The Power of NLP

Transforming Text into Actionable Intelligence

Busra Ecem Sakar

02.10.2024

Agenda

1. Introduction to NLP
2. Natural Language Processing (NLP) pipeline
3. Overview of Common NLP Techniques
4. Practical Use Cases of NLP
5. References
6. Questions & Answers

1. Introduction to NLP

What is NLP?

- Natural Language Processing (NLP) is a branch of artificial intelligence (AI) that helps computers understand, interpret, and respond to human language.*

How Does NLP Work?

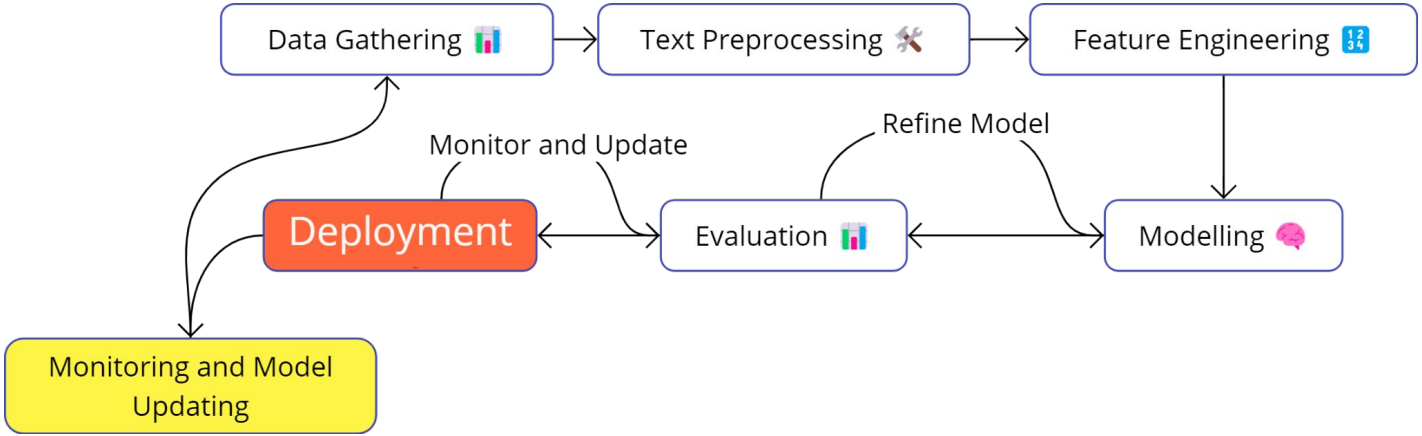
- By analyzing and processing vast amounts of unstructured text data,
- NLP can derive meaning, detect patterns, and generate insights.

Why is NLP Important?

- Automates Text Processing: Transforms large text datasets into meaningful insights.
- Enhances Decision-Making: Supports smarter business decisions by enhancing customer experiences, improving operational efficiency, and driving innovation across industries.

*: Oracle website link

2. Natural Language Processing (NLP) pipeline



There are a few **important points to remember**:

1. This pipeline is **not universal**—it serves as a general guide, but pipelines may vary depending on the specific NLP task.
2. This is primarily an **ML pipeline**. Deep learning pipelines are slightly different, especially in terms of **model training** and **feature engineering**
3. The **NLP pipeline is non-linear**. This means the stages don't always follow a strict sequence.

3. Overview of Common NLP Techniques

3.1 Tokenization

Purpose: Splitting text into smaller units (tokens), like words or sentences.
Use Case: Almost every NLP task requires tokenization.
Methods:

- **NLTK**: nltk.tokenize.word_tokenize()
- **spaCy**: nlp(text) (spaCy's built-in tokenizer)
- **Keras**: keras.preprocessing.text.text_to_word_sequence()

Example:

```
import nltk
from nltk.tokenize import word_tokenize

text = "Women Coding Community empowers women in their tech careers."
tokens = word_tokenize(text)
print("Tokenization:", tokens)
```

```
Tokenization: ['Women', 'Coding', 'Community', 'empowers', 'women', 'in', 'their', 'tech', 'careers', '.']
```

3.2 Stopword Removal

Purpose: Removing common words (like "the", "is", "and") that don’t add much meaning.
Use Case: Reduces noise, allowing models to focus on meaningful words.
Methods:

- **NLTK**: nltk.corpus.stopwords
- **spaCy**: token.is_stop
- **Gensim**: gensim.parsing.preprocessing.STOPWORDS

Example:

```
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
tokens = word_tokenize("Women Coding Community empowers women in their tech careers.")
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
print("Stopword Removal:", filtered_tokens)
```

```
Stopword Removal: ['Women', 'Coding', 'Community', 'empowers', 'women', 'tech', 'careers', '.']
```

3.3 Stemming and Lemmatization

Purpose: Reducing words to their root or base form.

