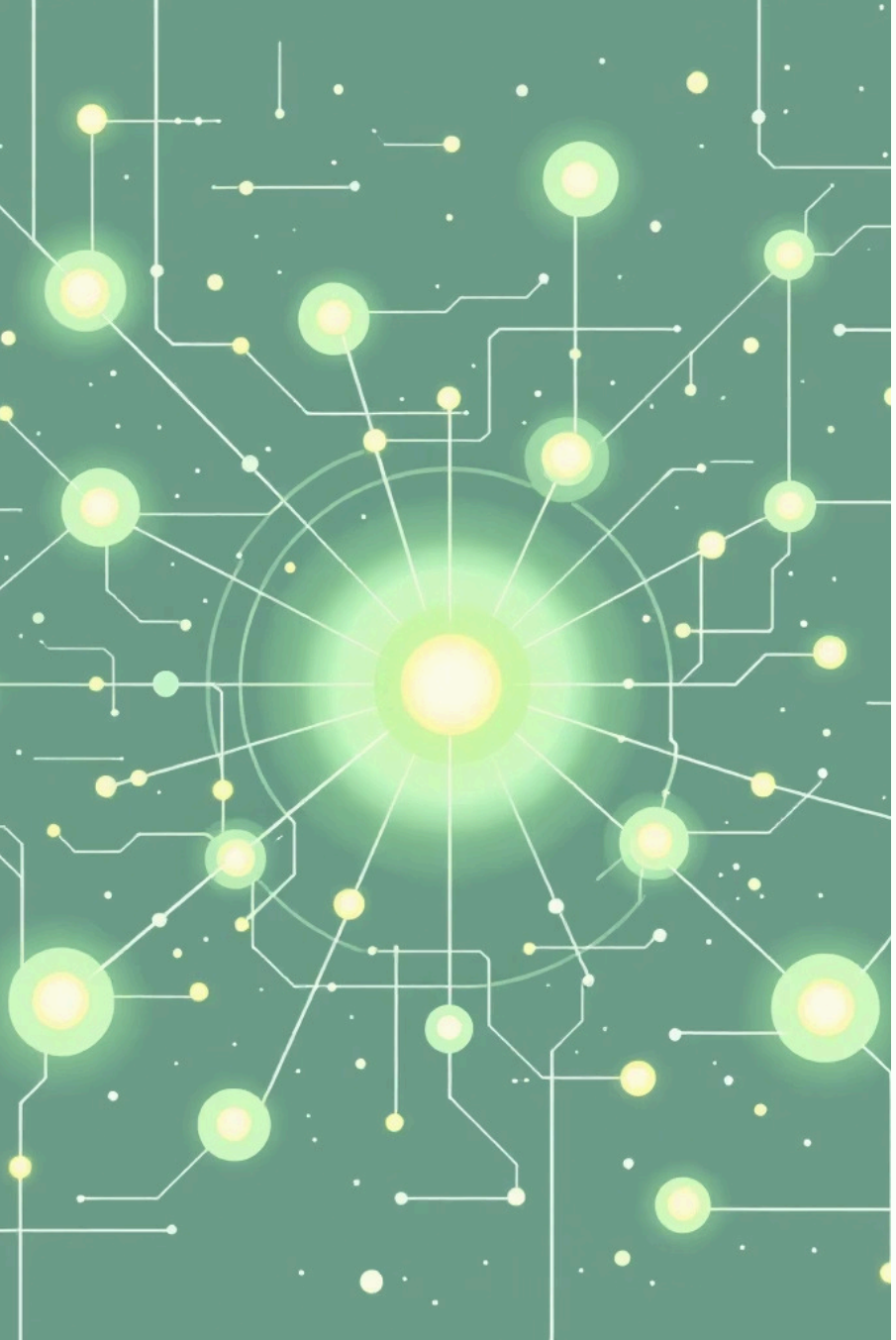


Natural Language Processing

Natural Language Processing (NLP) teaches computers how to understand, interpret, and create human language in helpful ways.

