Natural Language Processing FINAL CA

Student Name:
Pouya Haji Mohammadi Gohari

SID:810102113

Date of deadline
Sunday 9th June, 2024

Dept. of Computer Engineering

University of Tehran

Contents

1 Dataset 4

2 Part One: Load and Prepare Dataset 5
2.1 Derive Chapters . . . 5
2.2 Load Each PDF . . . 6
2.3 Recursive Character Text Splitter . . . 6
2.4 Explaining Chunking Process and Why Chunking is Necessary . . . 7
2.5 Chose Chunk Size and Overlap Carefully. . . . 7

3 Part Two: Create Embedding and Vector Store 8
3.1 Embedder . . . 8
3.2 Importance of Embedder . . . 8

4 Part Three: Combination of Sparse and Dense Retrievers 9
4.1 Differences Between Dense and Sparse . . . 9
4.2 Combined Retriever . . . 9
4.3 Proper Weights for Combination . . . 9
4.4 Three Example . . . 11

5 Part Four: Router Chain 13
5.1 Together AI . . . 13
5.2 Load Llama . . . 13
5.3 Create Router Chain . . . 13
5.4 Add Parser to Router Chain . . . 14
5.5 Why Temperature is Set 0? . . . 14

6 Part Five: Search Engine Chain 15
6.1 Tavily API . . . 15
6.2 Example . . . 15
6.3 Create Search Engine Chain . . . 15

7 Part Six: Relevancy Checker Chain 17
7.1 Prompt . . . 17
7.2 Model . . . 17
7.3 Post-Process . . . 17
7.4 Why Relevancy Chain Is Necessary? . . . 17

8 Part Seven: Fallback Chain 19
8.1 Prompt . . . 19
8.2 Model . . . 19
8.3 Parser . . . 19

9 Part Eight: Generate With Context Chain **21**
9.1 Prompt 21
9.2 Model 21
9.3 Parser 21

10 Part Nine: Preparation Graph With LangGraph **23**
10.1 State 23
10.2 Router Node 23
10.3 Vector Store Node 24
10.4 Search Engine Node 24
10.5 Filter Docs Node 24
10.6 Fallback Node 24
10.7 Workflow 25
10.8 Chats 26
References26

1 Dataset

In this CA, we consider to use chapters from source book of NLP[1] where can be achieved from this Stanford site.

[1]Natural Language Processing

2 Part One: Load and Prepare Dataset

2.1 Derive Chapters

From utilizing code below, we can obtain links corresponding to chapters.

```python
def extract_all_links(given_url:str) -> list[str]:
    response = requests.get(url)
    if response.status_code == 200:
        soup = BeautifulSoup(response.content, "html.parser")
        links = soup.find_all("a")
        pdf_links = [link.get('href') for link in links if link.get('href') and
           ↪ link.get('href').endswith('.pdf') and re.match(r'^\d', link.get('
           ↪ href'))]
        full_pdf_links = [urljoin(url, pdf_link) for pdf_link in pdf_links]
        return full_pdf_links
    else:
        print(f"Failed to retrieve the webpage. Status code: {response.status_code
           ↪ }")
        return None
```

We have written detailed descriptions for almost every piece of code. However, we decided not to include these descriptions in the report to keep it concise.
In this piece of code, we decide to use 'requests' and 'bs4' libraries to achieve wanted links. From give a URL, we send a GET request using 'request.get' and stores the corresponding respond. If request was successful(200 for status), 'BeautifulSoup' parses the HTML content and store it to 'soup' object. Now we will find all hyperlinks using 'find_all' and store it to in a list called links. Afterward, filtering process will be proceed to achieve only the chapters(Note that chapters ends with pdf followed by a digit number). Then each pdf will be concatenate with given URL to achieve the corresponding links.
The output would be like this:

```
['https://stanford.edu/~jurafsky/slp3/2.pdf',
 'https://stanford.edu/~jurafsky/slp3/3.pdf',
 'https://stanford.edu/~jurafsky/slp3/4.pdf',
 'https://stanford.edu/~jurafsky/slp3/5.pdf',
 'https://stanford.edu/~jurafsky/slp3/6.pdf',
 'https://stanford.edu/~jurafsky/slp3/7.pdf',
 'https://stanford.edu/~jurafsky/slp3/8.pdf',
 'https://stanford.edu/~jurafsky/slp3/9.pdf',
 'https://stanford.edu/~jurafsky/slp3/10.pdf',
 'https://stanford.edu/~jurafsky/slp3/11.pdf',
 'https://stanford.edu/~jurafsky/slp3/13.pdf',
 'https://stanford.edu/~jurafsky/slp3/14.pdf',
 'https://stanford.edu/~jurafsky/slp3/15.pdf',
 'https://stanford.edu/~jurafsky/slp3/16.pdf',
 'https://stanford.edu/~jurafsky/slp3/17.pdf',
 'https://stanford.edu/~jurafsky/slp3/18.pdf',
 'https://stanford.edu/~jurafsky/slp3/19.pdf',
 'https://stanford.edu/~jurafsky/slp3/20.pdf',
 'https://stanford.edu/~jurafsky/slp3/21.pdf',
 'https://stanford.edu/~jurafsky/slp3/22.pdf',
 'https://stanford.edu/~jurafsky/slp3/23.pdf']
```

