Natural Language Processing (NLP)

Core of Modern NLP

Equipping You with Research Depth and Industry Skills – Data Science Oriented

By:

Dr. Zohair Ahmed

Recommended Books

Speech and Language Processing

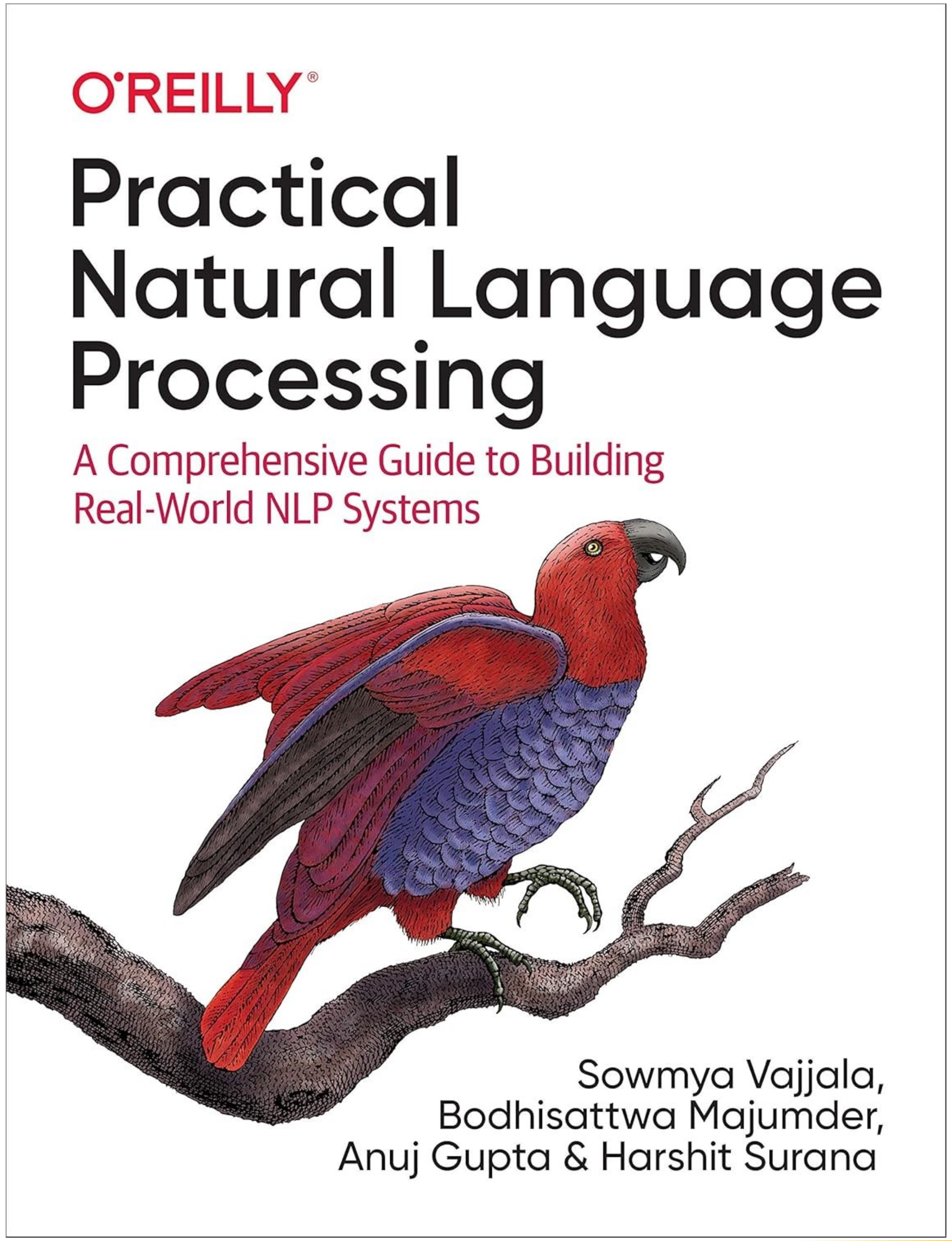
An Introduction to Natural Language Processing,
Computational Linguistics, and Speech Recognition
with Language Models

Third Edition draft

Daniel Jurafsky
Stanford University

James H. Martin
University of Colorado at Boulder

Draft of August 24, 2025. Comments and typos welcome!



