Text Mining and Low-Tech Fundamentals of NLP

A Short Course Tailored for Tomorrow's Business Leaders

by

Muhammad Al-Zafar

September 14, 2025

Learning Outcomes

1. **Module 1:** Reasoning and Proof Models
2. **Module 2:** Foundations of Text Mining
3. **Module 3:** Text Preprocessing
4. **Module 4:** Exploratory Text Mining
5. **Module 5:** Applied Low-Tech NLP
6. **Additional Material:** (If time allows).
   - Exploring the `nltk` package in `Python`.
   - NLP best Practices in Business.

Lesson Plan

Our Approach: *Scaffolding*, i.e., some part theory and some part practical/guided lab sessions.

- These lectures are designed to be intensive and impactful.
- Spend $\pm$ 36 minutes per module.
- ± 20 minutes for theoretical foundations of each module.
- ± 15 minutes for practical examples and lab guided coding exercises that emphasize real-world applications.
- ± 1 minute of rest between modules.

Of course, this is just a rough outline, and we may adjust this based on the needs and requirements of the class; we may spend more time in certain modules and sections, and less time in others.

Classroom Discipline

- You must abide by all the rules and regulations set out by your institution. This includes upholding the moral, ethical, and behavioral standards outlined when registering.
- Please do not disturb your fellow classmates by having conversations with your mic on, as it will disrupt the flow of the delivery of the lesson.
- Feel free to mute and switch off your camera for any comfort breaks (using the washroom, or stretching), or attending to any emergencies.
- This is a conversational style lesson, so feel free to unmute your mic at any point and ask me questions or seek clarification on anything.

Course Material

- **Where can I access the material for this course?** https://github.com/MuhammadAlZafarKhan/Text-Mining-and-Low-Tech-Fundamentals-of-NLP.

Module 1: Reasoning and Proof Models

Models of Argument

- **Definition:** Structured frameworks for reasoning, showing how claims are supported by evidence and logic.
- In NLP, it is used in argument mining, discourse analysis, and AI reasoning.
- **Example:**
  - **Claim:** "The model is biased."
  - **Evidence:** "It misclassifies 70% of sentences with gendered pronouns."
  - **Warrant:** "Consistent misclassification indicates bias."

Proof by Demonstration

- **Definition:** Showing correctness by **working example** or execution, not formal math.
- Demonstrating an algorithm works on a real input.
- **Example:**
    - Running a sentiment analysis model on the text:
        - **Input:** "The movie was fantastic!".
        - **Output:** Positive.
- Demonstrates that the sentiment classifier functions.

Proof by Empirical Methods

- **Definition:** Validating a claim by repeated experiments and data-driven evidence.
- In NLP, it is used for model evaluation through benchmarks, test sets, or metrics.
- **Example:**
  - An NLP classifier tested on $10\,000$ emails achieves 92% accuracy, 0.89 F1-score.
  - This is empirical proof that the model performs well on average.

Mathematical Proofs

- **Definition:** Rigorous derivations using logic, theorems, and formal definitions.
- Used in algorithm analysis, complexity proofs, or correctness proofs.
- **Example:**
  - **Claim:** "The edit distance dynamic programming algorithm runs in $\mathcal{O}(nm)$".
  - **Proof:** By analyzing the DP (dynamic programming) table size $(n \times m)$ and constant-time cell updates.

Proof by Hermeneutics

- **Definition:** Interpretation-based justification, often used in qualitative, philosophical, or human-centered CS/NLP research.
- In NLP, it is used for explaining results by interpreting meaning rather than strict numerical or formal reasoning.
- **Example:**
  - Analyzing why a chatbot's response "I understand your feelings" is more empathetic:
    - **Interpretation:** It mirrors human conversational norms.
  - Proof through interpretive understanding of language use.

Lab 1

To get familiar with the basics of the aforementioned methods of proof in `Python`.

Module 2: Foundations of Text Mining

What is Text Mining?

Text mining (also called **text data mining** or **text analytics**) is the process of extracting useful information, patterns, and knowledge from unstructured text data using techniques from natural language processing (NLP), machine learning (ML), statistics, and information retrieval.

