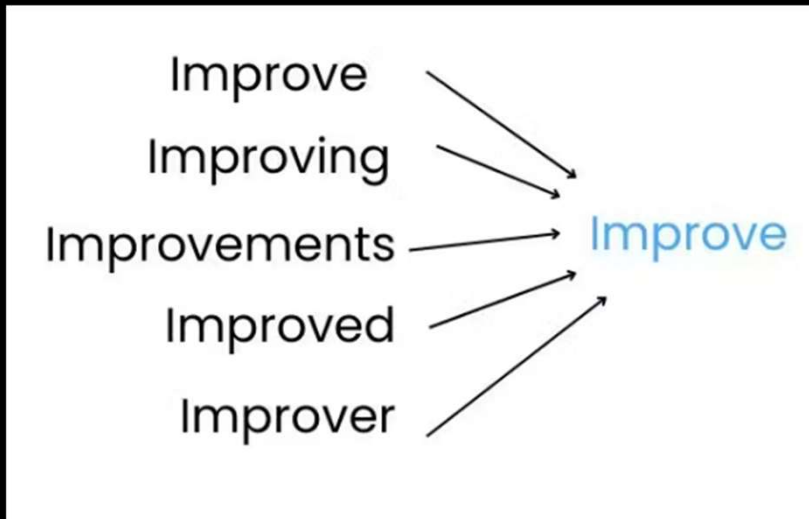
```
runner --> runner  
running --> run  
ran --> ran
```



# NLP - Lemmatization

lemmatization is a more sophisticated linguistic process that aims to reduce a word to its base or dictionary form

Lemmatization takes into account factors such as part-of-speech (POS) tags and contextual understanding to ensure accurate and meaningful transformations.



Vs Stemming

# Parts Of Speech (POS) Tagging

Part of speech refers to the grammatical category of a word in a sentence, such as noun, verb, adjective, adverb, pronoun, preposition, conjunction, or interjection.

Used to identify named entities and speech recognition

## Stop words

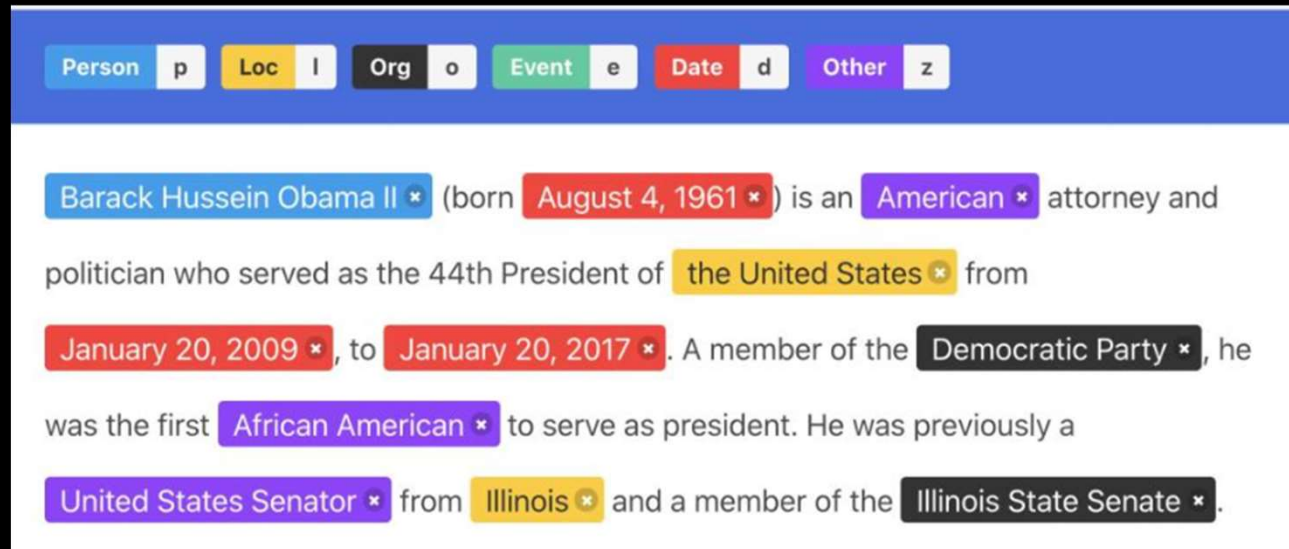
Common words such as “a” and “the” occur so frequently in text that they often do not carry significant meaning compared to nouns, verbs, and modifiers.

