# Exploring Text Representations

Week #2 - 18 Sept '24

NLP Lab

# NLP Lab Project : Build your own GPT

- Text Processing and Bag of Words Model
- N-grams and TF-IDF
- Theory of Word Embeddings : Word2Vec, GloVe
- RNNs: LSTM, GRU; Attention Mechanisms
- Transformer Model, Advanced Transformers: GPT, BERT
- Fine-Tuning, SFT, Continued Pre-training
- Custom Language Model using Prompt Engineering, RAG

# 1. Bag of Words (BoW) Model

# 1.1 Introduction to Bag of Words (BoW)

**Definition:** A BoW model represents text as a collection of word frequencies or occurrences without considering grammar and word order.

**How it works:**

- **Vocabulary Creation:** Create a list of all unique words (vocabulary) from the text corpus.
- **Vectorization:** Convert text documents into vectors of word counts or binary values (presence/absence).

# 1.1 Introduction to Bag of Words (BoW)

**Example:**

**Documents:**

- "NLP is fun"
- "I love NLP"

**Vocabulary:** ["NLP", "is", "fun", "I", "love"]

**BoW representation:**

- Doc 1: [1, 1, 1, 0, 0]
- Doc 2: [1, 0, 0, 1, 1]

# 1.2 Advantages and Limitations of BoW

### Advantages

- Simple and easy to implement.
- Works well with smaller datasets.
- Effective for text classification tasks.

### Limitations

- Ignores the order and semantics of words.
- Can lead to a sparse matrix for large vocabularies.
- Does not handle synonyms well.
- Sensitive to the presence of stopwords and noisy data.

# 1.3 BoW Model Practical Applications

**Text Classification:** Use BoW to convert text into numerical format for machine learning models (e.g., spam detection).

**Sentiment Analysis:** Analyze sentiment of reviews or social media posts using BoW and text preprocessing techniques.

**Information Retrieval:** Create search algorithms that match queries to documents based on word frequencies.

**Recommendation Systems:** Build simple recommendation systems based on text similarity using BoW vectors.

# 2. Term Frequency - Inverse Document Frequency

# 2.1 TF-IDF : Weighing words for Text Analysis

**Term Frequency (TF) :** Measures how often a term appears in a document.

**Inverse Document Frequency (IDF) :** Weights the term based on its rarity across all documents.

**TF-IDF(t,d) = TF(t,d) * IDF(t)**

**Helps prioritize important, unique words in documents while downplaying common terms.**

# 2.2 Term Frequency

$$\mathrm{TF}(t, d) = \frac{\text{Frequency of } t \text{ in } d}{\text{Total words in } d}$$

**Example :**

| Documents | Text | Total number of words in a document |
|---|---|---|
| A | Jupiter is the largest planet | 5 |
| B | Mars is the fourth planet from the sun | 8 |

| Words | TF (for A) | TF (for B) |
|---|---|---|
| Jupiter | 1/5 | 0 |
| Is | 1/5 | 1/8 |
| The | 1/5 | 2/8 |
| largest | 1/5 | 0 |
| Planet | 1/5 | 1/8 |
| Mars | 0 | 1/8 |
| Fourth | 0 | 1/8 |
| From | 0 | 1/8 |
| Sun | 0 | 1/8 |

Source Images : https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d

# 2.3 Inverse Document Frequency

$$\mathrm{IDF}(t) = \log\left(\frac{\text{Total Documents}}{1 + \text{Number of Documents containing } t}\right)$$

**Example :**

| Documents | Text | Total number of words in a document |
|---|---|---|
| A | Jupiter is the largest planet | 5 |
| B | Mars is the fourth planet from the sun | 8 |

| Words | TF (for A) | TF (for B) | IDF |
|---|---|---|---|
| Jupiter | 1/5 | 0 | ln(2/1) = 0.69 |
| Is | 1/5 | 1/8 | ln(2/2) = 0 |
| The | 1/5 | 2/8 | ln(2/2) = 0 |
| largest | 1/5 | 0 | ln(2/1) = 0.69 |
| Planet | 1/5 | 1/8 | ln(2/2) = 0 |
| Mars | 0 | 1/8 | ln(2/1) = 0.69 |
| Fourth | 0 | 1/8 | ln(2/1) = 0.69 |
| From | 0 | 1/8 | ln(2/1) = 0.69 |
| Sun | 0 | 1/8 | ln(2/1) = 0.69 |

Source Images : https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d

# 2.4 Advantages and Limitations of TF-IDF

## Advantages

- Weighs words based on importance, discounts common words.
- Improves Search and Retrieval Relevance.

## Limitations

- Context Ignorance
- Synonym Problem
- Cannot Handle Complex Relationships

# 3. N-Grams (Language Models)

# 3.1 What are N-grams in context of Text Representation?

**Definition:** An n-gram is a contiguous sequence of 'n' items (words or characters) from a given text or speech.

- Unigram (n = 1): Single words (e.g., "machine")
- Bigram (n = 2): Two-word sequences (e.g., "machine learning")
- Trigram (n = 3): Three-word sequences (e.g., "machine learning models")

**Text Representation Intuition:** We calculate the frequency of groups of words.

**Advantage:** Captures Semantic Meaning.

**Disadvantage:** Out of Vocabulary Words.

# 3.2 What are Language Models?

**Definition:** Models that assign a probability to each possible next word. Language models can also assign a probability to an entire sentence, telling us that the following sequence has a much higher probability of appearing in a text.

Goal : Compute the probability of a sentence or sequence of words.

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

Related Task : Probability of an upcoming word.

$$P(w_5 | w_1,w_2,w_3,w_4)$$

# 3.3 How to compute P(W)?

- How to compute the joint probability :

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

- Example - P(its, water, is, so, transparent, that) ?

- **Intuition: Chain Rule of Probability**

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

- P("its water is so transparent") = P(its) × P(water|its) × P(is|its water) × P(so|its water is) × P(transparent|its water is so) x P(that|its water is so transparent)

# 3.4 How to estimate these probabilities?

$$P(\texttt{blue}|\texttt{The water of Walden Pond is so beautifully})$$

$$=$$

$$\frac{C(\texttt{The water of Walden Pond is so beautifully blue})}{C(\texttt{The water of Walden Pond is so beautifully})}$$

Can we just count and divide?

- No! Too many possible sentences.
- We will never see enough data to estimate these.

Solution: Markov Assumption!

- We can approximate the history by just the last few words

# 3.5 What are N-Grams Language Models?

**Definition:** We also (in a bit of terminological ambiguity) use the word ‘n-gram’ to mean a probabilistic model that can estimate the probability of a word given the n-1 previous words, and thereby also to assign probabilities to entire sequences.

**Unigram Model:**

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

**Bigram Model:**

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

# 3.6 Key Points about N-Gram Language Model

- In general this is an insufficient model of language because language has long-distance dependencies.

  Example : “The computer(s) which I had just put into the machine room on the fifth floor is(are) crashing.”
- We can still get away with N-Grams as they :
  - Capture Local Context
  - Simple Intuitive
  - Computationally Efficient
- We do everything in log space :
  - Avoid underflow
  - Also adding is faster than multiplying
- As 'n' increases, the model needs a large amount of training data to cover all possible word combinations. Also, they don't capture the meaning or relationships beyond word proximity.

# Kahoot Time!

# Group Picture!