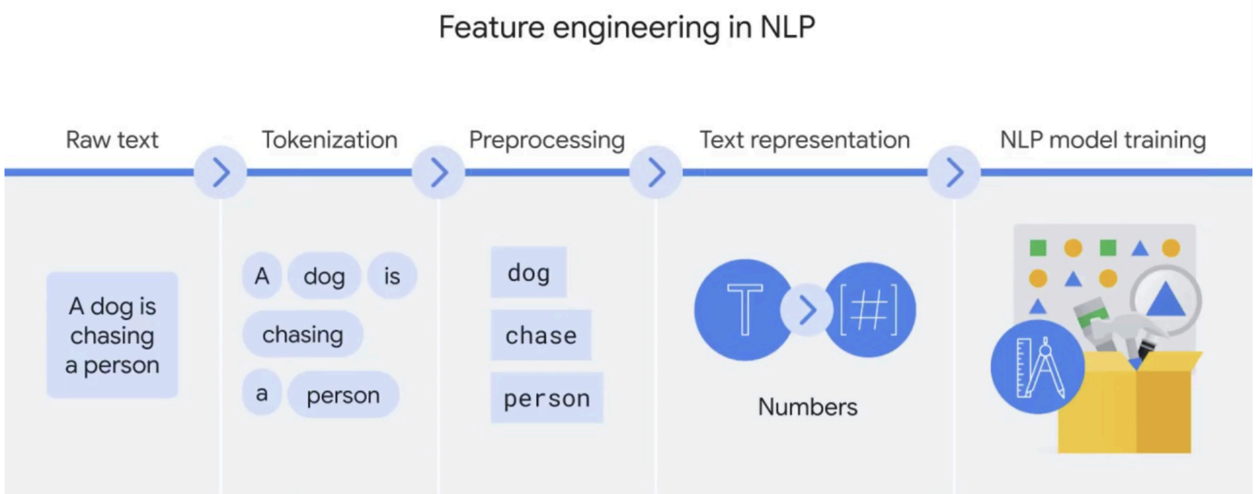
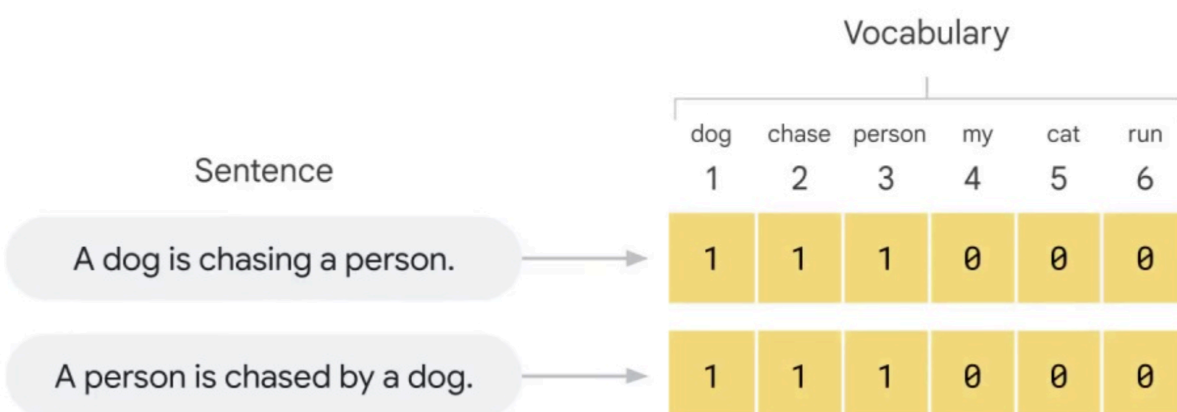


https://www.cloudskillsboost.google/course_templates/40/video/514899?locale=pl

https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/



Bag-of-words does not consider the order of the words and that is why it's called a "bag" of words.

