## Your names

Roee Ezer 312493281, Aviv Weidenfeld 315589937

## A description of your system and the choices you made. We should be able to understand how your system works in all the different parts, and especially the key choices and key points that are required to replicate it. If something worked particularly well, specify it. Structure this part with a dedicated section for each of the stages: pre-processing, indexing (sparse and dense), retrieval, RAG (unconstrained and budget-constrained).

## Our RAG system is composed of 4 components (each has its interface and implementations):

**pre\_proccessor** – process the html pages to WebTextUnit (interface).

Here we decided that each WebTextUnit will represent a paragraph or a “section” as defined in the script extract\_content.py and each section will contain the id of the doc and the id of the section (just the index of the section inside the doc) and the full index of the section will be:

self.doc\_id + "\_" + self.section\_id

This approach worked well for us and we didn’t tried to changes it except for a caching mechanism (pickle serialization) to avoid processing the html files in each run.

**index\_optimizers –** these optimizers manipulate the query or the documents or both to improve the indexing performance.

The changes made to the optimizers are relevant only to the indexing part. The “final\_answer\_retriever” for example will use the original query/documents.

We tried many kinds of optimizers as you can see in the folder components/IndexOptimizer. The one that worked the best was prefix\_suffix\_splitter\_optimizer.py which was our implementation of lemmatization with a few rules of thumb that achieved very fast and good results in the recall,mrr metrics.

**data\_indexer - @aviv fill this part**

**final\_answer\_retriever**

**This class has a one implementation that sends Gemini through the free tier API the query and the text to answer by in the following pattern:**

**“**

**תקרא את השאלה הבאה:**

**{Query}**

**תענה בקצרה על השאלה לפי הטקסט הבא:**

**{AnswerSource}**

**“**

**We Assumed here that we need to answer the question only on based of the web pages data and never get an answer from the LLM without a source.**

## The evaluation results you obtained.

* 1. End-to-end evaluation, in which you go from query to final answer. This will have to be done manually. Choose 10-20 queries and manually evaluate their answers. Beyond accuracy, see if you can find common trends.

**We will split this question to 3:**

**Queries with low rank (1-3):**

**These queries’ final answer is usually good and sometimes more elaborative then the one in the gold document as expected – because it contains data from more documents.**

**Although there are some cases where the rank is misleading because we did retrieve the wrong section of the correct document like in this query "איזה תושבים זכאים לגמילה מסמים באשפוז?" in which we retrieved only section number 1 without section number 3 which seem more relevant.**

**Queries with high rank (10+):**

**These queries acted similarly to the queries with low rank. Most of the final answer were accurate except where the rank was misleading because we did retrieve the wrong section of the correct document like in this query "אילו סוגי דיור מתמודדי נפש זכאים"**

**Queries no rank (gold document was not retrieved):**

**The final answers here do not answer the question according to the specified text and sometimes returns that there is no answer in the text to the question.**

**In some cases, the reason the document is not found is the use of synonyms like in:**

**"האם יש זכאות למלגה לילדים של ניצולי שואה" – we use “מלגה” instead of “מענק”**

**In some cases it’s the use of number range like "האם כללית צריכה לממן את טיפולי הגמגום של בני בן ה-7?"**

**In some cases the query seem to be wrongly labeled like in here:**

**"gold\_doc\_id": "7cfab80d",**

**"query": "האם גם בן הזוג יכול לקבל חופשת לידה?",**

* 1. Additionally, see if you can identify cases where the retrieval found the correct document but the overall system produced a wrong answer, or the other way around: cases where the retrieval failed but the system overall produced an adequate answer.

**Yes, in this query which was probably wrongly labeled with the wrong gold document id, our rag system produced the correct answer according to the correct document 228fae0f**

**"gold\_doc\_id": "7cfab80d",**

**"query": "האם גם בן הזוג יכול לקבל חופשת לידה?",**

**"rank": null,**

**"final\_answer": "כן, בן הזוג יכול לקבל חופשת לידה בתנאים מסוימים, כגון: החלפת היולדת לאחר 6 שבועות, שבוע במקביל ליולדת (בהסכמתה בכתב), במקרה של לידה מרובת עוברים או אם היולדת אינה מסוגלת לטפל בילד.\n",**

* 1. Do things change from the unconstrained to the token-budget constrained cases?