2.2 Load Each PDF

Now it is time to load each PDF with 'PyPDFLoader' in 'LangChain' library. The following code will help us:

```
def load_pdfs_from_urls(pdf_links: list[str]) -> list[list[PyPDFLoader]]:
    documents = []
    for url in pdf_links:
        loader = PyPDFLoader(url)
        documents.append(loader.load())
    return documents
```

Since 'PyPDFLoader' just load just one URL at a time, we iterate on the URLS in order to load each PDF. The length of documents list is 21(Since we have 21 chapters in the source book!)

2.3 Recursive Character Text Splitter

For chunking process, we create an object of 'RecursiveCharacterTextSplitter' as follow:

```
chunk_size = 1024
chunk_overlap = 64
text_splitter = RecursiveCharacterTextSplitter(
  length_function=len,
  chunk_size=chunk_size,
  chunk_overlap=chunk_overlap
)
```

Before chunking documents, we will extend documents since documents list contains a list of Document objects.(Documents have 2d list where docuemnts$[i][j]$ corresponds to j-th page of i-th chapter). The total number of pages are 568. First we will extend the documents meaning that we append each page in another list.(The i-th index of this list is (i-1)-th page of the concatenated chapters)

```
extended_docs = []
for chapter in documents:
  for page in chapter:
    extended_docs.append(page)
```

Now we can easily chunk them using 'RecursiveCharacterTextSplitter':

```
chunks = text_splitter.split_documents(extended_docs)
```

Total number of chunks are 1984 where each has at most 1024 characters.

2.4 Explaining Chunking Process and Why Chunking is Necessary

According to documentation[4], there is default separator list ["$\n\n$", "$\n$", " ", ""].This algorithm first attempt to chunk the text into paragraphs. If the number of characters does not exceed the specified chunk size the paragraph is kept as a chunk. Otherwise the paragraph is further split into sentences. The same process is applied recursively: if the chunk exceeds the chunk size, it is split into smaller units such as words and finally characters if necessary. Overlapping between chunks will ensure that no critical information is lost between these chunks.
According to another documentation[2], the main reason of chunking is to split a long document into smaller chunks that can fit our model's context window. The second reason is explain in an example:
Assume you have 10-pages and someone asking you a question that can be answered by just a paragraph within that document. In this scenario, questioner prefer to get just the paragraph rather than whole 10-pages! Furthermore, the second reason is we only want the relevant pieces of information.
With suggested parameters, through investing them, we detect no overlapped chunks!

```python
for index in range(len(chunks)-1):
  if len(chunks[index].page_content) > chunk_size:
    print(chunks[index])
    print("-----------")
    print(len(chunks[index].page_content))
    print("-----------")
    print(chunks[index + 1])
    print("-----------")
    break
```

2.5 Chose Chunk Size and Overlap Carefully.

We should very careful with these parameters. Some logical reasons are:

1. If chunk size is too small, it might have not enough context to our model to understand and process the text effectively.
2. If chunk size is too large, from the same reason explained in previous subsection, the model's context window(maximum number of input's tokens) can be exceeded!
3. With setting low number for chunk overlap, important information where lies between this chunks, can be lost.
4. Setting high number for chunk overlap, the same information can be repeated along with chunks.