Spacy provides a built-in list of approximately 305 English stop words that can be readily utilized.

# Named Entity Recognition (NER)

Named Entity Recognition (NER) entails the identification and classification of named entities present in the given text.

Named entities represent recognizable entities : individuals, organizations, locations, dates, numerical expressions, and others.



The screenshot displays a web-based NER tool interface. At the top, a legend bar identifies entity types with colored boxes and single-letter codes: Person (p, blue), Loc (l, yellow), Org (o, black), Event (e, green), Date (d, red), and Other (z, purple). Below the legend, a text snippet about Barack Obama is shown with various phrases highlighted in colored boxes corresponding to the legend. Each highlighted phrase is followed by a small 'x' icon. The entities identified are: 'Barack Hussein Obama II' (Person, blue), 'August 4, 1961' (Date, red), 'American' (Other, purple), 'the United States' (Loc, yellow), 'January 20, 2009' (Date, red), 'January 20, 2017' (Date, red), 'Democratic Party' (Org, black), 'African American' (Other, purple), 'United States Senator' (Other, purple), 'Illinois' (Loc, yellow), and 'Illinois State Senate' (Org, black).

Person p Loc l Org o Event e Date d Other z

Barack Hussein Obama II x (born August 4, 1961 x) is an American x attorney and politician who served as the 44th President of the United States x from January 20, 2009 x, to January 20, 2017 x. A member of the Democratic Party x, he was the first African American x to serve as president. He was previously a United States Senator x from Illinois x and a member of the Illinois State Senate x.

<https://demos.explosion.ai/displacy-ent>

# Text representation - tokenization

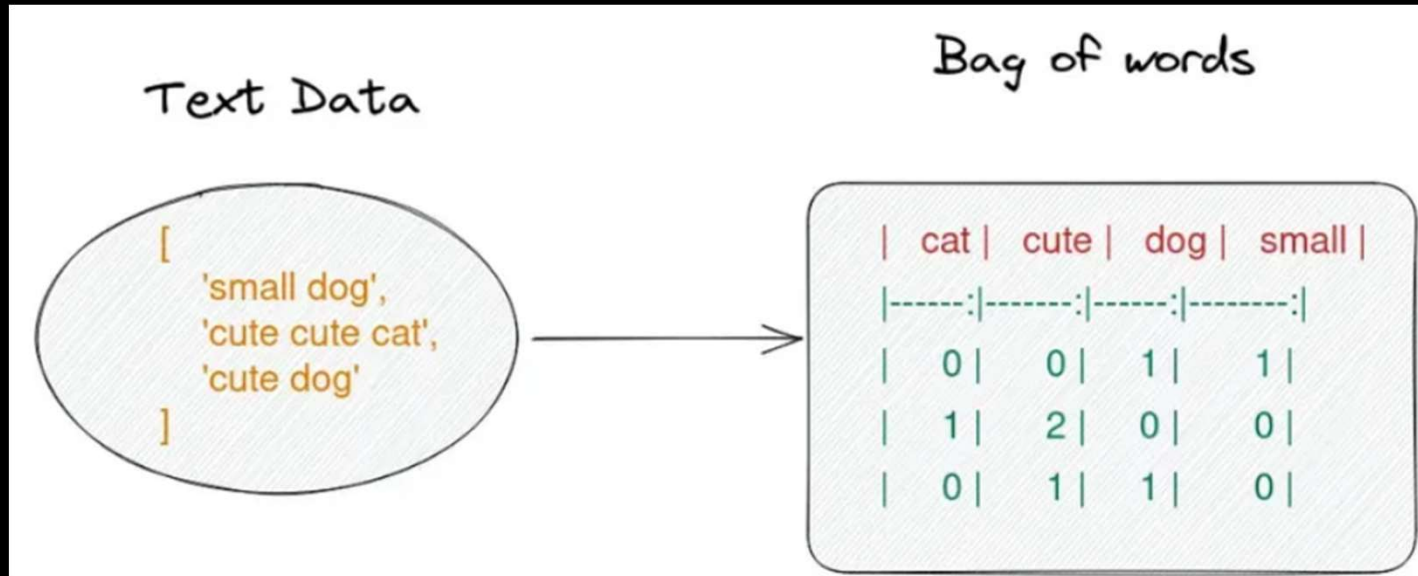
Tokenization involves breaking down the original text into smaller components known as tokens. These tokens can be created based on contiguous sequences of characters or words.