Natural Language Processing (NLP)

- **Natural Language Processing (NLP)** is a branch of artificial intelligence (AI) that focuses on helping computers understand, interpret, and respond to human language.
- Think of it like this:
- Humans speak in natural languages (like English, Arabic, Hindi, French).
- Computers “speak” in binary or programming languages.
- NLP acts as the **bridge** between the two.
- Some common **examples of NLP** you already interact with:
- Google Translate (language translation)
- Siri or Alexa (voice assistants)
- ChatGPT (text understanding and generation)
- Spam filters in email (detecting unwanted messages)

Real Time Use Cases

Common NLP Libraries

- **General NLP toolkits**
- **NLTK (Natural Language Toolkit)** – classic library, great for teaching basics: tokenization, stemming, POS tagging.
- **spaCy** – fast, industrial-strength, with built-in pipelines for tokenization, lemmatization, POS tagging, dependency parsing, and named entity recognition.
- **Deep Learning–based NLP**
- **Hugging Face Transformers** – the go-to for pretrained models like BERT, GPT, T5. Widely used in research & production.
- **AllenNLP** – built on PyTorch, good for experiments in academic NLP.
- **Fairseq (Meta AI)** – powerful for sequence-to-sequence models like translation and summarization.

Common NLP Libraries

- **Text Processing & Utilities**
- **TextBlob** – simple API for sentiment analysis, part-of-speech tagging, and translation (good for quick demos).
- **Gensim** – specializes in topic modeling and word embeddings (e.g., Word2Vec, Doc2Vec, LDA).
- **Stanford CoreNLP** – Java-based, but very strong for parsing and linguistic features.
- **Speech & Multimodal NLP**
- **SpeechRecognition** – for converting speech to text.
- **OpenAI Whisper** (and Hugging Face integration) – state-of-the-art speech recognition.

Why NLP is Popular right now

- **Why is NLP Booming Today?**
- NLP has existed in academia for decades.
- Real industry applications rushed in the last 7–10 years.

Reason 1 – Pre-trained Models & APIs

- Training NLP models needs **huge data + compute**.
- Big companies (Google, Facebook, OpenAI, Amazon) provide **pre-trained models**:
  - **fastText (Facebook)**
  - **TensorFlow Hub (Google)** – BERT, USE, etc.
  - **GPT-3 (OpenAI)** – $4M training cost, now accessible.
- **Transfer learning** makes NLP accessible to small companies.

Reason 2 – Open-Source Ecosystem

- Earlier: coding from scratch in C++
- Now: Python + open-source libraries = rapid prototyping
  - **spaCy**
  - **Gensim**
  - **NLTK**
- Backed by community, continuously improving.

Reason 3 – Affordable Hardware & Cloud

- Past: costly GPUs & infrastructure.
- Present:
  - Cheaper GPUs.
  - Cloud platforms: **AWS, Azure, Google Cloud**.
  - Rent resources instead of buying.

Reason 4 – Learning Resources

- **Reason 4 – Learning Resources**
- Before: limited access (only universities).
- Now: global accessibility.
  - Free content: **YouTube, blogs, Kaggle, StackOverflow**
  - Courses: **Coursera, Udemy, bootcamps**
  - Communities: **Discord, Zoom, competitions**
- Anyone can learn NLP quickly.

Reason 5 – Big Tech Investments

- Tech giants building ecosystem:
  - Google: TensorFlow, Google Home
  - Amazon: Alexa, Echo devices
  - Meta: fastText, FAIR research
- Heavy investments → smaller companies follow (FOMO effect).
- FOMO = Fear of Missing Out
- When big tech companies (Google, Amazon, Meta, etc.) invest in NLP
- Smaller companies feel pressure to adopt it too, because they don't want to miss opportunities, market share, or innovation trends.