Note that if an LLM's maximum input token is 4k, it is better to consider 2k token as input since these models will also concatenate the chat history to our query. Therefore we shall use chunks with accurate number of tokens.(In here we can specify just characters but they are also some splitter where use token strategy to split)

3 Part Two: Create Embedding and Vector Store

3.1 Embedder

In this subsection, we will utilize default embedder from 'huggingface' and implement a cache backed embedding system. This system will embed documents once and storing embeds and used the stored embeddings to avoid the need to re-embed the documents.

```
embedding_function = HuggingFaceEmbeddings(show_progress=True, multi_process=True,
    ↪ model_kwargs={"device": device})
store = LocalFileStore("./cache/")
namespace = str(embedding_function.model_name)
cached_embedder = CacheBackedEmbeddings.from_bytes_store(
    embedding_function, store, namespace=namespace
)
```

Note that embedder has been loaded to the GPU to recompute the embedding faster.

3.2 Importance of Embedder

There are several embedders where a leader board in huggingface leader board[1]. Which best embedder can be chosen by our task since they are multiple evaluation benchmarks this leader board has.
Let's say we use an embedder where did not see Persian data while training. Now if use this embedder for Persian sentences we will face some issues like:

1. The fact that each language have a unique syntactic, semantic and grammatical rules. The chosen embedder may not capture these features effectively for another language like Persian.

2. Tasks like sentiment analysis, text classification or named entity recognition are so reliable to embeddings. Using wrong embedder can effect our performance so much.

3. Embeddings obtained by this embedder are used to measure semantic similarity between sentences and documents as well.Therefore, an embedder that are not familiar with Persian sentences can result in poor retrieval results(Like in our case)

Instead, we can use multilingual embedders where trained on multiple languages.

4 Part Three: Combination of Sparse and Dense Retrievers

4.1 Differences Between Dense and Sparse

Generally, there are two kinds of retrievers:(The following context are according to ensemble retriever[3])

1. Sparse(Lexical): This retrievers are effective at identifying relevant documents based on the keyword matching.(or lexical matching)
2. Dense(Semantic): In contrast, Dense retrievers are capable of finding relevant documents through semantic similarities with the help of an embedder.(Note that Sparse won't use embedder)

Since their strengths are complementary, it is so common to combine them.

4.2 Combined Retriever

In order to combine two kind of retrievers based on the previous subsection, 'BM25Retriever' and 'FAISS' were instantiated as sparse and dense retrievers respectively. Then using 'EnsembleRetriever', we will combine these two.
'BM25Retriever':

```
bm25_retriever = BM25Retriever.from_documents(
    documents=chunks
)
bm25_retriever.k = 2
```

'FAISS':

```
faiss_vectorstore = FAISS.from_documents(
    documents=chunks,
    embedding=cached_embedder
)
faiss_retriever = faiss_vectorstore.as_retriever(search_kwargs={"k": 2})
```

In next subsection, we have multiple ensemble retriever in order to find the best weights for each retriever.

4.3 Proper Weights for Combination

In order to see how weights effect the order of documents and which one is better.(Note that each retrievers will give us two documents therefore for each given query we will have 4 documents)
First combination with $weights = [0.5, 0.5]$

```
ensemble_retriever_1 = EnsembleRetriever(
    retrievers=[bm25_retriever, faiss_retriever], weights=[0.5, 0.5]
)
```

Second combination with $weights = [0.4, 0.6]$:

```
ensemble_retriever_2 = EnsembleRetriever(
    retrievers=[bm25_retriever, faiss_retriever], weights=[0.4, 0.6]
)
```

