# Text-Searching with Sentence Transformers and NLP

Presented By: Tristan Pank

# Sign In

https://forms.gle/TgrDFoLKEgydwHmD8

# Agenda

1. Presentation
    a. Overview of text-searching
    b. How to utilize sentence-transformers for searching
    c. Optimizing the algorithm with NLP
2. Hands On (Google Colab)
    a. Work on implementing the sentence transformer algorithm
    b. Build a class and use it to search through text
    c. Fine-Tune the sentence transformer on additional data (Time Permitting)

# Why do we need efficient/effective text searching

- With larger datasets and resources, it becomes harder to effectively find information.
- We want to rely on text searching algorithms to better index and retrieve information from these sources.
- Examples:
  - Searching for a specific section of a textbook
  - Matching articles to a query
  - Finding regions of a transcript

# Traditional Algorithms: Inverted Index

- Count the presence of each word in the query to each document.
- Return the document that matches the most number of words.

Query: “How to study machine learning?”

| Documents | Matched Keywords | Score |
|---|---|---|
| “How to study biology” | How, to, study | 3 |
| “Guide for machine learning” | machine learning | 2 |

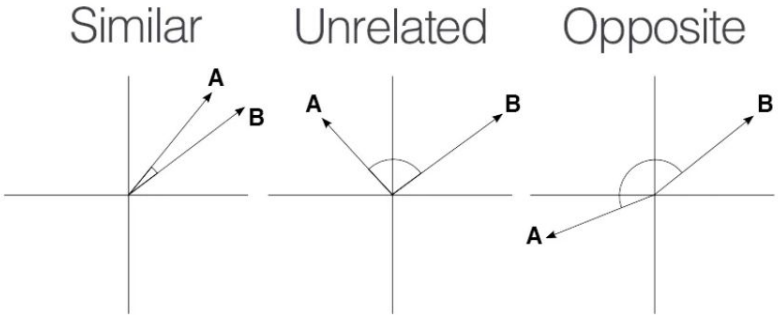
# Searching Based on Meaning (Sentences as Vectors)

- Instead of searching for a query based on keywords, we want to search based on meaning, and find the “most similar” sentence to it.
- We need an effective way to compare similarity of two sentences:
  - Vector Encoding
  - Encode sentences in a way that captures their meaning as a direction in the vector space.
  - Sentences that are “similar” will be better aligned than other sentence vectors.
  - We measure this direction using cosine similarity, which is defined from the dot product

$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}},$$

# Using Sentence Transformers to Encode Sentences

- The easiest way for us to convert sentences into vectors is to utilize a pre-trained sentence transformer.
- This will parse each sentence and based on its words and their neighbors (aka context) it'll convert the sentence into a vector of dimension 384.
- To find the most similar sentence to a query, we simply calculate the cosine similarity between the query and each sentence and return the sentence with the largest similarity.



# Using NLP to optimize the encoding

- Removing Stopwords:
  - Commonly used words that don’t carry much meaning
  - Is, a, the, etc.
- Lemmatizing the text:
  - Takes words and returns them to their root form
  - Maintains consistency among words for better encoding
  - running -> run
- Utilize both techniques on the sentences prior to sentence encoding for better results.