Instructions:

# Run on Colab:

1. Upload the “Solution.ipynb” file to colab, or open on GitHub and click 
2. Upload “J. K. Rowling - Harry Potter 1 - Sorcerer's Stone.txt” file to colab: A screenshot of a computer

   Description automatically generated
3. Click *runtime -> run all*
4. (Optional) you can use a GPU to speed up the process *using runtime -> change runtime type*.

# Run locally:

1. Clone the repository using “

git clone <https://github.com/maor63/PassageRetrievalTechniques.git>

cd PassageRetrievalTechniques

1. Create python virtual environment:

python -m venv venv

1. Activate the virtual environment:
   1. On Windows:

venv\Scripts\activate

* 1. On Linux/Mac:

source venv/bin/activate

1. Install the required dependencies:

pip install -r requirements.txt

1. Open jupyter notebook:

jupyter notebook

1. Open “Solution.ipynb” and *run -> run all cells*

# **Discussion:**

\*The full discussion also appears in the notebook.

\*The **interactive retrieval system** is under the UI section.

## **Issues with TF-IDF**

* Exact word matches are required; synonyms or related terms won't work.
* No understanding of semantic relationships.

## **Semantic Search**

I choose to use the [multi-qa-MiniLM-L6-cos-v1](https://www.google.com/url?q=https%3A%2F%2Fhuggingface.co%2Fsentence-transformers%2Fmulti-qa-MiniLM-L6-cos-v1) model since it is the best preforming MiniLM model for semantic search. [Sentence transformer performance leaderboard](https://www.google.com/url?q=https%3A%2F%2Fwww.sbert.net%2Fdocs%2Fsentence_transformer%2Fpretrained_models.html)

# **Evaluation**

I assume that semantic search is ground truth. I use the following metrics to evaluate the TF-IDF vectors performance:

### **P@K**

Measure the precision at the top K results. This measure help us understands how many relevant result are at the top of the retrieval. High precision means TF-IDF retrieves mostly relevant results.

### **Spearman’s Rank Correlation**

Measure how similar TF-IDF ranking score is to the MiniLM ranking scores. High values means that both scores behave similarly.

### **Normalized Discounted Cumulative Gain (NDCG@k)**

This measures how well TF-IDF ranked the top@K results compared to MiniLM both in relevance and in order. High NDCG indicates that TF-IDF ranks the relevant passages well.

**Conclusion:**  
Our analysis shows that while TF-IDF ranks the top results in a reasonable order (as indicated by a high NDCG@k), it struggles to place the most relevant result at the very top (resulting in a low P@k). Additionally, TF-IDF ranks results only slightly similarly to MiniLM, as reflected by the low positive correlation between their rankings.

# **Open Qs**

**Q1: Discuss potential improvements for retrieval using an LLM. Clarify whether this approach applies to lexical, semantic, or both searches.**

Potential improvements for retrieval using an LLM include:

* For Both Searches:
  1. Use an LLM to expand queries by generating synonyms, related terms, or paraphrased versions. Calculate the average similarity scores of the reformulated queries against the results to identify the most relevant passages.
  2. Ask an LLM to generate relevance scores directly for each result, enhancing the ranking precision by incorporating semantic and contextual understanding.
     + Apply a natural language inference (NLI) model to filter irrelevant results. An NLI model evaluates the relationship between a **premise** and a **hypothesis. I u**se the NLI model to determine whether the query (hypothesis) is entailed by the result (premise).
* For Semantic Search:
  1. Fine-tune the word embeddings on the Harry Potter domain or a similar dataset. This would replace the general-purpose MiniLM embeddings with ones more attuned to the narrative style and vocabulary of the target text.

**Q2: Highlight the drawbacks or limitations of your suggestion.**

* LLMs are computationally expensive
* Post-analysis steps, such as query expansion, relevance scoring, or applying NLI filtering, increasing the system's response time.
* LLM and NLI can make mistakes, and it important to evaluate them.
* Finetuning a word embeddings on a specific domain will probably reduce the embeddings ability to generalize to other domains.

**Q3: Explain how you would create a reliable ground truth for evaluation (instead of simply using the miniLM results). How would you tag each chunk?**

I would use the following ways to create a reliable ground truth:

1. Generate pairs of questions and answers for each chapter in the book using an LLM (e.g. ChatGPT) and also ask it to provide quotes from the book that support the answer.
2. Manually review a sample of the generated question-answer pairs to ensure that the answers are correct and fully supported by the quoted text. This step helps validate the quality of the LLM's output.
3. Use both semantic and lexical search methods to generate a set of results (retrieved paragraphs) for each question.
4. For each question-result pair, use an LLM to evaluate the relevance of the retrieved paragraph and specifically, ask **How relevant is the paragraph to the question?** and **Does the paragraph explicitly contain the answer to the question?**
5. Assign labels based on the LLM’s evaluation:
   * Label 2: The paragraph explicitly contains the answer.
   * Label 1: The paragraph is relevant (contains information that helps infer the answer) but does not explicitly contain the answer.
   * Label 0: The paragraph is irrelevant.
   * Since it is possible that the answer can be infered only from combining multiple paragraph it is important to label such paragraph as relevant (label 1).
6. Manually verify a sample of the LLM-labelled pairs to ensure accuracy and consistency in the labelling process.
7. Use this new dataset to evaluate the retrieval system.