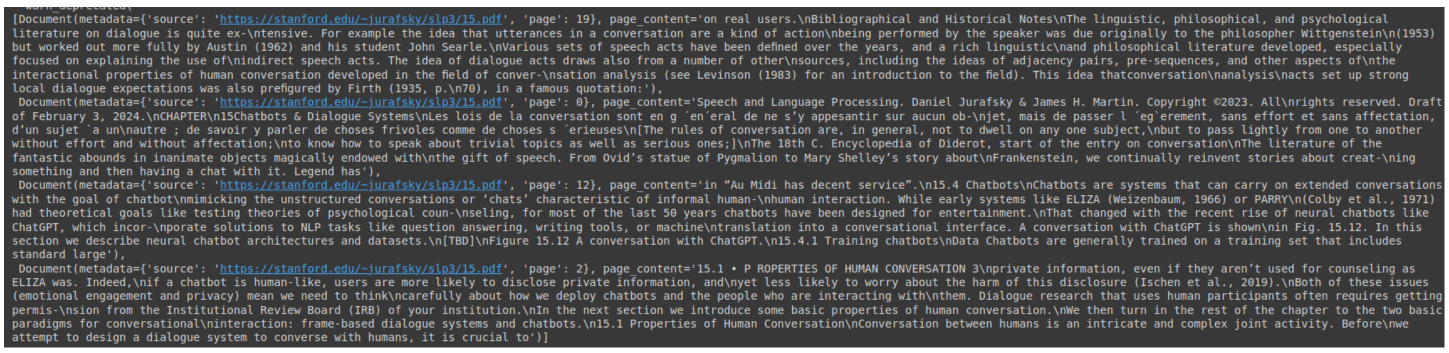
Third combination with $weights = [0.6, 0.4]$:

```python
ensemble_retriever_3 = EnsembleRetriever(
    retrievers=[bm25_retriever, faiss_retriever], weights=[0.6, 0.4]
)
```

Now let's see what order and documents they will provide for us based on this query:

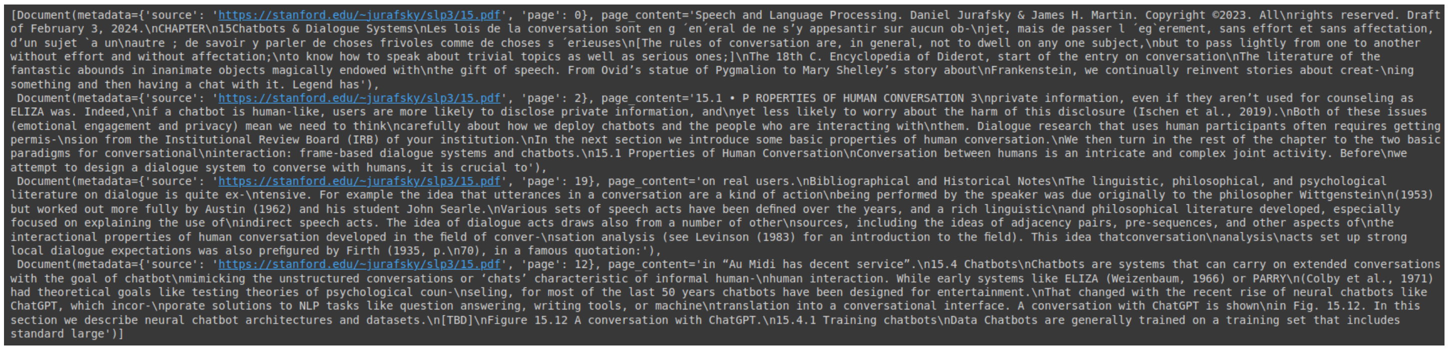
$$\text{Query} = \text{properties of human conversation}$$

