Talent Bootcamp Series -1

Day-3

NLP Lab

Lab created by: Hasan Shahid Kamal

Core NLP Terminology

Corpus

A large, structured collection of text documents used for linguistic analysis and model training.

Token

The smallest meaningful unit of text, typically a word or sentence, created through tokenization.

Vocabulary

The complete set of unique tokens present in a corpus, forming the basis for text representation.

Document

A single text sample or instance within a corpus, such as an article, review, or message.

Text Pre-Processing

Clean Raw Text

Removes unwanted characters, formatting issues, and inconsistencies from unstructured text sources.

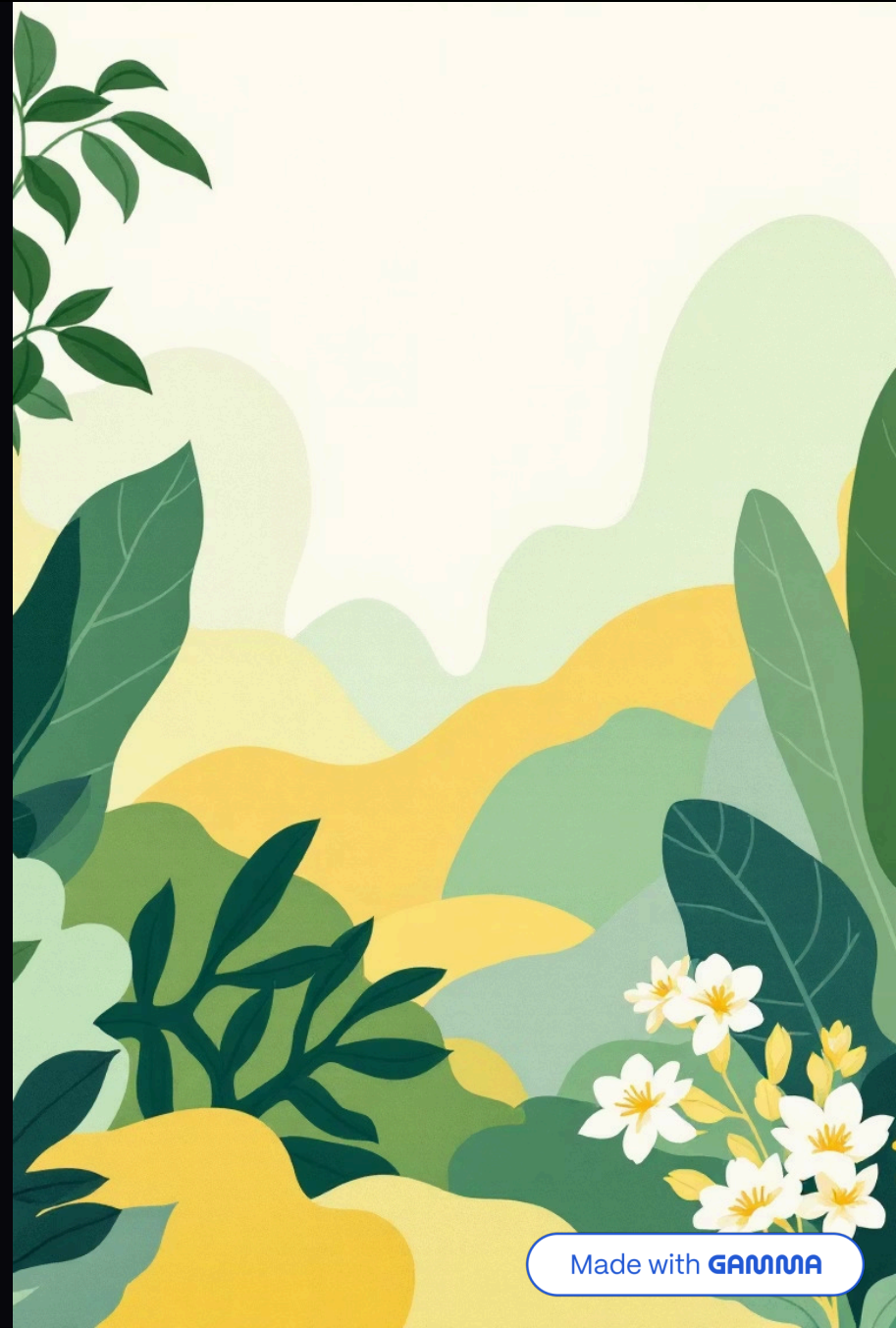
Reduce Noise

Eliminates irrelevant information that could interfere with accurate analysis and model performance.