FOUNDATION

Key Components of NLP

Tokenization

Splitting text into individual words or sentences, called tokens. Example: "I am good" -> ["I", "am", "good"].

Parts of Speech

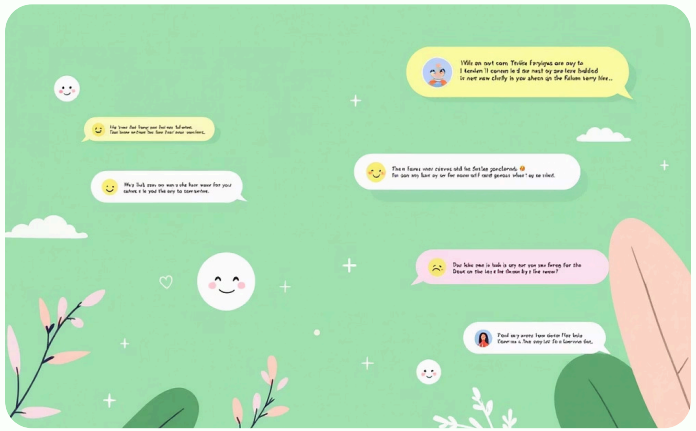
Labeling words with their grammatical roles: noun, verb, adjective, etc.

Named Entity Recognition

Locating and classifying entities such as people, places, and organizations. Example: "John (person) lives in Paris (place)."

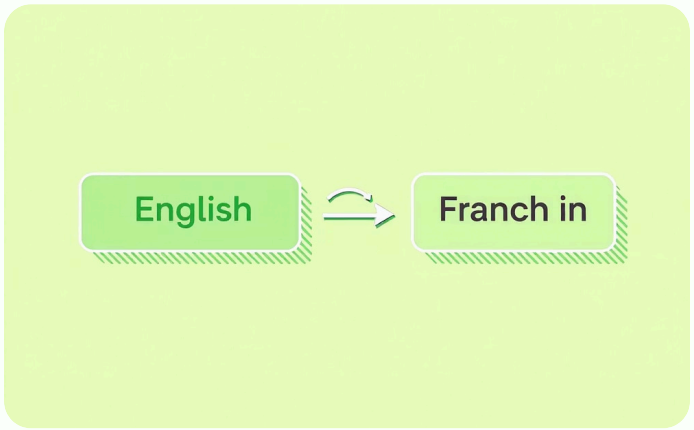
Core NLP Capabilities

NLP helps computers understand and create human language. Important uses include Sentiment Analysis, Machine Translation, and Text Summarization. These tools power many AI applications we use every day.



Sentiment Analysis

This AI tool figures out the feeling behind text. It tells if text is positive, negative, or neutral. It's very useful for understanding what customers think and watching public opinion.



Machine Translation

Machine Translation changes text or speech from one language to another automatically. Smart NLP models understand what the words mean to give correct translations. This helps people around the world talk to each other easily.

Text Summarization

This tool makes long documents shorter, keeping only the main ideas. It helps people quickly get the key points without reading everything. This saves time when there's too much information.

PROCESS

How NLP Works

1

Data Collection

Gather text data from various sources.

2

Pre-processing

Remove unwanted characters, tokenization, lemmatization.

3

Cleaning

Stemming, lemmatization, removing stop words.

4

Modeling

Apply Bag of Words, TF-IDF, or Word Embeddings.

5

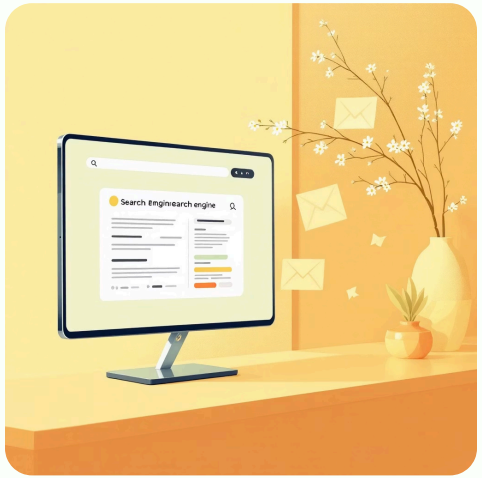
Training

Learn patterns from data to create the final model.