N-gram refers to a consecutive sequence of  $n$  items, where an item can be a character or a word.

<https://platform.openai.com/tokenizer>

Determines the vocabulary of the model

# Text representation – Bag of Words



Converts each word into a vector based on the frequency of the text, without considering order

# Text representation – TF-IDF

Term Frequency (TF) / Inverse Document Frequency (IDF)

## Term Frequency (TF)

This measures how frequently a term (word) appears in a document. It assigns a higher weight to words that occur more frequently within the document.

## Inverse Document Frequency (IDF)

This part measures how unique or rare a term is across all documents. It assigns a higher weight to words that appear less frequently across the corpus but provide more unique or informative content within a document.



# Text representation – TF-IDF

Term Frequency (TF) / Inverse Document Frequency (IDF)

Corpus D		
$+1 \rightarrow$	$d_1$ A quick brown fox jumps over the lazy dog. What a fox!	$TF-IDF = 0.17 \times 0 = 0$ (“fox”, $d_1$ , D)
$+1 \rightarrow$	$d_2$ A quick brown fox jumps over the lazy fox. What a fox!	$TF-IDF = 0.25 \times 0 = 0$ (“fox”, $d_2$ , D)

Answer: Using TF-IDF, the word “fox” is equally relevant for both document  $d_1$  and document  $d_2$

Question: How word fox is relevant to corpus D documents?

Solution:

**TF-IDF**

TF is the frequency of any “term” in a given “document”.

IDF is constant per corpus, and accounts for the ratio of documents that include that specific “term”.

$$TF(\text{“fox”}, d_1) = 2 / 12 = 0.17$$

$$TF(\text{“fox”}, d_2) = 3 / 12 = 0.25$$

$$IDF(\text{“fox”}, D) = \log(2/2) = 0$$

# Limitations of BoW and TF/IDF

BoW and TF-IDF are traditional text representation techniques, they have major limitations:

## **No Semantic Meaning, cannot capture similarity**

Represent text as sparse vectors where each word is a separate dimension.

For example, "king" and "queen" have no relationship in BoW/TF-IDF.

## **High Dimensionality (Sparse Representation)**

If a vocabulary has 100,000 words, each document is a 100,000-dimensional vector, making computations inefficient.