Prepare for Analysis

Transforms text into a standardized format that algorithms can effectively process and understand.

PRE-PROCESSING IS AN ESSENTIAL PREREQUISITE STEP THAT DIRECTLY IMPACTS THE QUALITY AND ACCURACY OF ALL DOWNSTREAM NLP TASKS.



Tokenization

Breaking Text into Meaningful Units

Tokenization is the foundational process of splitting continuous text into discrete, analyzable units. This critical step transforms unstructured text into structured data that machines can process.

Word Tokenization

Splits text into individual words: "I love NLP" becomes ["I", "love", "NLP"]

Sentence Tokenization

Divides text into complete sentences for document-level analysis and context preservation.

['Natural Language Processing is interesting.', 'It helps computers understand human language.', 'NLTK makes NLP easy.']

What is punkt in NLTK?

> punkt is a **pre-trained tokenization model** in **NLTK** that is used to **split text into sentences and words.**

It is **not a function**, but a **data package (model)** that NLTK needs to perform tokenization correctly.

Why is punkt Needed?

Computers cannot automatically understand:

- Where sentences end
- Whether a dot (.) ends a sentence or is part of an abbreviation

punkt solves this by using **language-aware rules and statistics**.

Stop Words Removal

What Are Stop Words?

Common words that appear frequently but carry minimal semantic meaning for most NLP tasks. Removing them reduces dimensionality and computational cost.

Common examples include: the, is, at, and, to, in, of, for, with, on

Improved Efficiency

Reduces vocabulary size and speeds up processing without losing critical information.

Better Focus

Helps models concentrate on content-bearing words that drive meaning and classification.

TEXT NORMALIZATION

Stemming

Root-Based Approach

Reduces words to their base or root form by removing suffixes using rule-based algorithms.

Fast Processing

Computationally efficient and quick, making it ideal for large-scale text processing tasks.

Imperfect Results

May produce non-dictionary words (e.g., "running" → "run", "studies" → "studi") due to crude rules.

Example: running, runs, ran all become run

1️⃣ PorterStemmer

What it is

- A **class in NLTK**
- Implements the **stemming process**

What it does

- Cuts words using **rule-based suffix stripping**
- Does **NOT** check a dictionary

Lemmatization

1

Dictionary Lookup

Uses comprehensive vocabulary databases and morphological analysis to find proper base forms.

2

Grammar-Aware

Considers part of speech and grammatical context to determine the correct lemma.

3

Higher Accuracy

Produces valid dictionary words, ensuring better semantic preservation than stemming.

While slower than stemming, lemmatization produces linguistically correct results: "better" → "good", "am/are/is" → "be"

2️⃣ WordNetLemmatizer

What it is

- A **class in NLTK**
- Implements the **lemmatization process**

What it does

- Uses **WordNet (a dictionary)**
- Considers **word meaning and grammar**

Part-of-Speech Tagging

Understanding Grammatical Roles

Part-of-speech (POS) tagging assigns grammatical categories to each word, revealing the syntactic structure and relationships within sentences. This enables deeper linguistic analysis and meaning extraction.

Noun (NN)
Person, place, thing, or idea

Verb (VB)
Action or state of being

Adjective (JJ)
Describes or modifies nouns

Adverb (RB)
Modifies verbs or adjectives

Application: POS tagging helps with named entity recognition, information extraction, and understanding sentence structure for machine translation and question answering systems.

Sentiment Analysis

Sentiment Analysis is an NLP technique used to **identify the emotional tone or opinion** expressed in a piece of text.

- Could be Positive, Negative, Neutral

How NLTK Performs Sentiment Analysis

NLTK uses **VADER (Valence Aware Dictionary for sEntiment Reasoning)**

- Rule-based model
- Designed for short texts
- Works well with informal language

VADER combines three main ideas:

1️⃣ Sentiment Lexicon (Dictionary-Based)

VADER has a **pre-built dictionary** where words are assigned sentiment scores.