- What is the importance of text mining, and why do we need it? Since most data in the real world (emails, articles, reports, social media posts, reviews, etc.) is in unstructured text form, text mining helps convert this raw text into structured and meaningful insights.

What are the Applications of Text Mining to Humanities, Social Science, Medicine, and Business?

- **Business Intelligence:** It is used to review customer reviews and feedback about products or services offered.
- **Healthcare:** It is used for mining medical records for diagnosis patterns.
- **Finance:** It is used for extracting insights from news articles for stock prediction.
- **Social Media:** It is used for sentiment analysis and trend tracking.
- **Legal:** It is used for analyzing case documents for precedents.
- **Academic Research:** It is used for summarizing and categorizing scientific articles.

Low-Tech Approaches to Text Mining

- These are simpler, rule-based, or statistical methods that do not require heavy computation or advanced ML. They often focus on word frequency, simple matching, or dictionaries.
- It is easy to implement.
- It is transparent and interpretable.
- It requires minimal computations.
- However, it can give shallow insights, it cannot handle synonyms, it does not understand context (for example, like sarcasm), nor can it understand semantics.
- The methods do not scale well to larger datasets.
- **Common Methods:** Word Count/Frequency Analysis, Concordance/Keyword in Context (KWIC), Lexicon-Based Sentiment Analysis, Collocation Detection, Regular Expression (RegEx) and Rules-Based Extraction.

High-Tech Approaches to Text Mining

- These use ML, Deep Learning (DL), and advanced NLP techniques to extract semantic, contextual, and predictive insights from text.
- They require more computational resources, large datasets, and specialized models.
- They provide rich and contextual insights.
- The methods scale to large datasets and even Big Data.
- The methods can handle ambiguity, synonyms, and context.
- **Common Methods:** Vector Space Models, Topic Modeling, ML Classification, DL for NLP, Named Entity Recognition (NER), Knowledge Graphs, etc.

Characters

- **Character:** The smallest unit of text, akin to how atoms are to a Chemist, an elementary particle is to a Physicist, or a prime number is to a Mathematician.
- Examples of characters: The Latin alphabet (a–z, A–Z), digits $(0-9)$, punctuation marks (.,?!;), white spaces (tabs, new lines), special symbols/characters (@, #, $, %), emojis, non-Latin alhphabets like Hindi, Chinese, Russian, Arabic, etc.

Encoding

- Computers do not “understand” characters directly – They only work with numbers (binary $= \{0, 1\}$).
- **Encoding** is the mapping between characters and their numeric representation.
- Common encoding standards:
  - **ASCII:** *American Standard Code for Information Interchange*.
  - **Extended ASCII/ISO-8859-1 (Latin-1)**
  - **Unicode**
  - **UTF-8 (Unicode Transformation Format)**

ASCII (American Standard Code for Information Interchange)

- Early encoding scheme composed of 7 bits that supports 128 characters.
- Supports only English letters, digits, and punctuation.
- **Example:** “A” $\rightarrow 65$, “a” $\rightarrow 97$.

Extended ASCII / ISO-8859-1 (Latin-1)

- Has 8 bits that represent 256 characters.
- The big innovation here is that it adds accented letters like ñ, á that are prevalent in many European languages.

Unicode

- A universal standard to represent text across all languages.
- Supports more than 150 scripts, including emojis.
- Within Unicode, Chinese has a unique code point.

UTF-8 (Unicode Transformation Format – 8-bit)

- It is the most ubiquitous encoding standard on the web.
- The encoding scheme has variable length. This means that:
    - 1 byte = 8 bits is used for ASCII. For example, A $\rightarrow 01000001$.
    - Up to 4 bytes for other characters. For example, the smiling emoji $\rightarrow$ F0 9F 98 80.

The Importance of Encoding in NLP

- **Data Preprocessing:** Reading a file with the wrong encoding may cause errors.
- **Tokenization:** Proper encoding ensures punctuation, emojis, and non-Latin characters are handled correctly. We will speak more about *tokenization* later on!
- **Multilingual NLP:** Unicode allows mixing scripts (e.g., English and Hindi in the same dataset).
- **Model Training:** Neural networks rely on consistent numerical representations of characters/words.