Sample Reviews:

1. **Review 1:** "Phone is good"
2. **Review 2:** "Phone is very good and performance is good"
3. **Review 3:** "Performance is not good"

Word	Index
phone	1
is	2
good	3
very	4
and	5
performance	6
not	7

Total Vocabulary Size: 7

One-Hot Encoding represents each word as a binary vector where only one position (corresponding to the word index) is **1**, and all other positions are **0**.

Encoded Reviews using One-Hot Encoding:

- **Review 1: "Phone is good"**
  - phone → [1, 0, 0, 0, 0, 0, 0]
  - is → [0, 1, 0, 0, 0, 0, 0]
  - good → [0, 0, 1, 0, 0, 0, 0]

- **Review 2: "Phone is very good and performance is good"**
  - phone → [1, 0, 0, 0, 0, 0, 0]
  - is → [0, 1, 0, 0, 0, 0, 0]
  - very → [0, 0, 0, 1, 0, 0, 0]
  - good → [0, 0, 1, 0, 0, 0, 0]
  - and → [0, 0, 0, 0, 1, 0, 0]
  - performance → [0, 0, 0, 0, 0, 1, 0]
  - is → [0, 1, 0, 0, 0, 0, 0]
  - good → [0, 0, 1, 0, 0, 0, 0]
- **Review 3: "Performance is not good"**
  - performance → [0, 0, 0, 0, 0, 1, 0]
  - is → [0, 1, 0, 0, 0, 0, 0]
  - not → [0, 0, 0, 0, 0, 0, 1]
  - good → [0, 0, 1, 0, 0, 0, 0]

Observation:

- One-Hot Encoding does **not capture the frequency** of words, only their **presence**.
- If a word appears multiple times (like "good" in Review 2), it still gets encoded **individually** each time.

Bag-of-Words (BoW) Encoding:

BoW represents the **frequency** of each word in the entire sentence.

Encoded Reviews using Bag-of-Words:

Word	Review 1	Review 2	Review 3
phone	1	1	0
is	1	2	1
good	1	2	1
very	0	1	0
and	0	1	0
performance	0	1	1
not	0	0	1

Limitations to Consider:

- **Lack of Context:** BoW does not consider the **order of words**, so sentences like "Dog bites man" and "Man bites dog" have the same representation.
- **High Dimensionality:** For large vocabularies, the vector size becomes massive, leading to sparsity issues.
- **No Semantic Understanding:** Words like **"good"** and **"excellent"** are treated as completely unrelated, even though they have similar meanings.

When to Use BoW:

- When **word frequency** is more important than **context or order**.
- For tasks like **text classification**, **spam filtering**, or **document clustering**.
- When simplicity and computational efficiency are crucial.
- When you have a relatively **small vocabulary size**.

TF- IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure used to determine the importance of a word to a document in a collection of documents (corpus). In e-commerce product search, platforms like Amazon and Flipkart utilize TF-IDF to understand the relevance of search queries to product listings. Here's how it's typically stored and used:

Storage:

1. **Indexing:** E-commerce platforms create an index of all the words present in their product descriptions, titles, and other relevant text fields.
2. **TF-IDF Calculation:** For each word in the index, the platform calculates its TF-IDF score for every product.
    - **Term Frequency (TF):** Measures how often a word appears in a specific product's text.
    - **Inverse Document Frequency (IDF):** Measures how rare a word is across all product listings. Common words like "the" or "is" have low IDF, while specific terms have high IDF.
3. **Inverted Index:** The TF-IDF scores are stored in an inverted index. This data structure maps each word to the products it appears in, along with its corresponding TF-IDF score. This allows for efficient retrieval of products based on search terms.
4. Remove stopwords from the listings.
5. Recalculate TF-IDF step-by-step with the filtered words.
6. Show storage and ranking for the query "laptop."