✔️ Positive words → positive score
✔️ Negative words → negative score

2️⃣ Rule-Based Heuristics

VADER does not just count words — it applies **linguistic rules**.

a) Capitalization

- **ALL CAPS** increases intensity

Example:

- *“This is GOOD”* → more positive than *“good”*

3️⃣ Sentiment Aggregation

After analyzing all words and rules, VADER:

- Combines scores
- Normalizes them
- Produces **four final scores**

Score	Meaning
pos	Strength of positive sentiment
neg	Strength of negative sentiment
neu	Neutral proportion
compound	Overall sentiment (−1 to +1)

Example for Sentiment Analysis

```
from nltk.sentiment import SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer() text = "I really enjoyed this lab session!"
scores = sia.polarity_scores(text)
print(scores)
```

```
Output:
{
 'neg': 0.0,
 'neu': 0.3,
 'pos': 0.7,
 'compound': 0.7269
}
```

Compound Score Interpretation

- **> 0.05** → Positive 😊
- **< −0.05** → Negative 😞
- **Between −0.05 and 0.05** → Neutral 😐

Named Entity Recognition (NER)

> Identifies real-world entities

- Examples: Person, Organization, Location
- Used in information extraction

Dependency Parsing

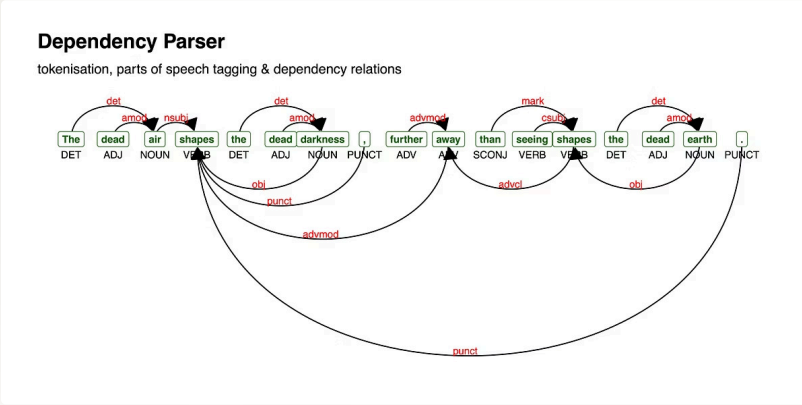
> Analyzes grammatical relationships

- Shows how words are connected
- Useful for syntactic analysis

👉 It answers:

- Who did what?
- Which word depends on which?

Each word (token) is connected to another word (its **head**).



Sentence:

> **“Students study NLP.”**

Dependency Relationships:

- **study** → main verb (ROOT)
- **Students** → subject of *study*
- **NLP** → object of *study*

Students → nsubj → study ← dobj ← NLP

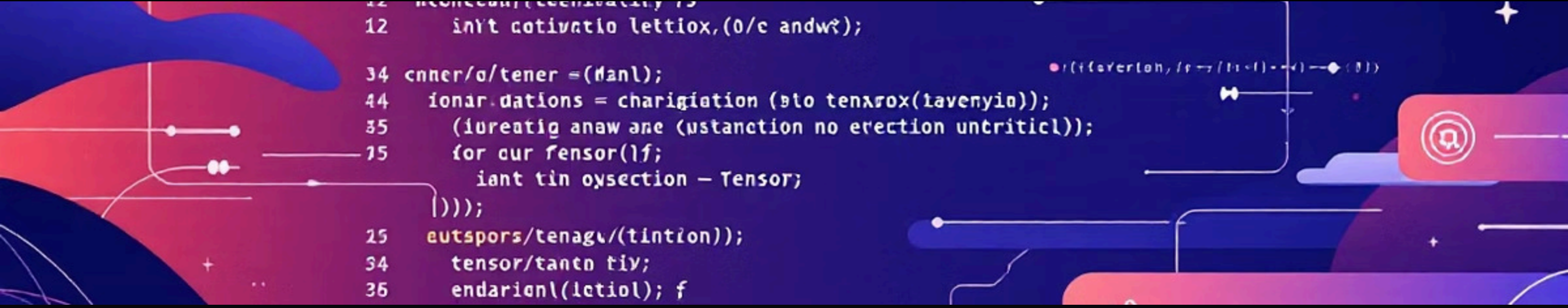
Text Similarity

> Measures semantic similarity between texts

- Used in plagiarism detection and search
- Based on vector similarity

spaCy Library

- Industrial-strength NLP library
- Fast and efficient processing
- NER, dependency parsing, similarity