Words, Subwords, Tokens, Sentences, and Embeddings

Depending on the granularity of analysis, text can be broken into:

- **Words:** The basic linguistic units of meaning in human language. Words are typically separated by spaces in languages like English. For example, "Natural Language Processing" has 3 words. However, in other languages like Chinese and Japanese, there are no explicit spaces.
- **Subwords:** These are smaller units obtained by splitting words into meaningful pieces. For example, the word "unhappiness" can be broken up into "un", "happi", and "ness". Famous models like `BERT` (Bidirectional Encoder Representations from Transformers) use algorithms like `WordPiece` and `SentencePiece` to handle subwords.

- **Tokenization:** It is the fundamental process of breaking down raw text into smaller, manageable units called **tokens**. These tokens are typically words, but can also be subwords or characters. The primary purpose of tokenization is to transform unstructured text into a standardized format that NLP models can understand and process, enabling them to analyze, understand, and derive meaning from human language for various tasks.
- **Tokens:** These are the units a model actually processes after tokenization. For example, consider "Hello World", the tokens can be:
  - **Characters:** [H, e, l, l, o, W, o, r, l, d]
  - **Words:** [Hello, World]
  - **Subwords:** [Hel, ##lo, World]

  A common misconception is that a token = word, but this is not necessarily the case!

- **Sentence:** A *sentence* is a specific combination of words and subwords that form a linguistic unit. In NLP, sentences are often detected by punctuation marks (., ,, ;, ?, !,. . .) For example, “I love NLP. It is fascinating!” has two sentences. However, to a computer, segmentation is not always as trivial as the previous example. As a demonstration, see this sentence “Dr. Patel went to the U.S. He works in A.I.” Thus, we need careful handling of abbreviations.
- **Embeddings:** These are numerical vector representations of text units (words, tokens, sentences). With embeddings, the goal is to capture semantic meaning so that similar items are close in vector space.

Corpora

- The singular form is *corpus*, the plural is *corpora*.
- They are large collections of text documents used for training or evaluation.
- Examples of corpora in NLP are:
  - News articles.
  - Wikipedia
  - Research databases (PubMed, arXiv, etc.)
  - Dialogue datasets (Reddit, Quora, etc.)
- In NLP research, a corpus is the raw material from which tokens, embeddings, and models are built.

Terminology Summary

Term	**Definition**	**Example**
Word	Linguistic unit separated by spaces	"language", "processing"
Subword	Piece of a word (useful for rare words)	"un", "kind"
Token	Unit processed by the model (can be word, subword, or character)	"un", "equal"
Sentence	Sequence of words/tokens forming a unit of meaning	"I love chocolate. It tastes so good!"
Embedding	Vector representation of text in $\mathbb{R}^d$ (d-dimensional real vector space)	$[0.21, -0.437, \ldots]$
Corpus	Large text collection for NLP tasks	Wikipedia articles, news datasets

Lab 2

Applications of frequency analysis, KWIC, lexicon-based sentiment analysis, collocation detection, RegEx, and the various encoding schemes (ASCII, Unicode, UTF-8) in the **business context**.

Module 3: Text Preprocessing

What is Text Processing?

In NLP, text preprocessing is the initial step of cleaning and transforming raw text data into a structured and analyzable form. Since most NLP algorithms and models work on numerical or standardized representations, preprocessing ensures the **text is consistent**, **meaningful**, and **ready for analysis or modeling**.

Key Aspects of Text Preprocessing

1. **Cleaning the Text:** Removing punctuation, numbers, stopwords.
2. **Tokenization:** Splitting text into words, subwords, or sentences.
3. **Normalization:** Converting to lowercase, stemming, and lemmatization.

Lowercasing

- **Lowercasing** converts all text to lowercase to reduce duplication.
- Example: “Customer Service” $\rightarrow$ “customer service”.
- Example: “ELON MUSK” $\rightarrow$ “elon musk”.

Tokenization

- We have seen it before in Module 2.
- Essentuially, tokens split text into smaller units: Words, subwords, and sentences.
- Example: “I love NLP” $\rightarrow$ [“I”, “love”, “NLP”].

Removing Punctuation and Special Characters

- Cleans the text by removing characters like !, @, #, $, etc.
- Example: “Great product!!!” → “Great product”.