The Figure 1, Figure 2 and Figure 3 are retrieval results respectievly based on these three combination.



```
[Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 19}, page_content='on real users.\nBibliographical and Historical Notes\nThe linguistic, philosophical, and psychological
literature on dialogue is quite ex-\ntensive. For example the idea that utterances in a conversation are a kind of action\nbeing performed by the speaker was due originally to the philosopher Wittgenstein\n(1953)
but worked out more fully by Austin (1962) and his student John Searle.\nVarious sets of speech acts have been defined over the years, and a rich linguistic\nand philosophical literature developed, especially
focused on explaining the use of\nindirect speech acts. The idea of dialogue acts draws also from a number of other\nsources, including the ideas of adjacency pairs, pre-sequences, and other aspects of\nthe
interactional properties of human conversation developed in the field of conver-\nsation analysis (see Levinson (1983) for an introduction to the field). This idea thatconversation\nanalysis\nacts set up strong
local dialogue expectations was also prefigured by Firth (1935, p.\n70), in a famous quotation:'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 0}, page_content='Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright ©2023. All\nrights reserved. Draft
of February 3, 2024.\nCHAPTER\n15Chatbots & Dialogue Systems\nLes lois de la conversation sont en g ´en´eral de ne s'y appesantir sur aucun ob-\njet, mais de passer l ´eg`erement, sans effort et sans affectation,
d'un sujet `a un\nautre ; de savoir y parler de choses frivoles comme de choses s ´erieuses\n[The rules of conversation are, in general, not to dwell on any one subject,\nbut to pass lightly from one to another
without effort and without affectation;\nto know how to speak about trivial topics as well as serious ones;]\nThe 18th C. Encyclopedia of Diderot, start of the entry on conversation\nThe literature of the
fantastic abounds in inanimate objects magically endowed with\nthe gift of speech. From Ovid's statue of Pygmalion to Mary Shelley's story about\nFrankenstein, we continually reinvent stories about creat-\ning
something and then having a chat with it. Legend has'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 12}, page_content='in "Au Midi has decent service".\n15.4 Chatbots\nChatbots are systems that can carry on extended conversations
with the goal of chatbot\nmimicking the unstructured conversations or 'chats' characteristic of informal human-\nhuman interaction. While early systems like ELIZA (Weizenbaum, 1966) or PARRY\n(Colby et al., 1971)
had theoretical goals like testing theories of psychological coun-\nseling, for most of the last 50 years chatbots have been designed for entertainment.\nThat changed with the recent rise of neural chatbots like
ChatGPT, which incor-\nporate solutions to NLP tasks like question answering, writing tools, or machine\ntranslation into a conversational interface. A conversation with ChatGPT is shown\nin Fig. 15.12. In this
section we describe neural chatbot architectures and datasets.\n[TBD]\nFigure 15.12 A conversation with ChatGPT.\n15.4.1 Training chatbots\nData Chatbots are generally trained on a training set that includes
standard large'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 2}, page_content='15.1 • P ROPERTIES OF HUMAN CONVERSATION 3\nprivate information, even if they aren't used for counseling as
ELIZA was. Indeed,\nif a chatbot is human-like, users are more likely to disclose private information, and\nyet less likely to worry about the harm of this disclosure (Ischen et al., 2019).\nBoth of these issues
(emotional engagement and privacy) mean we need to think\ncarefully about how we deploy chatbots and the people who are interacting with\nthem. Dialogue research that uses human participants often requires getting
permis-\nsion from the Institutional Review Board (IRB) of your institution.\nIn the next section we introduce some basic properties of human conversation.\nWe then turn in the rest of the chapter to the two basic
paradigms for conversational\ninteraction: frame-based dialogue systems and chatbots.\n15.1 Properties of Human Conversation\nConversation between humans is an intricate and complex joint activity. Before\nwe
attempt to design a dialogue system to converse with humans, it is crucial to')]
```