NLP is booming

- Free pre-trained models
- Open-source libraries
- Cloud & cheap hardware
- Easy learning resources
- Big tech investments
- Future: **More rapid advancements, wider adoption**.

Techniques to Solve NLP Problems

- **Techniques to Solve NLP Problems**
- NLP has many approaches.
- **3 broad categories of techniques**.

Technique 1 – Rules & Heuristics

- Based on **patterns, rules, and regex**.
- Example: Gmail flight ticket → auto-extracted summary.
- **Regex approach:**
  – “Booking Ref: XXXXX” → extract confirmation number.
- Pros: Simple, precise.
- Cons: Not scalable, hard for complex text.
- **Example – Rules in Action**
- Search “Elon Musk” → Google sidebar info.
- Likely uses **pattern-based extraction**.
- **Key idea:** Rules can solve information extraction tasks without ML.

Technique 2 – Statistical / Machine Learning

- **Spam Detection**
  - Raw email text → Number vector → Classifier.
- Common workflow:
  - Raw text
  - Preprocessing (cleaning, lemmatization, etc.)
  - Convert text → numbers (e.g., CountVectorizer, TF-IDF)
  - Classification (e.g., Naïve Bayes)
- Pros: More general than rules.
- Cons: Struggles with unseen words/phrases.

- **Example – Spam Email**
- Text: *"Urgent business assistance… 55 million USD…"*
- Suspicious keywords trigger spam detection.
- **Pipeline:**
  - CountVectorizer → numeric representation.
  - Naïve Bayes → spam/not spam.

Technique 2 – Statistical / Machine Learning

- **Spam Detection**
  - Raw email text → Number vector → Classifier.
- Common workflow:
  - Raw text
  - Preprocessing (cleaning, lemmatization, etc.)
  - Convert text → numbers (e.g., CountVectorizer, TF-IDF)
  - Classification (e.g., Naïve Bayes)
- Pros: More general than rules.
- Cons: Struggles with unseen words/phrases.

- **Example – Spam Email**
- Text: *"Urgent business assistance… 55 million USD…"*
- Suspicious keywords trigger spam detection.
- **Pipeline:**
  - CountVectorizer → numeric representation.
  - Naïve Bayes → spam/not spam.

What is CountVectorizer?

- **CountVectorizer** is a technique to convert text into a **numeric representation** that machine learning models can understand. It essentially counts how often each word appears in a document. This is also known as **Bag of Words** (BoW).

What is CountVectorizer?

- **Step 1 – Tokenization**
  - Split text into individual words (tokens).
  - **Sentence 1:** "I love NLP." → Tokens: ["I", "love", "NLP"]
  - **Sentence 2:** "NLP is amazing!" → Tokens: ["NLP", "is", "amazing"]
- **Step 2 - Build Vocabulary**
  - Create a vocabulary of unique words from the entire text.
  - Vocabulary: ["I", "love", "NLP", "is", "amazing"]
- Step 3 - Count Word Occurrences
  - Count how often each word appears in each document.
  - **Sentence 1:** ["I", "love", "NLP"] → Word counts: I:1, love:1, NLP:1
  - **Sentence 2:** ["NLP", "is", "amazing"] → Word counts: NLP:1, is:1, amazing:1
- Step 4 - Create Vectors
  - Convert the word counts into vectors (numeric form).
  - Sentence 1 Vector: [1, 1, 1, 0, 0]
  - Sentence 2 Vector: [0, 0, 1, 1, 1]

Word	I	love	NLP	is	amazing
Sentence 1	1	1	1	0	0
Sentence 2	0	0	1	1	1

Challenge with Statistical ML

- Example:
  - Training: *“Hurry up to win cash”*, *“You won a laptop”*.
  - New text: *“Rush for this great deal to win money”*.
- CountVectorizer fails if words are unseen.
- **Solution:** Use embeddings that capture semantic meaning.