Removing Stopwords

- **Stopwords** are common words with little semantic meaning.
- Examples: “the”, “is”, “and”, etc.

Stemming

- **Stemming** is the process of reducing words to their root words.
- Example: “running”, “runs”, “ran” $\rightarrow$ “run”.
- Example: “connection”, “connects”, “connected”, “connecting”, “connections” $\rightarrow$ “connect”.
- **Note:** The stem may not always be a valid word in the English language.
- Example: “argue”, “argued”, “argument”, “arguing”, “arguer” $\rightarrow$ “argu”.

Lemmatization

- **Lemmatization** is the process of reducing words to their dictionary form, which makes them more accurate and meaningful.
- Example: “better” $\rightarrow$ “good”.
- Example: “ate”, “eaten”, “eating” $\rightarrow$ “eat”.

Handling Numbers / Dates / URLs / Emails

- Replace or remove domain-specific entities if not needed for analysis.
- Example: “Order #12345” $\rightarrow$ “order”.

Text Normalization

- Converts special characters, accents, or multiple spaces into standard forms.
- Example: “café” $\rightarrow$ “cafe”.

The Importance of Text Preprocessing

- **Reduces Noise:** Removes irrelevant or misleading characters.
- **Improves Consistency:** Ensures similar words are treated the same way.
- **Enhances Model Performance:** Models learn better on clean, standardized input.
- **Enables Feature Extraction:** Such as frequency counts, embeddings, or sentiment features.

Lab 3

In this lab, we shall apply the text preprocessing skills learned in this module to real-world business scenarios.

Module 4: Exploratory Text Mining

What is Text Mining?

Text mining is the process of extracting meaningful information and patterns from unstructured text data. It combines techniques from linguistics, statistics, machine learning, and information retrieval to transform raw text into structured knowledge that can be analyzed.

The Process of Text Mining

1. **Text Collection:** Gathering documents, emails, reviews, reports, etc.
2. **Text Preprocessing:** Cleaning and preparing data (lowercasing, stopword removal, stemming, lemmatization, handling numbers/dates/URLs).
3. **Feature Extraction:** Converting text into structured forms (Bag-of-Words, TF-IDF, word embeddings).
4. **Pattern Discovery:** Applying algorithms to find associations, clusters, sentiments, or trends.
5. **Knowledge Representation:** Presenting results as summaries, dashboards, or decision-support insights.

How is Text Mining Used in Business?

- **Customer Sentiment Analysis:** Used for understanding reviews and feedback from customers.
- **Fraud Detection:** Used for finding suspicious terms in financial/textual records.
- **Market Intelligence Analysis:** Used to discover trends in news and reports.
- **Email Filtering:** Used for classifying spam vs. non-spam.
- **Contractual Analysis:** Used to extract legal terms, risks, obligations, etc.

A Warm-Up Example

Scenario: Suppose a company collects $10\ 000$ customer reviews. One of the raw reviews is: "The delivery was late and the support team was unhelpful."
Task: What can we gather from this review, based on text mining?

- Upon preprocessing, the following keywords standout: ["delivery", "late", "support", "team", "unhelpful"].
- Frequency analysis shows words like "late", and "unhelpful" are common among many other reviews as well.
- This signals service issues.
- Based on only this one review, we can gather that the customer was not very happy. However, what happens when you are a multinational conglomerate like Amazon or Reliance Industries, it is impractical to read each review one-by-one, hence, we need to do sentiment analysis.

Word Counts

- **Word counts** are the total number of times a word appears in a text or a collection of documents.
- The purpose of word counts is to help identify common words or potential keywords in a corpus.
- Example: A customer posts the following review: "The product is good. The product is cheap."
- The word counts analysis is

Word	**Count**
"the"	2
"product"	2
"is"	2
"good"	1
"cheap"	1

Term Frequencies

- The **term frequency** is the frequency of a term in a document, usually normalized by the total number of words.
- Mathematically, it is given by

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

- It is used to measure how important a word is in a document, relative to its length.
- Example: In the previous example, the term frequency for the word "product" is

$$TF(\text{“product”}) = \frac{2}{8} = 0.25.$$

n-Grams

- ***n*-grams** are sequences of n consecutive words in a text.
- They are used to capture context and word patterns, not just individual words.
- **Unigrams:** These are **single** words: [“The”, “product”, “is”].
- **Bigrams:** These are **2-word** sequences: [“The product”, “product is”, “is good”].
- **Trigrams:** These are **3-word** sequences: [“The product is”, “product is good”].
- Applications in business:
    - Used to detect common phrases in customer reviews (“fast delivery”, “poor customer service”).
    - Used to improve text classification by including phrases instead of just single words.