- **Stemming**: Chops off suffixes (e.g., "running" → "run").
- **Lemmatization**: Converts words to their dictionary base form (e.g., "better" → "good", “went” → “go”).

Use Case: Helps treat different forms of a word as the same word.

Methods:

- nltk.stem.PorterStemmer() for stemming
- nltk.stem.WordNetLemmatizer() for lemmatization.

Example:

- **Stemming**: "running" → "run"
- **Lemmatization**: "better" → "good"

3.4 Named Entity Recognition (NER)

Purpose: Identifying and classifying named entities in text (people, organizations, places, dates).
Use Case: Extracting valuable information, especially useful in domains like legal, healthcare, and finance.

Example:

```
import spacy

nlp = spacy.load('en_core_web_sm')
text = "Apple is looking at buying a startup in San Francisco for $1 billion."
doc = nlp(text)
print("Named Entity Recognition:")
for ent in doc.ents:
    print(ent.text, ent.label_)
```

```
Named Entity Recognition:
Apple ORG
San Francisco GPE
$1 billion MONEY
```

3.5 Part-of-Speech (POS) Tagging:

Purpose: Assigning grammatical labels (like noun, verb, adjective) to each word in a sentence.
Use Case: Helps in understanding the structure of a sentence, useful for syntactic parsing and other grammar-based tasks.
Methods:

- spaCy use token.pos_ to perform POS tagging.
- NLTK apply `nltk.pos_tag()` after tokenizing the sentence.

Example:

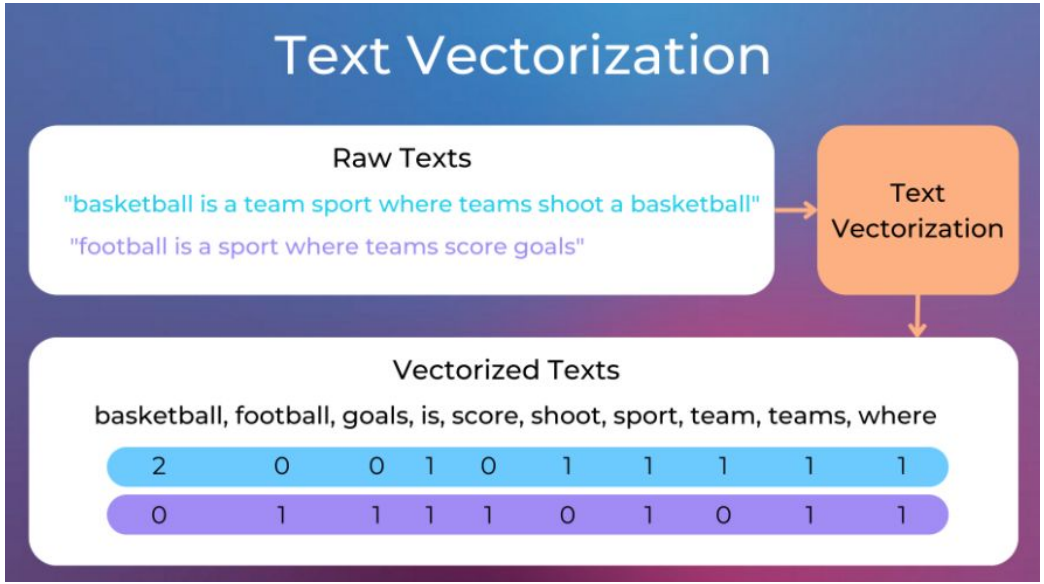
- **Input**: "WCC hosts workshops to help women learn coding and build networks."
- **Output**: [('WCC', 'NNP'), ('hosts', 'VBZ'), ('workshops', 'NNS'), ('to', 'TO'), ('help', 'VB'), ('women', 'NNS'), ('learn', 'VB'), ('coding', 'NN'), ('and', 'CC'), ('build', 'VB'), ('networks', 'NNS'), ('.', '.')]

3.6 Text Vectorization (TF-IDF, Bag of Words)

Purpose: Converting text into numerical features for machine learning models.

- ➔ **TF-IDF**: Measures how important a word is in a document relative to others.
- ➔ **Bag of Words**: Represents text by counting how often each word appears.

Use Case: Used in text classification, topic modeling, and sentiment analysis.



Source: NLP Planet

Bag of Words

Text Input:

- Document 1: "The cat sat on the mat."
- Document 2: "The dog sat on the log."