Technique 3 – Deep Learning

- Uses **word embeddings / sentence embeddings**.
- Example: **BERT (Google)**, Hugging Face Transformers.
- Embeddings allow similar sentences to have similar vectors.
- Example:
  - “Win cash fast” ≈ “Hurry up to win money”
  - Embedding vectors → Cosine similarity.
- **Hugging Face Sentence Transformers**
- Input: “I love baby yoda!” vs. spam sentences.
- Output: Similarity scores (cosine similarity).
- Shows how embeddings capture **semantic meaning**
- https://huggingface.co/sentence-transformers

Cosine Similarity

- Imagine two arrows (vectors) starting at the same point.
- If they point in **exactly the same direction**, they are **very similar**.
- If they point in **opposite directions**, they are **very different**.
- If they are **at a right angle**, they are **not similar at all**.
- Cosine similarity just measures **the angle between two vectors**.

- For two vectors **A** and **B**:
- Cosine Similarity= $\frac{\boldsymbol{A}.\boldsymbol{B}}{||\boldsymbol{A}||\,||\boldsymbol{B}||}$
- ||A||= length (magnitude) of vector A.
- ||B||= length (magnitude) of vector B.
- Cosine similarity **high** (**close** meaning).
- Cosine similarity **low** (**different** meaning).

Naive Bayes

- **Naive Bayes** is a simple but effective **probabilistic machine learning algorithm** based on **Bayes' Theorem**, and it's particularly well-suited for **classification tasks** like **text classification** (e.g., spam detection, sentiment analysis).
- It’s called "Naive" because it assumes that all features (words, in this case) are **independent** of each other, which is often **not true** in real-world scenarios. However, this simplification works surprisingly well in practice.

Bayes' Theorem:

- Bayes' Theorem is the foundation of Naive Bayes. It calculates the **probability** of a class C given some features X (in NLP, the features are words in the document).

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

- Where:
- P(C|X) = Posterior: Probability of class C given the features X.
- P(X|C) = Likelihood: Probability of features X given the class C.
- P(C) = Prior: The probability of class C occurring in the dataset.
- P(X) = Evidence: The probability of features X occurring, independent of the class.

Bayes' Theorem:

- Bayes' Theorem is the foundation of Naive Bayes. It calculates the **probability** of a class C given some features X (in NLP, the features are words in the document).

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

- Where:
- P(C|X) = Posterior: Probability of class C given the features X.
- P(X|C) = Likelihood: Probability of features X given the class C.
- P(C) = Prior: The probability of class C occurring in the dataset.
- P(X) = Evidence: The probability of features X occurring, independent of the class.

Bayes' Theorem:

Email	Class
"Win money now"	Spam
"Cheap offer, win now"	Spam
"Meeting schedule"	Not Spam
"Lunch at noon"	Not Spam

- **Class Priors:**
- $\mathrm{P}(\text{Spam}) = \frac{2}{4} = 0.5\,;\ P(\text{Not Spam}) = \frac{2}{4} = 0.5$
- **Likelihoods:**
- Let's say we are given the new email: **"win money".**
- **P(win|Spam):** From the training data, "win" appears 2 times in 2 spam emails (out of 2 total spam emails). So,
- $P(\text{win}|\text{Spam}) = \frac{2}{2} = 1$
- **P(money|Spam):** "money" appears 1 time in 2 spam emails (out of 2 total spam emails). So,
- $P(\text{money}|\text{Spam}) = \frac{1}{2} = 0.5$
- Similarly, we calculate for the Not Spam class.

- **Posterior Calculation:**
- **For Spam:**
- $P(\text{Spam}|\text{win, money}) \propto P(\text{win}|\text{Spam}) \cdot P(\text{money}|\text{Spam}) \cdot P(\text{Spam}) = 1 \cdot 0.5 \cdot 0.5 = 0.25$
- **For Not Spam:**
- $P(\text{Not Spam}|\text{win, money}) \propto P(\text{win}|\text{Not Spam}) \cdot P(\text{money}|\text{Not Spam}) \cdot P(\text{Not Spam}) = 0.25 \cdot 0.25 \cdot 0.5 = 0.03125$
- **Prediction:**
- Since the posterior probability of Spam is higher (0.25 vs. 0.03125), the algorithm classifies the email as Spam.