Zipf's Law

- **Zipf's law** is an empirical law about the frequency of words in natural language. It states that:

  In a given corpus, the frequency of any word is inversely proportional to its rank in the frequency table.

- Mathematically, Zipf's law is given by

$$\boxed{f(r) \propto \frac{1}{r^s}}$$

where $f(r)$ is the frequency of the word, r is the rank of the word when words are sorted by frequency (most frequent $= 1$), $s \approx 1$ for natural languages (close to 1).

- Equivalently, Zipf's law can be expressed as

$$\boxed{f(r) = \frac{C}{r}}$$

  where C is a constant that depends on the corpus size.
- The key insights that we gauge from Zipf's law are
  - There are a few words which are extremely common, e.g., "the", "and", "of".
  - Most words are rare, appearing only once or twice in a large corpus.
  - This creates a long-tail distribution: The top 10 words might cover 20–30% of all word occurrences.

Applications of Zipf's Law in Business

- **Stopword Identification:** Words that appear extremely often (like "the", "is") can be safely removed since they carry little semantic information.
- **Keyword Extraction:** Focuses on words in the middle frequency range, which often represent important topics.
- **Data Compression/Storage:** Rare words appear less often. This implies that dictionaries and storage in NLP pipelines are optimized.
- **Language Modeling:** Helps in predicting word probabilities and smoothing techniques.

Example Application of Zipf's Law

Situation: While reading an executive report, we have the following word counts in the corpus:

Word	**Count**	**Rank**
"the"	1000	1
"product"	500	2
"is"	400	3
"excellent"	50	4
"fast"	25	5

- Checking Zipf's law
  - $f(1)/f(2) = 1000/500 = 2 = 2/1.$
  - $f(2)/f(3) = 500/400 = 1.25 \approx 1.5 = 3/2.$
- The law is approximate, but captures the heavy head and long tail behavior.

Keywords in Context (KWIC)

- **Keywords in Context (KWIC)** is a concordance tool widely used to study how words are used in real texts.
- It is a way of displaying all occurrences of a given keyword in a text (or a corpus) together with their surrounding words (the context).
- Instead of just listing word counts, KWIC shows the meaning, usage, and nuance of how words appear in sentences.

Why is KWIC Important?

- Helps understand word semantics and usage.
- Reveals collocations (words that often appear together).
- Useful for sentiment analysis, market research, business intelligence, and lexical studies.
- Often used in search engines, chatbots, and digital assistants.

Co-occurrences

- A **co-occurrence** simply means that two words appear near each other in a corpus.
- It is a more general concept than collocation.
- Any two words appearing within a defined window (say, ± 5 words, or the same sentence, or the same document) are considered co-occurring.
- Example: "AI" and "innovation" co-occur within the same sentence. This suggests a semantic link.

Collocations

- A **collocation** is a pair or group of words that tend to occur together more often than expected by chance.
- They form a kind of "natural combination" in a language.
- Example: We say "strong tea", not "powerful tea".
- Example: We say "make a decision", not "do a decision".

The key difference between collocation and co-occurrence is:

- Collocations = statistically strong word pairs (like "customer satisfaction").
- Co-occurrences = any words appearing together (like "AI" and "innovation" in the same sentence).

- Collocations are detected using association measures:
  - **Pointwise Mutual Information (PMI):** This measures how much more often x and y occur together than if they were independent. Mathematically,

    $$\boxed{PMI(x,y) = \log\left[\frac{P(x,y)}{P(x)\cdot P(y)}\right]}$$

    where $P(x,y)$ is the joint probability distribution, and $P(x)$ and $P(y)$ are the probability distributions of x and y, respectively.
  - **Chi-Square Test (χ^2):** This tests whether co-occurrence frequency is significantly higher than random chance. Mathematically,

    $$\boxed{t(x,y) = \frac{O-E}{\sqrt{O}}}$$

    where O and E are the observed and expected co-occurrences, respectively.