Original Listings and Stopwords

- **Product 1 (P1):** "Powerful laptop with fast processor and large storage." (Rating: 4/5)
- **Product 2 (P2):** "Affordable laptop for students, great for note-taking." (Rating: 3/5)
- **Product 3 (P3):** "High-quality digital camera with 20MP sensor and zoom lens." (Rating: 5/5)

Common Stopwords: with, and, for, is

Step 1: Pre-Processing (Remove Stopwords)

P1: "Powerful laptop with fast processor and large storage"

- Remove: with, and
- Filtered: powerful, laptop, fast, processor, large, storage
- Total Words: 6

P2: "Affordable laptop for students, great for note-taking"

- Remove: for, for (appears twice)
- Filtered: affordable, laptop, students, great, note-taking

- Total Words: 5

P3: "High-quality digital camera with 20MP sensor and zoom lens"

- Remove: with, and
- Filtered: high-quality, digital, camera, 20mp, sensor, zoom, lens
- Total Words: 7

Corpus: 3 products, 18 total words after stopword removal.

Unique Words: powerful, laptop, fast, processor, large, storage, affordable, students, great, note-taking, high-quality, digital, camera, 20mp, sensor, zoom, lens

Step 2: Calculate Term Frequency (TF)

TF = Number of times a term appears in a document / Total words in that document (after stopword removal).

For "laptop":

- **TF(laptop, P1):** 1 / 6 ≈ 0.1667
- **TF(laptop, P2):** 1 / 5 = 0.2
- **TF(laptop, P3):** 0 / 7 = 0.000

Full TF for All Words (for completeness):

- **P1:**
  - powerful: 1/6 ≈ 0.1667
  - laptop: 1/6 ≈ 0.1667
  - fast: 1/6 ≈ 0.1667
  - processor: 1/6 ≈ 0.1667
  - large: 1/6 ≈ 0.1667
  - storage: 1/6 ≈ 0.1667
- **P2:**
  - affordable: 1/5 = 0.2
  - laptop: 1/5 = 0.2
  - students: 1/5 = 0.2
  - great: 1/5 = 0.2
  - note-taking: 1/5 = 0.2
- **P3:**
  - high-quality: 1/7 ≈ 0.1429
  - digital: 1/7 ≈ 0.1429
  - camera: 1/7 ≈ 0.1429
  - 20mp: 1/7 ≈ 0.1429
  - sensor: 1/7 ≈ 0.1429
  - zoom: 1/7 ≈ 0.1429

- lens: 1/7 ≈ 0.1429

Step 3: Calculate Inverse Document Frequency (IDF)

IDF = log(N / df), where N = total documents (3), df = documents with the term. Using natural log (ln) as in your example.

IDF(laptop):

- Documents with "laptop": 2 (P1, P2)
- IDF = ln(3 / 2) = ln(1.5) ≈ 0.4055

Full IDF for All Words:

- powerful: ln(3/1) ≈ 1.0986 (P1)
- laptop: ln(3/2) ≈ 0.4055 (P1, P2)
- fast: ln(3/1) ≈ 1.0986 (P1)
- processor: ln(3/1) ≈ 1.0986 (P1)
- large: ln(3/1) ≈ 1.0986 (P1)
- storage: ln(3/1) ≈ 1.0986 (P1)
- affordable: ln(3/1) ≈ 1.0986 (P2)
- students: ln(3/1) ≈ 1.0986 (P2)
- great: ln(3/1) ≈ 1.0986 (P2)
- note-taking: ln(3/1) ≈ 1.0986 (P2)
- high-quality: ln(3/1) ≈ 1.0986 (P3)
- digital: ln(3/1) ≈ 1.0986 (P3)
- camera: ln(3/1) ≈ 1.0986 (P3)
- 20mp: ln(3/1) ≈ 1.0986 (P3)
- sensor: ln(3/1) ≈ 1.0986 (P3)
- zoom: ln(3/1) ≈ 1.0986 (P3)
- lens: ln(3/1) ≈ 1.0986 (P3)

Step 4: Calculate TF-IDF

TF-IDF = TF × IDF

For "laptop":

- **TF-IDF(laptop, P1):** 0.1667 × 0.4055 ≈ 0.0676
- **TF-IDF(laptop, P2):** 0.2 × 0.4055 ≈ 0.0811
- **TF-IDF(laptop, P3):** 0 × 0.4055 = 0.000