Real-World Applications

Chatbots & Virtual Assistants

Voice assistants like Siri and Alexa understand and respond to spoken words.



Search Engines

Provides intelligent search results and query understanding.

Spam Detection

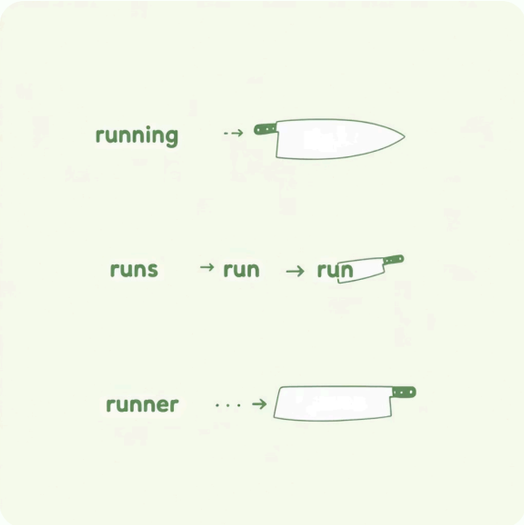
Filters unwanted messages and identifies malicious content.

Social Media Monitoring

Analyzes sentiments and trends across social media platforms.

TECHNIQUES

Stemming vs. Lemmatization



Stemming

Reduces words to root form by chopping off endings. Example: "running" → "run", "congratulations" → "congrat".

- PorterStemmer
- LancasterStemmer
- SnowballStemmer (supports multiple languages)

Lemmatization

Reduces words to dictionary form considering context and part of speech. Example: "better" → "good", "running" → "run".

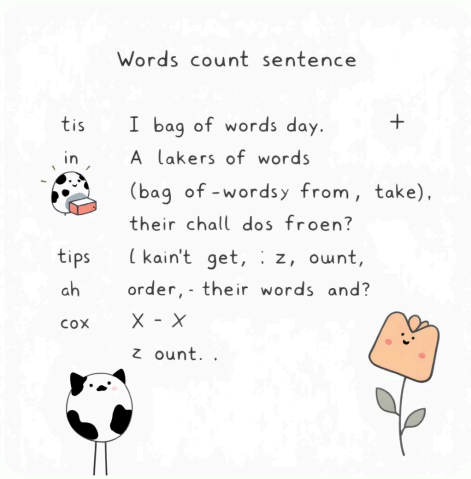
- WordNetLemmatizer
- Context-aware processing
- More accurate than stemming

Text Embedding Methods

Converting text into numerical vectors for machine learning algorithms.

One-Hot Encoding

Simple binary vectors (e.g., cat=[1,0,0]). Doesn't capture word relationships.



Bag of Words

Word count vectors ignoring grammar and order. Simple but loses context.

TF-IDF

Weights words by importance using TF × IDF. Highlights rare, meaningful words.

Word2Vec

Dense vectors capturing semantic meaning. Similar words have similar vector representations.