Word Clouds, Bar Charts, and Co-occurrence Networks

- **Word clouds** give a visual representation of word frequency. Words that appear more often are displayed larger and/or bolder. They are used for exploratory data analysis (EDA) and summarization.
- **Bar charts** A quantitative visualization of word counts, term frequencies, or TF-IDF scores (which we will see later on). Unlike word clouds, they show exact values and comparisons. They are used for the top k-terms, sentiment terms, and for n-grams.
- **Co-occurrence networks** are graphs where the nodes are the words, and the edges are the co-occurrences (how often the words appear together). The edges can be weighted by frequency or PMI scores. They are used for relationship discovery, cluster analysis, and trend analysis.

Lab 4

In this lab session, we will use all that we learned in exploratory text mining and apply it to **real-world business problems**.

Module 5: Applied Low-Tech NLP

Rules-Based Sentiment Analysis

- **Rules-based sentiment analysis** is an approach that uses manually defined linguistic rules, lexicons, and heuristics (rather than ML) to determine whether a text expresses a positive, negative, or neutral sentiment.
- Instead of training on large datasets, we encode human knowledge into:
    - Sentiment lexicons (dictionaries of words with positive/negative scores).
    - Rules about negation, intensifiers, and context.
    - Simple scoring functions.

The Core Components of Rules-Based Sentiment Analysis

- A list of words associated with sentiment polarity (positive/negative).
- A scoring mechanism that sums up the polarity scores of words in a text. Mathematically, we sum up all the sentiments of the words w in the particular token T

$$\boxed{\text{Sentiment Score} = S = \sum_{w \in T} S(w)}$$

- **Negation Handling:** Rules to flip polarity when negation words appear. Example: If the word "not" precedes a positive word, multiply score by -1.
- **Intensifiers and Diminishers:** Words that amplify (very, extremely) or weaken (slightly, somewhat) sentiment. Example: "Very good" gets a score of $+4$ as opposed to $+2$.

The Pros and Cons of Rules-Based Sentiment Analysis

- **Advantages**
  - Transparent (easy to explain why a decision was made).
  - Works well with small datasets (no training required).
  - Easy to customize for specific industries.
- **Disadvantages**
  - Struggles with sarcasm ("Great service. . . not!").
  - Hard to scale to very large vocabularies.
  - Needs manual updates to lexicons and rules.

Python libraries like VADER (**V**alence **A**ware **D**ictionary for s**E**ntiment **R**easoning) implement these rules and are widely used for social media, reviews, and business analytics.

Lexicons

- A **lexicon** is a structured list of words used by NLP systems to understand and process text.
- The contents of a lexicon are:
    - Words or phrases (unigrams, bigrams, etc.)
    - Part-of-speech tags (nouns, verbs, adjectives, etc.)
    - Morphological information (plurals, tenses, lemmas, etc.)
    - Semantic information (meanings, synonyms, antonyms, etc.)
    - Sentiment scores (positive, negative, neutral)

Types of Lexicons

- **General-Purpose Lexicons:** WordNet, Oxford Dictionary, etc.
- **Domain-Specific Lexicons:** Financial, medical, or business lexicons tailored for specialized vocabulary.
- **Sentiment Lexicons:** Lists of words with associated sentiment polarity. For example, “happy” $\implies$ positive, “bad” $\implies$ negative.

Applications of Lexicons in Business

- Using a sentiment lexicon, analyze customer reviews or Tweets. For example, the statement: "The delivery was fast but the product is terrible", can be used to match words with positive/negative lexicon scores.
- Used for keyword matching with a focus on business-specific lexicon to detect mentions of products, services, or competitors.
- Used to assign categories to emails, documents, or complaints based on the presence of lexicon terms.

A Real-World Application of Lexicons: Customer Review Analysis

Scenario: Data Scientists at your organization construct the following sentiment lexicon:

Word	**Sentiment: S**
"excellent"	$+1$
"good"	$+1$
"bad"	-1
"terrible"	$+1$
"fast"	$+1$

Task: If a customer posted: "The delivery was fast but the product is terrible", what can be concluded about this review?