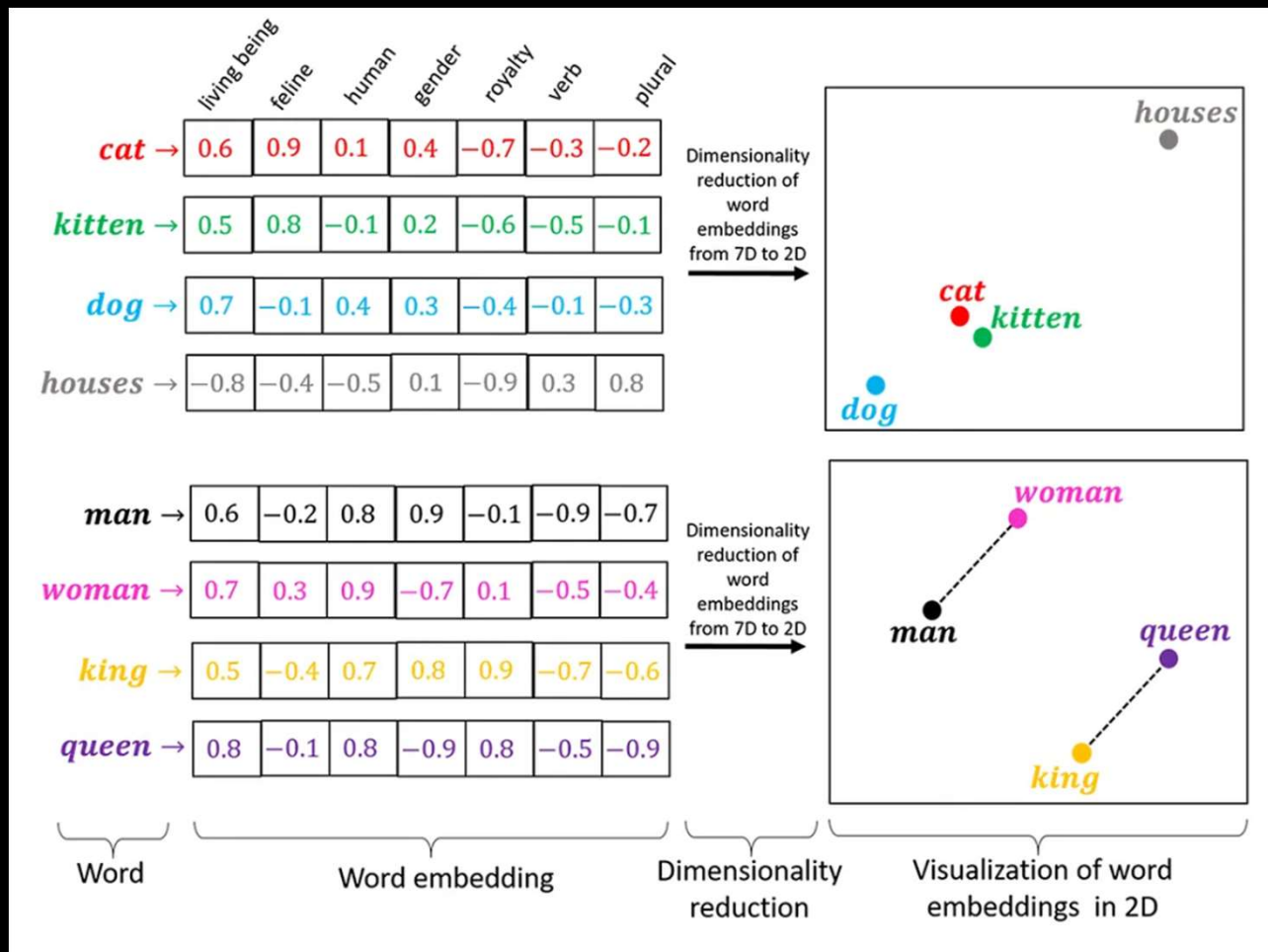
## **No Context Awareness**

BoW & TF-IDF treat all occurrences of a word the same, regardless of sentence structure.

## **No Word Order Information**

BoW and TF-IDF ignore word order, so "I love cats" and "Cats love me" have the same representation.

# Vectors and Embeddings



# Video: Vectors and Embeddings

# Word2vec (Google 2013)

Solves many of the issues with previous approaches

Word2Vec is a

- Neural network-based model for learning word embeddings
- Introduces word embeddings: dense vector representations of words that capture their semantic meaning based on the context they appear in.
- It transforms words into high-dimensional numerical vectors, allowing words with similar meanings to have similar vector representations.

*A major breakthrough in NLP*

# How Word2vec (Google 2013) solves these issues?

## **Dense Vector Representations**

Represents words as low-dimensional dense vectors  
(e.g., 300 dimensions instead of 100,000).

## **Captures Semantic Meaning**

Similar words have similar vector representations:

king - man + woman  $\approx$  queen

## **Learns Context from Word Usage**

Uses CBOW (Continuous Bag of Words) and Skip-gram to learn relationships based on nearby words.

## **Handles Synonyms and Similarity**

Places similar words close in vector space, making it useful for NLP tasks.

*A major breakthrough in NLP*

# Training the word2vec model

CBOW (Continuous Bag Of Words)

Predicts center word from context

Skip-gram

Predicts context from the center word

# How Word2vec learns Symantec meaning and relationship?

## **Self supervised learning**

### **Skip-gram Model**

Given a word, it predicts surrounding words.

Example: From "The cat sits on the mat," the model learns that "cat" is often surrounded by words like "sits," "on," and "mat."

This helps capture the semantic meaning of "cat."

### **Continuous Bag of Words (CBOW) Model**

Given surrounding words, it predicts the missing word.

Example: Given "The \_\_\_\_ sits on the mat," the model predicts "cat."