ADVANCED

TF-IDF: Term Frequency-Inverse Document Frequency

Formula

$$TF\text{-}IDF = TF \times IDF$$

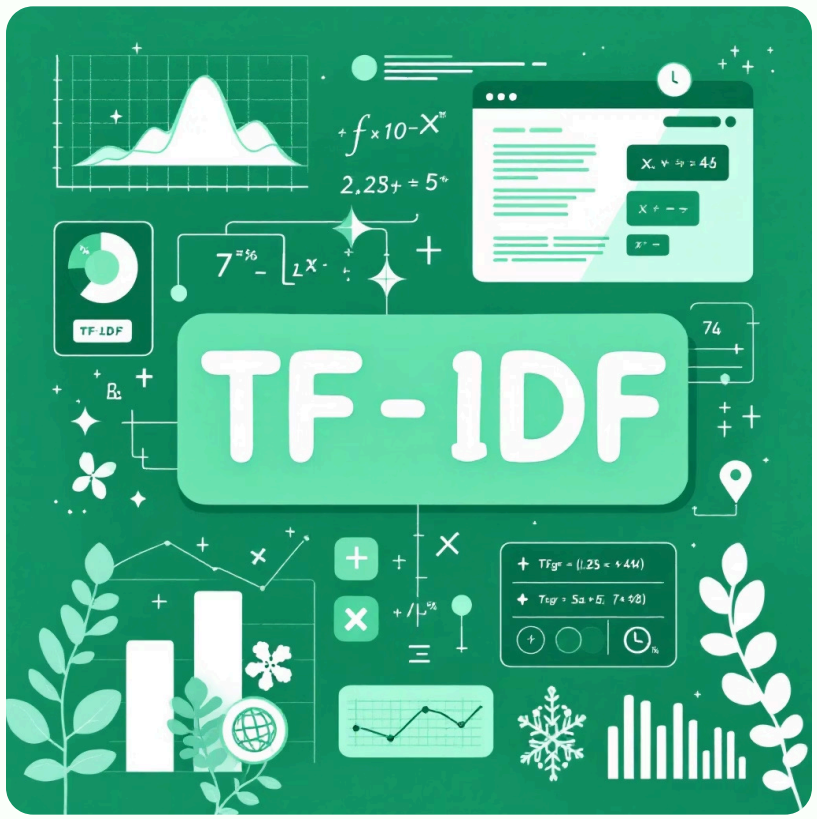
TF: How often a word appears in a document

IDF: How common or rare a word is across all documents

Frequent words in a document that are rare in the corpus receive higher weights, reflecting their importance.

Pros: More informative than BOW, considers word importance, fixed size.

Cons: Sparse matrix can lead to overfitting, still ignores context and word order.



Word2Vec: Semantic Understanding

Beyond Frequency: Semantic Representation

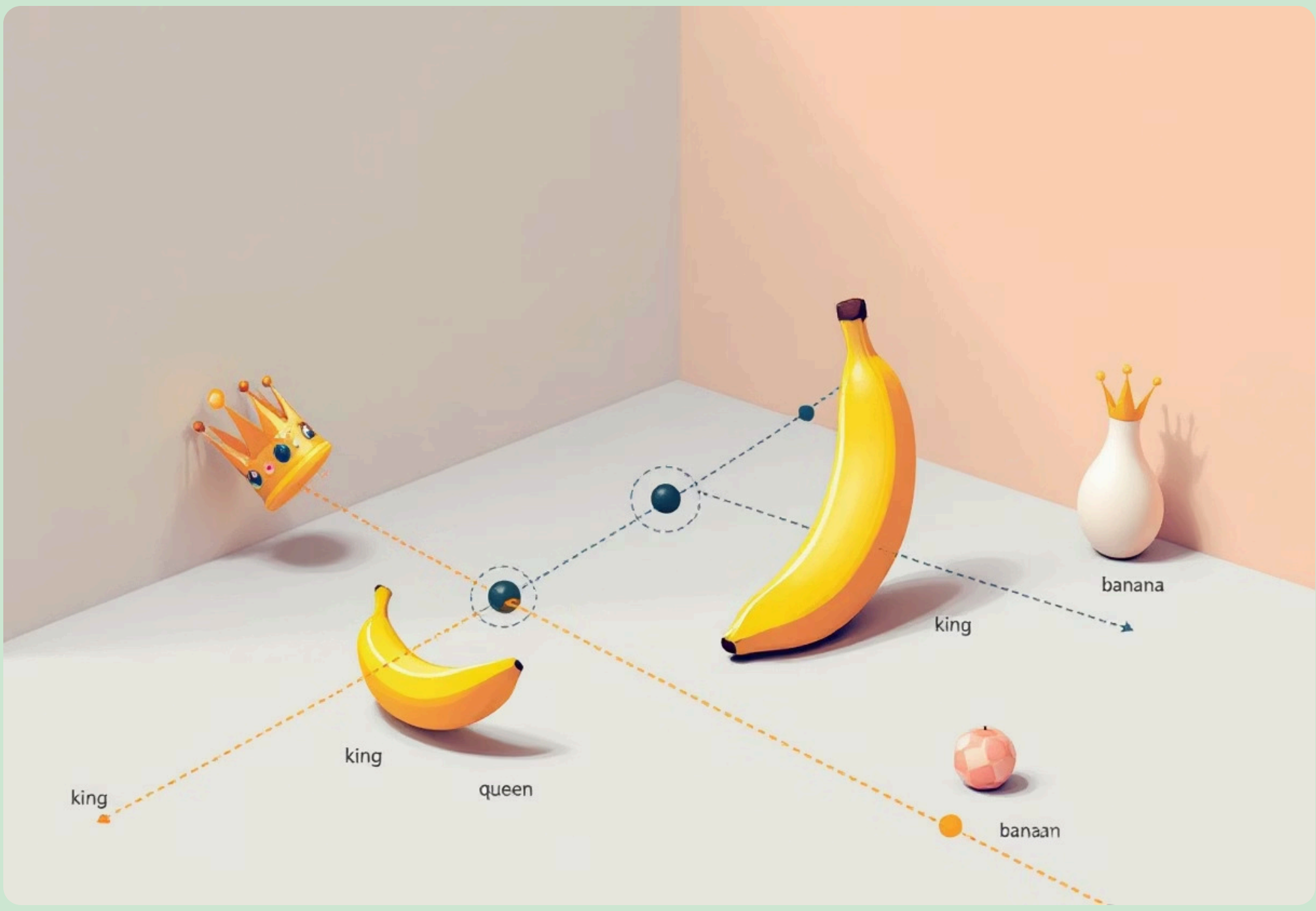
Word2Vec is a breakthrough in how computers understand language. It learns to represent words as numerical codes (vectors) in a way that captures their meaning. Unlike older methods, Word2Vec understands what words mean and how they relate to each other, placing words with similar meanings close together in this numerical space.

