# **Unlocking Language with AI: A Beginner’s Guide to NLP**

By Reid Dial, Tony Reyes, Jose Marin, Milo Dufresne-MacDonald

Natural Language Processing (NLP) is reshaping how we interact with technology — from intelligent search engines to voice assistants and beyond.

But how do machines actually "understand" human language?

In this article, we break down the core concepts of NLP, show why they matter, and share simple practical examples to make the ideas come alive.

## **What is Natural Language Processing?**

NLP is the field of computer science focused on enabling machines to analyze, interpret, and generate human language.

It bridges the communication gap between people and computers by making sense of unstructured text data.

## **Why NLP Matters**

* Unlocks massive amounts of unstructured data (emails, articles, social media posts)
* Powers critical applications in healthcare, finance, marketing, and education
* Helps organizations derive insights that were previously unreachable

## **Building Blocks of NLP**

Before diving deeper, here are a few key concepts:

* **Corpus/Corpora**: Large collections of text used for training models
* **Tokens**: Individual units (words, numbers, or symbols) extracted from text
* **Stop Words**: Common words like "the", "is", and "in" that add little meaning and are often removed during analysis

Rather than memorizing terminology, it’s more useful to think about how these pieces work together during the NLP process.

## **Preparing Text for Analysis: Preprocessing**

Raw text is messy. Preprocessing transforms it into a cleaner format that machines can work with.

Common preprocessing steps include:

* **Tokenization**: Breaking text into individual words or tokens
* **Stopword Removal**: Eliminating frequently used words that add little meaning
* **Stemming and Lemmatization**: Reducing words to their base form (e.g., "running" becomes "run")

Example:

Original Text:

"The cats were running across the fields."

After Preprocessing:

"cat run field"

Simple Code Example (text version):

import spacy

nlp = spacy.load('en\_core\_web\_sm')

text = "The cats were running across the fields."

doc = nlp(text)

tokens = [token.lemma\_ for token in doc if not token.is\_stop]

print(tokens) # Output: ['cat', 'run', 'field']

## **Representing Text as Data: Feature Extraction**

Once cleaned, text must be converted into numerical data. Two foundational techniques are:

* **Bag of Words (BoW)**: Counts the number of times each word appears, ignoring grammar and word order
* **TF-IDF (Term Frequency–Inverse Document Frequency)**: Weighs words by importance, reducing the impact of commonly used words

Example:

python

from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer

documents = ["John likes to watch movies", "Mary also likes to watch football games"]

# Bag of Words

vectorizer = CountVectorizer()

X\_bow = vectorizer.fit\_transform(documents)

print(vectorizer.get\_feature\_names\_out())

print(X\_bow.toarray())

# TF-IDF

tfidf = TfidfVectorizer()

X\_tfidf = tfidf.fit\_transform(documents)

print(tfidf.get\_feature\_names\_out())

print(X\_tfidf.toarray())

These methods form the foundation for turning language into something machine-readable.

## **Extracting Deeper Meaning**

Beyond word counts, NLP models capture context and relationships through techniques like:

* **n-Grams**: Capturing word pairs like "not good" that hold special meaning
* **Chunking**: Grouping words into meaningful phrases (e.g., "the big dog" as a noun phrase)
* **Named Entity Recognition (NER)**: Identifying names, locations, dates, etc., in text

Example of using SpaCy for NER:

import spacy

nlp = spacy.load('en\_core\_web\_sm')

doc = nlp("Apple is looking at buying U.K. startup for $1 billion")

for ent in doc.ents:

print(ent.text, ent.label\_)

This helps machines recognize that "Apple" here refers to a company, not a fruit.

## **Embedding Meaning: Word Vectors**

Word embeddings represent each word as a dense vector of numbers, capturing semantic relationships.

For example:

* "King" and "Queen" appear close together in vector space
* "Paris" and "France" maintain a similar relationship

Techniques like **Word2Vec** allow models to detect nuanced relationships that simple counts miss.

## **Scaling Up: Advanced NLP Applications**

* **Topic Modeling**: Identifies hidden themes across large text collections by grouping related words
* **Sentiment Analysis**: Determines if a text expresses positive, negative, or neutral emotions — crucial for brand monitoring and customer feedback

Both tasks build directly on preprocessing and feature extraction techniques.

## **Applied Example: Naive Bayes Classifier**

Naive Bayes is a supervised machine learning algorithm that excels at text classification, like spam detection.

Basic Code Example:

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

# Example Data

texts = ["Win big prizes", "Low rates available", "Important update", "Congratulations, you won"]

labels = ["spam", "spam", "spam", "spam"]

# Vectorize

vectorizer = CountVectorizer()

X = vectorizer.fit\_transform(texts)

# Train Model

model = MultinomialNB()

model.fit(X, labels)

# Predict

sample = ["Win a million dollars now"]

sample\_vec = vectorizer.transform(sample)

print(model.predict(sample\_vec))

Even though it assumes that features are independent (which isn't always true), it works surprisingly well in practice.

## **Final Thoughts**

Natural Language Processing is no longer a niche field. It’s central to how we interact with information today.

Understanding the fundamentals — from cleaning text to building models — provides a strong foundation for exploring machine learning and AI even further.

As new models continue to emerge, the building blocks discussed here remain critical for anyone who wants to harness the power of language and data.