Full TF-IDF for All Words:

- **P1:**
  - powerful: 0.1667 × 1.0986 ≈ 0.1831
  - laptop: 0.1667 × 0.4055 ≈ 0.0676
  - fast: 0.1667 × 1.0986 ≈ 0.1831
  - processor: 0.1667 × 1.0986 ≈ 0.1831
  - large: 0.1667 × 1.0986 ≈ 0.1831
  - storage: 0.1667 × 1.0986 ≈ 0.1831
- **P2:**
  - affordable: 0.2 × 1.0986 ≈ 0.2197
  - laptop: 0.2 × 0.4055 ≈ 0.0811
  - students: 0.2 × 1.0986 ≈ 0.2197
  - great: 0.2 × 1.0986 ≈ 0.2197
  - note-taking: 0.2 × 1.0986 ≈ 0.2197
- **P3:**
  - high-quality: 0.1429 × 1.0986 ≈ 0.1570
  - digital: 0.1429 × 1.0986 ≈ 0.1570
  - camera: 0.1429 × 1.0986 ≈ 0.1570
  - 20mp: 0.1429 × 1.0986 ≈ 0.1570
  - sensor: 0.1429 × 1.0986 ≈ 0.1570
  - zoom: 0.1429 × 1.0986 ≈ 0.1570
  - lens: 0.1429 × 1.0986 ≈ 0.1570

Step 5: Store in Memory (Inverted Index)

Analogy: Like a catalog—each word has a card listing products and their TF-IDF scores.

Inverted Index (Sparse, zeros omitted):

- powerful: {P1: 0.1831}
- laptop: {P1: 0.0676, P2: 0.0811}
- fast: {P1: 0.1831}
- processor: {P1: 0.1831}
- large: {P1: 0.1831}
- storage: {P1: 0.1831}
- affordable: {P2: 0.2197}
- students: {P2: 0.2197}
- great: {P2: 0.2197}
- note-taking: {P2: 0.2197}
- high-quality: {P3: 0.1570}
- digital: {P3: 0.1570}
- camera: {P3: 0.1570}

- 20mp: {P3: 0.1570}
- sensor: {P3: 0.1570}
- zoom: {P3: 0.1570}
- lens: {P3: 0.1570}

Memory Format: Stored in a fast database (e.g., Elasticsearch) as a hash table—each word points to (product ID, TF-IDF score) pairs.

Step 6: Ranking and Displaying Results for Query "laptop"

- **Query:** "laptop"
- **TF-IDF Lookup:**
    1. laptop: {P1: 0.0676, P2: 0.0811}
- **Initial TF-IDF Rank:**
    1. Product 2: 0.0811
    2. Product 1: 0.0676
    3. Product 3: 0.000
- **Adjust with Ratings (Real-World Tiebreaker):**
    1. Multiply TF-IDF by (Rating/5):
        - P2: 0.0811 × (3/5) = 0.0811 × 0.6 ≈ 0.0487
        - P1: 0.0676 × (4/5) = 0.0676 × 0.8 ≈ 0.0541
        - P3: 0 × (5/5) = 0
    2. **Final Rank:**
        - Product 1: 0.0541 (Rating: 4/5)
        - Product 2: 0.0487 (Rating: 3/5)
        - Product 3: 0.000 (Rating: 5/5)
- **Search Results:**
    1. **"Powerful laptop with fast processor and large storage" (Rating: 4/5)**
        - Why: Solid TF-IDF (0.0676) and higher rating (4/5)—top pick.
    2. **"Affordable laptop for students, great for note-taking" (Rating: 3/5)**
        - Why: Higher TF-IDF (0.0811) but lower rating (3/5) drops it.
    3. **"High-quality digital camera with 20MP sensor and zoom lens" (Rating: 5/5)**
        - Why: No "laptop"—irrelevant despite 5/5 rating.