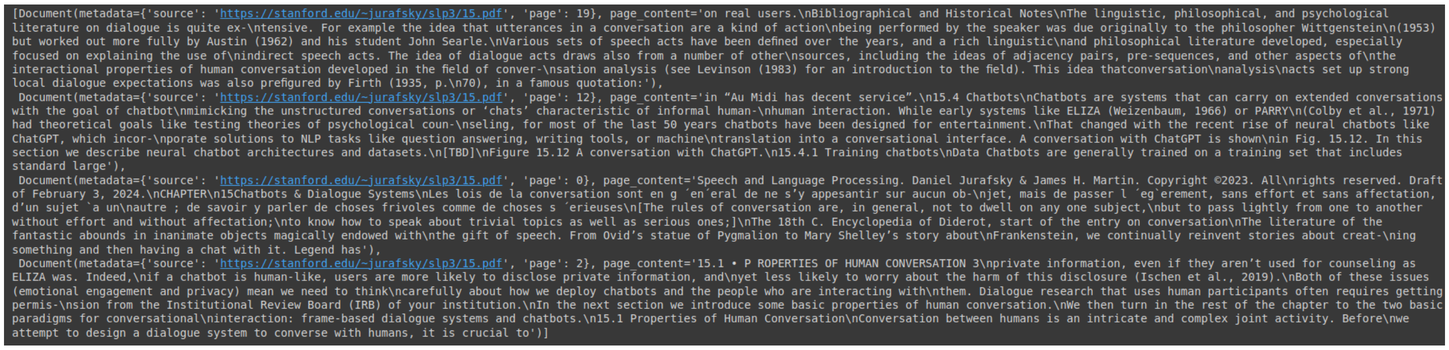
Figure 1: Combination $0.5, 0.5$



```
[Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 0}, page_content='Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright ©2023. All\nrights reserved. Draft
of February 3, 2024.\nCHAPTER\n15Chatbots & Dialogue Systems\nLes lois de la conversation sont en g ´en´eral de ne s'y appesantir sur aucun ob-\njet, mais de passer l ´eg`erement, sans effort et sans affectation,
d'un sujet `a un\nautre ; de savoir y parler de choses frivoles comme de choses s ´erieuses\n[The rules of conversation are, in general, not to dwell on any one subject,\nbut to pass lightly from one to another
without effort and without affectation;\nto know how to speak about trivial topics as well as serious ones;]\nThe 18th C. Encyclopedia of Diderot, start of the entry on conversation\nThe literature of the
fantastic abounds in inanimate objects magically endowed with\nthe gift of speech. From Ovid's statue of Pygmalion to Mary Shelley's story about\nFrankenstein, we continually reinvent stories about creat-\ning
something and then having a chat with it. Legend has'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 2}, page_content='15.1 • P ROPERTIES OF HUMAN CONVERSATION 3\nprivate information, even if they aren't used for counseling as
ELIZA was. Indeed,\nif a chatbot is human-like, users are more likely to disclose private information, and\nyet less likely to worry about the harm of this disclosure (Ischen et al., 2019).\nBoth of these issues
(emotional engagement and privacy) mean we need to think\ncarefully about how we deploy chatbots and the people who are interacting with\nthem. Dialogue research that uses human participants often requires getting
permis-\nsion from the Institutional Review Board (IRB) of your institution.\nIn the next section we introduce some basic properties of human conversation.\nWe then turn in the rest of the chapter to the two basic
paradigms for conversational\ninteraction: frame-based dialogue systems and chatbots.\n15.1 Properties of Human Conversation\nConversation between humans is an intricate and complex joint activity. Before\nwe
attempt to design a dialogue system to converse with humans, it is crucial to'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 19}, page_content='on real users.\nBibliographical and Historical Notes\nThe linguistic, philosophical, and psychological
literature on dialogue is quite ex-\ntensive. For example the idea that utterances in a conversation are a kind of action\nbeing performed by the speaker was due originally to the philosopher Wittgenstein\n(1953)
but worked out more fully by Austin (1962) and his student John Searle.\nVarious sets of speech acts have been defined over the years, and a rich linguistic\nand philosophical literature developed, especially
focused on explaining the use of\nindirect speech acts. The idea of dialogue acts draws also from a number of other\nsources, including the ideas of adjacency pairs, pre-sequences, and other aspects of\nthe
interactional properties of human conversation developed in the field of conver-\nsation analysis (see Levinson (1983) for an introduction to the field). This idea thatconversation\nanalysis\nacts set up strong
local dialogue expectations was also prefigured by Firth (1935, p.\n70), in a famous quotation:'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 12}, page_content='in "Au Midi has decent service".\n15.4 Chatbots\nChatbots are systems that can carry on extended conversations
with the goal of chatbot\nmimicking the unstructured conversations or 'chats' characteristic of informal human-\nhuman interaction. While early systems like ELIZA (Weizenbaum, 1966) or PARRY\n(Colby et al., 1971)
had theoretical goals like testing theories of psychological coun-\nseling, for most of the last 50 years chatbots have been designed for entertainment.\nThat changed with the recent rise of neural chatbots like
ChatGPT, which incor-\nporate solutions to NLP tasks like question answering, writing tools, or machine\ntranslation into a conversational interface. A conversation with ChatGPT is shown\nin Fig. 15.12. In this
section we describe neural chatbot architectures and datasets.\n[TBD]\nFigure 15.12 A conversation with ChatGPT.\n15.4.1 Training chatbots\nData Chatbots are generally trained on a training set that includes
standard large')]
```