Bag of Words Representation:

Each word from the documents is assigned a column, and the frequency of the word in each document is represented:

	The	**cat**	**sat**	**on**	**mat**	**dog**	**log**
Document 1	2	1	1	1	1	0	0
Document 2	2	0	1	1	0	1	1

TF-IDF
(Term Frequency - Inverse Document Frequency)

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

- *t*, is the term (word),
- *d*, is the specific document,
- *D*, is the collection of documents (corpus).

TF (Term Frequency): How often a word appears in a document.

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

IDF (Inverse Document Frequency): How important a word is across all documents.

$$\text{IDF}(t, D) = \log\left(\frac{\text{Total number of documents } |D|}{\text{Number of documents containing term } t}\right)$$

Example of TF-IDF

Text Input:

- Document 1: "The cat sat on the mat."
- Document 2: "The dog sat on the log."

TF-IDF Representation:

After calculating the TF-IDF values, you will get something like this:

	the	**cat**	**sat**	**on**	**mat**	**dog**	**log**
Document 1	0	0.0502	0	0	0.0502	0	0
Document 2	0	0	0	0	0	0.0502	0.0502

The term **"cat"** appears **1 time**.
The total number of terms in Document 1 is **6** ("The", "cat", "sat", "on", "the", "mat").

$$TF_{\text{cat, Doc 1}} = \frac{1}{6} \approx 0.1667$$

The total number of documents is **2** (Document 1 and Document 2).
The term **"cat"** appears in only **1 document** (Document 1).

$$IDF_{\text{cat}} = \log\left(\frac{2}{1}\right) = \log(2) \approx 0.3010$$

TF-IDF value is calculated by multiplying the TF and IDF values:

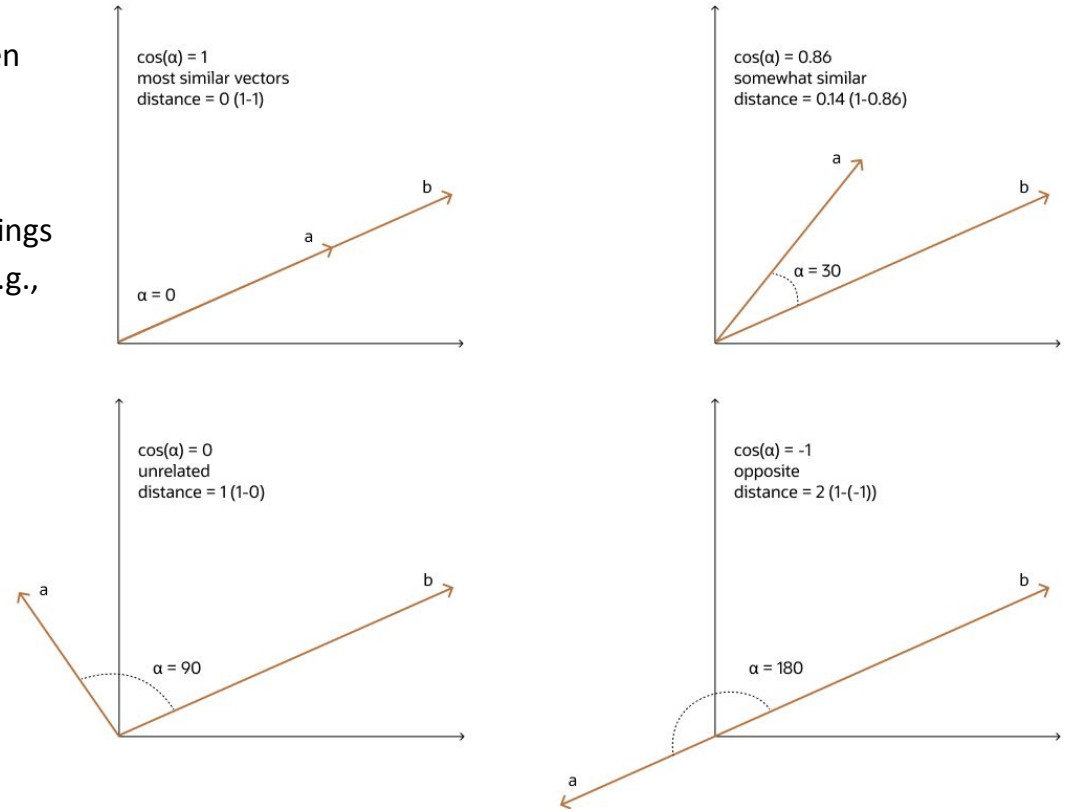
$$TF\text{-}IDF_{\text{cat, Doc 1}} = TF_{\text{cat, Doc 1}} \times IDF_{\text{cat}} = 0.1667 \times 0.3010 \approx 0.0502$$

Cosine Similarity

Cosine Similarity is a measure of similarity between two non-zero vectors of an inner product space.