- The total sentiment is $S_{\text{tot}} = S(\text{"fast"}) + S(\text{"terrible"}) = (+1) + (-1) = 0$. Thus, we conclude that we have a mixed review.

Bag of Words (BoW) Representation

- The **Bag of Words** (BoW) model is a text representation technique where a piece of text (sentence, paragraph, document) is represented as a collection of its words, ignoring grammar, word order, and context, but keeping word frequency.
- It is one of the most fundamental feature extraction methods in NLP.
- Let $V = \{w_1, w_2, \ldots, w_n\}$ be a vocabulary of size n. Suppose we want to document a sequence of d words. Then, the representation vector is

$$\boxed{\mathbf{x}_d = [f(w_1, d), f(w_2, d), \ldots, f(w_n, d)]}$$

where $f(w_i, d)$ is the frequency of word w_i in document d.

Example of BoW

Scenario : Suppose that your company wants to analyze customer reviews.

- **Review 1:** "The product is excellent".
- **Review 2:** "The service is excellent".

The vocabulary is $V = \{\text{the, product, is, excellent, service}\}$.
Task: Construct the representation vectors.

The representation vectors are:

$$\mathbf{x}_{d_1} = [1, 1, 1, 1, 0], \quad \mathbf{x}_{d_2} = [1, 0, 1, 1, 1].$$

Now these vectors can be fed into ML models for sentiment analysis, clustering, or recommendation systems.

Pros and Cons of BoW

- **Advantages**
  - Simple and fast to implement.
  - Good for traditional ML algorithms (Naïve Bayes, SVM, Logistic Regression).
- **Disadvantages**
  - Ignores word order. For example, “not good” and “good not” look the same.
  - High dimensionality, large vocabulary implies long vectors.
  - No semantic understanding. For example, “excellent” $\neq$ “great”.

BoW is like a shopping list of words – It only cares about what words are present and how many times, not how they are arranged.

Cosine Similarity

- **Cosine similarity** is a metric used to measure how similar two vectors are, by calculating the cosine of the angle between them.
- In NLP, once text is represented as vectors (via Bag of Words, TF-IDF, or embeddings), cosine similarity tells us how close in meaning two texts are.
- Mathematically, given two vectors $\mathbf{A}$ and $\mathbf{B}$, the cosine similarity between them is given by

$$\boxed{\cos\theta = \frac{\mathbf{A}\cdot\mathbf{B}}{||\mathbf{A}||\ ||\mathbf{B}||} = \frac{\sum_i A_i B_i}{\left(\sqrt{\sum_i A_i^2}\right)\left(\sqrt{\sum_i B_i^2}\right)}}$$

- Interpretation:
  - If $\cos\theta = 1$, you have identical direction (perfect similarity).
  - If $\cos\theta = 0$, you have orthogonality (no similarity).
  - If $\cos\theta = -1$, you have opposite directionality (rare in text, since frequencies are non-negative).

Example of Cosine Similarity

Given the corpus composed of two documents:

- **Document 1:** "I love data science".
- **Document 2:** "I love machine learning".

The vocabulary is

$$V = \{\text{I, love, data, science, machine, learning}\}.$$

Task: Calculate the cosine similarity between the two documents.

- The representation vectors for each document are

$$\mathbf{A} = [1, 1, 1, 1, 0, 0], \quad \mathbf{B} = [1, 1, 0, 0, 1, 1].$$

- The numerator in the cosine similarity equation is

$$\mathbf{A} \cdot \mathbf{B} = (1)(1) + (1)(1) + (1)(0) + (1)(0) + (0)(1) + (0)(1) = 2.$$

- The magnitudes of the representation vectors are

$$||\mathbf{A}|| = \sqrt{(1)^2 + (1)^2 + (1)^2 + (1)^2 + (0)^2 + (0)^2} = 2,$$
$$||\mathbf{B}|| = \sqrt{(1)^2 + (1)^2 + (0)^2 + (0)^2 + (1)^2 + (1)^2} = 2.$$

- Thus, the cosine similarity is

$$\cos\theta = \frac{2}{2 \times 2} = 0.5.$$

- This means that document 1 and document 2 are 50% similar.