CBOW

The Continuous Bag of Words (CBOW) model works by guessing a missing word based on the words around it. It takes the numerical codes of the surrounding words, combines them, and then predicts the word that should fit in the middle. CBOW usually trains faster and performs well when you have a large amount of text data.

Skip-Gram

The Skip-Gram model does the opposite of CBOW. It takes a single word and tries to predict the words that are likely to appear near it. This model is often better at understanding less common words and can be a bit more accurate, especially when training with smaller datasets.



Cosine Similarity: This method measures how similar two words are by looking at the angle between their numerical codes (vectors). The formula for distance is $1 - \cos(\theta)$, where θ is the angle between the vectors. A smaller distance (meaning a score closer to 1) means the words are more alike in meaning. Scores range from 0 (very different) to 1 (identical). This helps us find synonyms, give recommendations, and understand how words relate to each other.

Word2Vec's ability to understand word meanings has many real-world uses. It helps improve recommendation systems, makes search engines smarter, assists in translating languages, and can even analyze emotions in text.

MODERN ERA

Transformer-Based Embeddings

State-of-the-art models like BERT and GPT capture context dynamically–the same word can have different embeddings based on sentence context.

01

Input Text

The original sentence or document provided to the model.

02

Tokenization

Breaking down the input text into smaller units, such as words or subword units.

03

Embedding Lookup

Converting tokens into initial numerical vector representations.

04

Positional Encoding

Adding information about the relative or absolute position of tokens in the sequence to the embeddings.

05

Transformer Layers

Multiple layers applying self-attention mechanisms and feed-forward neural networks to process the vectors, capturing contextual relationships.

06

Output Embeddings

The final, rich contextualized vectors representing each token, ready for various downstream NLP tasks.

Unlike earlier methods, transformers process text through a sophisticated pipeline to generate these rich, contextualized representations. The example below shows how the word "bank" can have different embeddings depending on its context:

Popular Libraries: NLTK, spaCy, Transformers