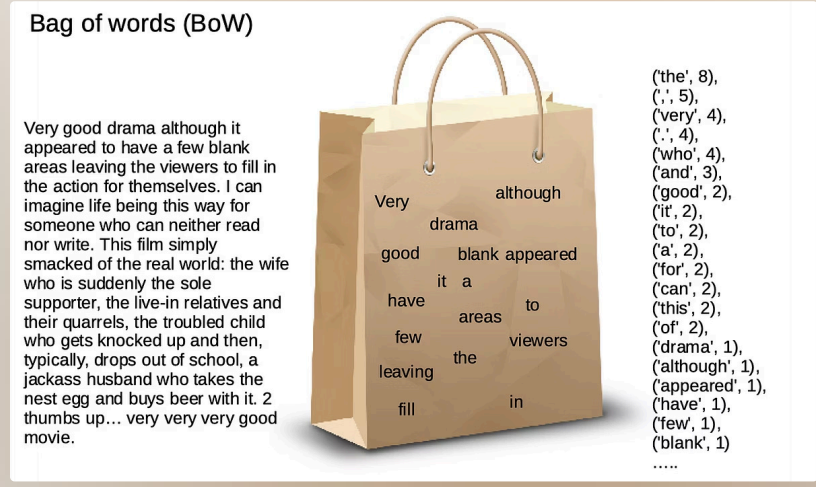


Text Vectorization

- Converts text into numerical form
- Required for ML models
- BoW, TF-IDF, Word Embeddings

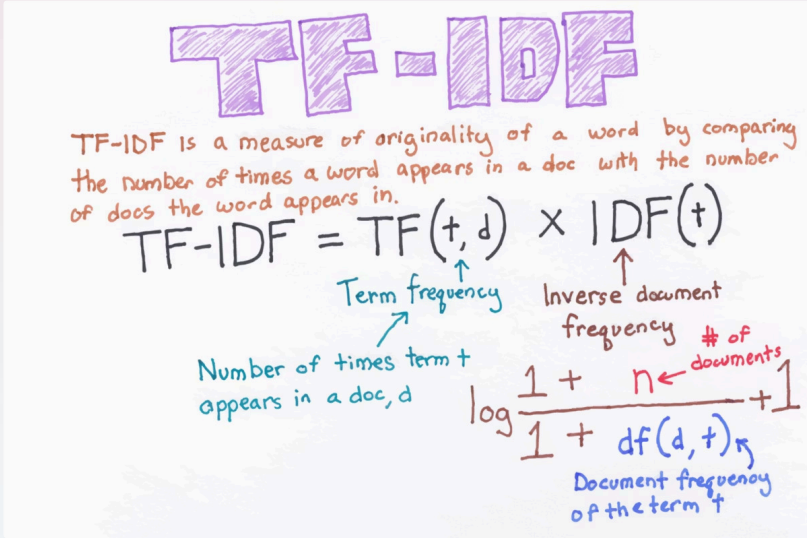
Bag of Words (BoW)

- Uses word frequency
- Ignores grammar and order
- Implemented using CountVectorizer



TF-IDF

- Measures word importance
- Reduces impact of common words
- Implemented using TfidfVectorizer



Text Classification

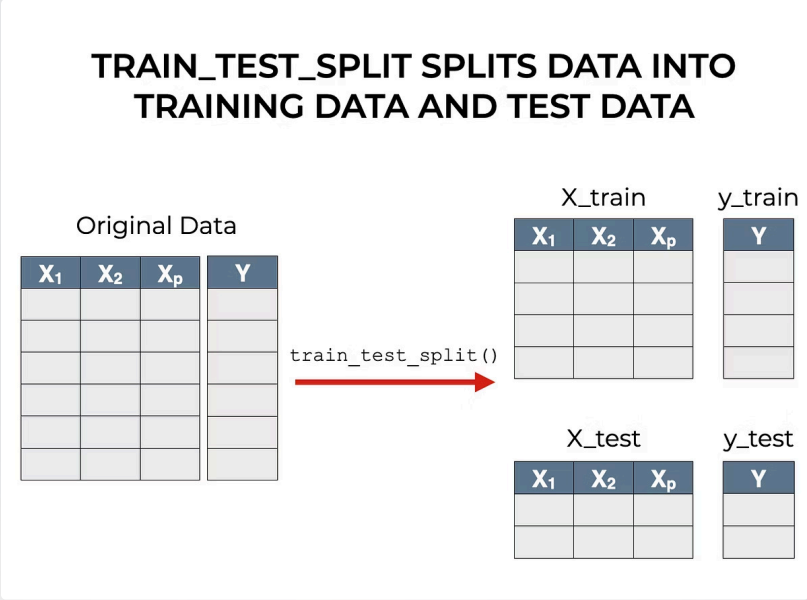
- Automatically categorizes text
- Spam detection, sentiment classification
- Uses ML algorithms