Why is Cosine Similarity Used in NLP?

- Unlike Euclidean distance, cosine similarity is scale-invariant (it does not matter if one document is much longer than another).
- It is excellent for comparing text in:
  - Document clustering (e.g., grouping news articles).
  - Recommendation systems (e.g., finding similar products from descriptions).
  - Plagiarism detection.
  - Semantic search engines.

(TF-IDF)

- **TF-IDF** is a numerical statistic used to reflect how important a word is in a document relative to a collection of documents (corpus).
- **TF (Term Frequency):** Measures how often a word appears in a document.
- **IDF (Inverse Document Frequency):** Reduces the weight of common words and increases the weight of rare but important words.
- Together, TF-IDF balances local importance (within a document) and global importance (across the corpus).

How is TF-IDF Calculated?

- Recall, the term frequency (TF) was calculated as

$$\boxed{TF(t,d) = \frac{f_{t,d}}{\sum_k f_{k,d}}}$$

  where $f_{t,d}$ is the raw count of term t in document d, and the denominator captures the total number of terms in document d.

- The inverse document frequency (IDF) term is calculated as

$$\boxed{IDF(t,D) = \log\left(\frac{N}{1 + |\{d \in D | t \in d\}|}\right)}$$

  where N is the total number of documents, and the denominator contains the number of documents containing t terms.

- Together, these form the TF-IDF, which is given by

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

Example TF-IDF Calculation

Scenario: Suppose that the corpus is composed of three documents:

- **Document 1:** “I love data science”.
- **Document 2:** “I love machine learning”.
- **Document 3:** “Data science and machine learning’.

Task: Calculate TF-IDF for the word “data” in document 1.

- The TF is

$$TF(\text{“data”, document 1}) = \frac{1}{4} = 0.25.$$

- Now, $N = 3$ (3 documents), and so we have that

$$IDF(\text{“data”}) = \log\left(\frac{3}{1+1}\right) = \log(3/2) \approx 0.4055.$$

- Thus, the TF-IDF score is

$$TF - IDF = 0.25 \times 0.4055 = 0.1014.$$

Why Use TF-IDF?

- Solves Bag of Words issue: Instead of treating all words equally, it gives more importance to unique terms.
- Removes bias of frequent words: Words like “the”, “is”, “and” do not dominate.
- Helps in search/recommendations: Matches rare but meaningful words better.

Lab 5

To apply the concepts learned in Applied Low-Tech NLP to solve real-world business problems.

Additional Material

(Contingent on Time)

Ethical Considerations in NLP

- Use balanced, representative datasets to reduce bias.
- Remove personal identifiers and follow data privacy laws (e.g., GDPR).
- Ensure transparency and explainability of model decisions.
- Prevent misuse for misinformation, spam, or harmful content.
- Support low-resource languages to promote inclusivity.
- Focus on AI augmentation, not full replacement of human jobs.
- Protect models against adversarial attacks and vulnerabilities.
- Provide model cards and dataset documentation for accountability.
- Keep a human-in-the-loop for critical applications.
- Continuously monitor, audit, and update deployed NLP systems.

NLP best Practices in Business

- **Define clear objectives:** Link NLP tasks (e.g., sentiment analysis, chatbots) to measurable business goals.
- **Curate high-quality data:** Ensure datasets are domain-specific, clean, and representative.
- **Preprocess consistently:** Apply normalization, stopword removal, and tokenization suited to your use case.
- **Choose the right model complexity:** Do not overuse deep models if simpler ones meet business needs.
- **Balance accuracy and interpretability:** Use explainable models for decisions that impact customers.

- **Automate but validate:** Automate workflows (customer support, text classification) but keep human oversight.
- **Account for multilingualism:** Support multiple languages if your business operates globally.
- **Respect privacy and compliance:** Handle customer data ethically and in line with GDPR/CCPA.
- **Monitor performance:** Track drift, accuracy, and customer satisfaction over time.
- **Assess ROI:** Evaluate cost savings, revenue impact, and efficiency improvements from NLP adoption.

Thank You
FOR YOUR ATTENTION
Let's Connect