**Yes, the 1000 tokens constraint didn’t affect most queries but it did affect some of the higher rank queries. For example this one:**

**"gold\_doc\_id": "7b3779b0",**

**"query": "האם מותר להעסיק אישה בהיריון בחודש השישי ביום שבת?",**

**"rank": 14,**

**"final\_answer": "אסור להעסיק אישה בהיריון בחודש השישי בשבת, אלא אם כן קיימים התנאים הבאים: העובדת הסכימה בכתב, קיים אישור רפואי של רופא מומחה ביילוד ובגניקולוגיה שאין מניעה להעסיקה, והעסקתה תהיה בהתאם לתנאים המפורטים באישור הרפואי.\n",**

**"final\_answer\_contraint": "הטקסט לא עוסק במותר או אסור להעסיק אישה בהיריון בשבת. הוא מתייחס למגבלות על שעות עבודה נוספות, מנוחה שבועית, וחגים, אך לא להיריון. לכן, אי אפשר לענות על השאלה על סמך הטקסט הזה.\n",**

* 1. Retrieval Evaluation, here, we evaluate the system on its ability to find the correct page (ignoring the RAG part). Following part 1, we ask you to use two metrics: The first metric is recall@k which measures how many times the correct page was within the top-k retrieved documents (after reranking, if you do them). Use k=5 and k=20. The second metric is MRR.
     1. Dev-set

**"recall\_20": 0.65,**

**"recall\_5": 0.5,**

**"mmr": 0.37,**

* + 1. Test-set

**"recall\_20": 0.89,**

**"recall\_5": 0.76,**

**"mmr": 0.63,**

## Can you think of ways to perform non-manual end-to-end evaluation, given the data you created? Note that we asked you to also include an annotated text-span containing the answer. Can you use this for automatic evaluation of the end-to-end RAG system? How? Discuss this in your report.

@aviv

## If you experimented with different methods and ideas, it is good to describe not only what worked, but also what didn’t work.

\*In general we focused on the queries who didn’t get a rank (the gold document was not retrieved) and from the instructions suggestions to get inspirations for improvements.

We experimented several ideas with the following components:

**Optimizers**

1. lemmatize\_optim\_trankit.py @aviv
2. lemmatize\_optim\_bert.py @aviv
3. prefix\_suffix\_splitter\_optimizer.py – this is very simple try to achieve something that is similar to lemmatization using a few rules of thumb for and splitting common Hebrew prefixes and suffixes. For example if we see the word “הדרכים” we will add the following forms to the text “הדרכים דרכים דרך הדרך”. We did the transformation to both the query and the documents text
4. hyde\_indexing\_optimizer.py – this optimizer implements the suggestion from the instruction “use an LLM to generate an answer, and then use the answer as the query”. This optimizer made the recall,mrr metrics non-deterministic because the gemini LLM doesn’t return the same answer to the query each time. We tried both with and without the original query and it in most runs made our metrics worse
5. word\_filtering\_indexing\_optimizer.py – this filters a constant list of words that seemed irrelevant to the query searching like “אבל”, “מה”, “האם”. In some cases combined with prefix\_suffix\_splitter\_optimizer it increased a bit the recall and decreased a bit the MRR. Eventually we decided not to use it.
6. synonym\_encrichment\_optimizer.py – This optimizer add for each word a list of synonym words using alephbert-base model. The synonym words are not good in quality and this optimizer did not improve our metrics.

**Indexers**

@aviv

## What would you have implemented if you had unlimited time and compute resources? What would be an ideal approach that you think would work very well? Be specific, e.g. don’t say general things like “I will train my own similarity model” but describe on what kind of data you will train it.

The main challenge of the rag system is the indexing and retrieve components. To improve this component we believe that good quality Lemmatization and synonym enrichment process can improve the indexing and retrieval significantly.

@aviv (optional): if you have anymore more to add here

## Any additional thoughts you had based on the assignment, or ideas you may have.

Usually looking and the queries with no rank (the golden document was not retrieved) was the best place to think about idea to improve. We notice that in some cases special characters interfere with the index and retrieval process.

Also, working on questions that contains number ranges like **"האם כללית צריכה לממן את טיפולי הגמגום של בני בן ה-7?" is quite a challenge and will probability require a significant dedicated effort to tackle.**

## A discussion of the challenges of part 2 compared to part 1, what are their causes, and what do you think is needed to overcome them.

@aviv