In NLP, **Cosine Similarity** is commonly used to compare documents, sentences, or word embeddings by converting them into vector representations (e.g., using **TF-IDF**, **Word2Vec**, or **BERT** embeddings).

It helps find the similarity between texts based on their content.



Source: [Oracle]

3.7 Sentiment Analysis

Purpose: Determining whether the sentiment of a text is positive, negative, or neutral.
Use Case: Social media monitoring, product reviews, customer feedback analysis.
Methods:

- **VADER Sentiment Analysis** (NLTK): nltk.sentiment.vader.SentimentIntensityAnalyzer()
- **TextBlob**: TextBlob(text).sentiment
- **Hugging Face Transformers**: Pretrained sentiment models via pipeline("sentiment-analysis")

Example:

- **Input**: "I love this product!"
- **Output**: Positive sentiment

Hugging Face

Hugging Face is a leading company and platform in the field of **Natural Language Processing (NLP)** and **Machine Learning** that provides easy-to-use tools, models, and libraries to accelerate the development of AI applications.

It is widely known for its **Transformers library**, which allows users to access pre-trained models for a wide range of NLP tasks such as text classification, sentiment analysis, translation, text generation, and more.

Key Aspects of Hugging Face:

- **Transformers Library**: An open-source Python library offering state-of-the-art transformer models like **BERT**, **GPT**, **T5**, and more.
- **Model Hub**: A community-driven repository where users can find, share, and contribute thousands of pre-trained models for different languages and tasks.
- **Pipelines**: Simplifies the use of pre-trained models by providing an easy interface to perform common NLP tasks.
- **Community**: Hugging Face fosters an active community of researchers and developers who contribute models, datasets, and tools.

Hugging Face's mission is to democratize AI by making advanced machine learning models and tools accessible to everyone.

3.8 Text Classification

Purpose: Assigning predefined categories or labels to a text (e.g., spam detection, topic classification).
Use Case: Spam detection, topic classification, sentiment analysis.
Methods:

- **sklearn Naive Bayes, SVM**: sklearn.naive_bayes.MultinomialNB(), sklearn.svm.SVC()
- **Hugging Face Transformers**: Pretrained models for text classification via pipeline("text-classification")

Example:

- **Input**: "This email is about an upcoming conference."
- **Output**: Category → "Business"

3.9 Word Embeddings (Word2Vec, GloVe, BERT)

Purpose: Mapping words to dense vectors based on their meanings and context.
Use Case: Semantic analysis, text classification, similarity search.
Methods:

- **Word2Vec**: gensim.models.Word2Vec()
- **GloVe**: Pretrained GloVe embeddings from glove.6B.100d.txt
- **BERT**: Hugging Face Transformers

Example (Word2Vec):

- **Input**: "king"
- **Output**: A dense vector representing "king" based on its context.

3.10 Topic Modeling (LDA)

Purpose: Discovering abstract topics within a collection of documents.
Use Case: Organizing large datasets by identifying themes in text data (e.g., news articles, customer reviews).
Methods:

- **Gensim LDA Model**: gensim.models.LdaModel()
- **sklearn LDA**: sklearn.decomposition.LatentDirichletAllocation()

Example:

- **Input**: A collection of news articles.
- **Output**: Topics such as "politics", "technology", "sports"

4. Practical Use Cases of NLP

1. Customer Feedback Analysis

➔ **Amazon** uses NLP to analyze product reviews, helping them identify customer sentiment and improve product recommendations.

2. Chatbots & Virtual Assistants

➔ **Bank of America** uses its virtual assistant **Erica**, which employs NLP to assist customers with tasks like checking balances and paying bills.

3. Healthcare

➔ **IBM Watson Health** uses NLP to diagnose diseases by analysing patient data and identifying patterns that may indicate a particular condition.

4. Finance

➔ **Monzo** uses NLP to detect fraudulent activities and conduct risk assessments.

5. Marketing

➔ **Spotify** uses NLP to analyze user listening habits and preferences, generating personalized playlists and recommendations for each user.

References

- https://www.oracle.com/my/artificial-intelligence/what-is-natural-language-processing/
- https://www.nlplanet.org/course-practical-nlp/01-intro-to-nlp/04-n-grams
- https://medium.com/@asjad_ali/understanding-the-nlp-pipeline-a-comprehensive-guide-828b2b3cd4e2#:~:text=In%20Natural%20Language%20Processing%20
- https://docs.oracle.com/en/database/oracle/oracle-database/23/vecse/cosine-similarity.html
- https://huggingface.co/
- https://pypi.org/project/gensim/
- https://spacy.io/
- https://www.nltk.org/


Q
A