Model Evaluation Metrics

- Accuracy, Precision, Recall, F1-score
- Confusion Matrix for error analysis
- Ensures model reliability

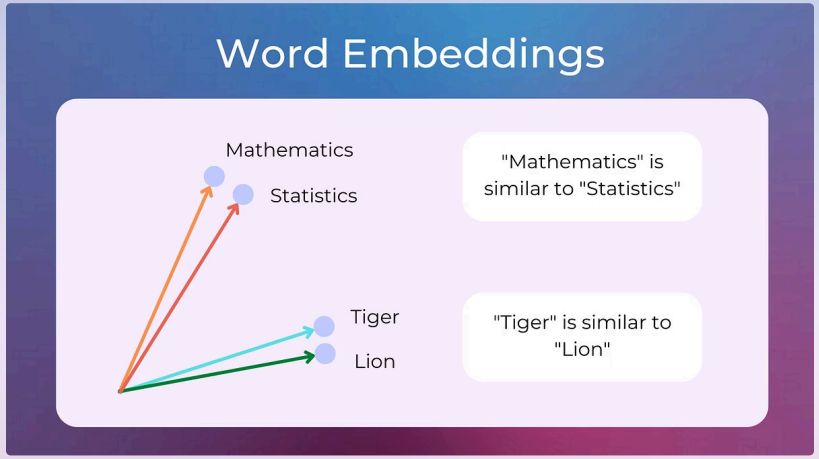
Scikit-Learn Tools

- CountVectorizer, TfidfVectorizer
- train_test_split, MultinomialNB
- classification_report, confusion_matrix



Word Embeddings

- Dense vector representation of words
- Captures semantic meaning
- Similar words have similar vectors

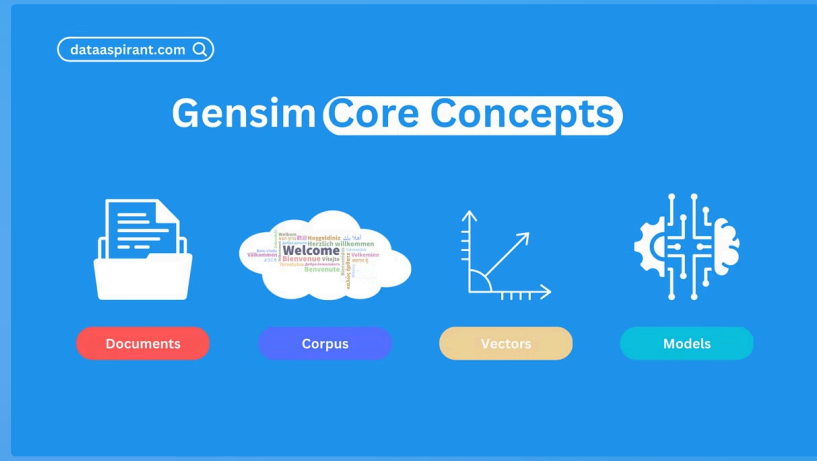


Word2Vec

- Neural network-based embeddings
- CBOW and Skip-gram models
- Implemented using Gensim

Gensim Functions

- Word2Vec()
- most_similar()
- similarity()



NLTK vs spaCy vs Gensim

Feature	**NLTK**	**spaCy**	**Gensim**
Main Role	Pre-processing & learning	Industrial NLP	Semantic modeling
Tokenization	✅	✅	❌
POS Tagging	✅	✅	❌
NER	❌	✅	❌
Word Embeddings	❌	Limited	✅ (Best)
Topic Modeling	❌	❌	✅
Speed	Medium	Very fast	Very fast
ML Focus	Low	Medium	High

Key Takeaways from the Lab

- NLP converts **raw human language** into a form machines can understand
- **Text pre-processing** (tokenization, stop words, stemming, lemmatization) is a **crucial first step**
- **NLTK** is best for learning and basic NLP tasks, while **spaCy** is used for fast, advanced linguistic analysis
- **VADER** enables quick sentiment analysis without training data
- **Vectorization techniques** (BoW, TF-IDF, Word2Vec) allow text to be used in machine learning models
- **Word embeddings** capture semantic meaning better than frequency-based methods
- Proper **feature extraction and evaluation** directly impact model performance