Figure 2: Combination $0.4, 0.6$

According to these Figures, all documents are same but order of these documents are different. But personally I will chose the second one, since in my opinion the top related chunk is the one with topic of 'properties of human conversation'. To be honest, there is another reason that I use this retriever.Because embedders are highly powerful and enriched, a retriever based on embeddings is likely to capture better results than a lexicon retriever. Therefore, the weight of the semantic retriever should be slightly higher.
So the best retriever (second one) is better and is set to be main retriever:



```
[Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 19}, page_content='on real users.\nBibliographical and Historical Notes\nThe linguistic, philosophical, and psychological
literature on dialogue is quite ex-\ntensive. For example the idea that utterances in a conversation are a kind of action\nbeing performed by the speaker was due originally to the philosopher Wittgenstein\n(1953)
but worked out more fully by Austin (1962) and his student John Searle.\nVarious sets of speech acts have been defined over the years, and a rich linguistic\nand philosophical literature developed, especially
focused on explaining the use of\nindirect speech acts. The idea of dialogue acts draws also from a number of other\nsources, including the ideas of adjacency pairs, pre-sequences, and other aspects of\nthe
interactional properties of human conversation developed in the field of conver-\nsation analysis (see Levinson (1983) for an introduction to the field). This idea thatconversation\nanalysis\nacts set up strong
local dialogue expectations was also prefigured by Firth (1935, p.\n70), in a famous quotation:'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 12}, page_content='in “Au Midi has decent service”.\n15.4 Chatbots\nChatbots are systems that can carry on extended conversations
with the goal of chatbot\nmimicking the unstructured conversations or ‘chats’ characteristic of informal human-\nhuman interaction. While early systems like ELIZA (Weizenbaum, 1966) or PARRY\n(Colby et al., 1971)
had theoretical goals like testing theories of psychological coun-\nseling, for most of the last 50 years chatbots have been designed for entertainment.\nThat changed with the recent rise of neural chatbots like
ChatGPT, which incor-\nporate solutions to NLP tasks like question answering, writing tools, or machine\ntranslation into a conversational interface. A conversation with ChatGPT is shown\nin Fig. 15.12. In this
section we describe neural chatbot architectures and datasets.\n[TBD]\nFigure 15.12 A conversation with ChatGPT.\n15.4.1 Training chatbots\nData Chatbots are generally trained on a training set that includes
standard large'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 0}, page_content='Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright ©2023. All\nrights reserved. Draft
of February 3, 2024.\nCHAPTER\n15Chatbots & Dialogue Systems\nLes lois de la conversation sont en g ´en´eral de ne s’y appesantir sur aucun ob-\njet, mais de passer l ´eg`erement, sans effort et sans affectation,
d’un sujet `a un\nautre ; de savoir y parler de choses frivoles comme de choses s ´erieuses\n[The rules of conversation are, in general, not to dwell on any one subject,\nbut to pass lightly from one to another
without effort and without affectation;\nto know how to speak about trivial topics as well as serious ones;]\nThe 18th C. Encyclopedia of Diderot, start of the entry on conversation\nThe literature of the
fantastic abounds in inanimate objects magically endowed with\nthe gift of speech. From Ovid’s statue of Pygmalion to Mary Shelley’s story about\nFrankenstein, we continually reinvent stories about creat-\ning
something and then having a chat with it. Legend has'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 2}, page_content='15.1 • P ROPERTIES OF HUMAN CONVERSATION 3\nprivate information, even if they aren’t used for counseling as
ELIZA was. Indeed,\nif a chatbot is human-like, users are more likely to disclose private information, and\nyet less likely to worry about the harm of this disclosure (Ischen et al., 2019).\nBoth of these issues
(emotional engagement and privacy) mean we need to think\ncarefully about how we deploy chatbots and the people who are interacting with\nthem. Dialogue research that uses human participants often requires getting
permis-\nsion from the Institutional Review Board (IRB) of your institution.\nIn the next section we introduce some basic properties of human conversation.\nWe then turn in the rest of the chapter to the two basic
paradigms for conversational\ninteraction: frame-based dialogue systems and chatbots.\n15.1 Properties of Human Conversation\nConversation between humans is an intricate and complex joint activity. Before\nwe
attempt to design a dialogue system to converse with humans, it is crucial to')]
```

Figure 3: Combination $0.6, 0.4$

4.4 Three Example

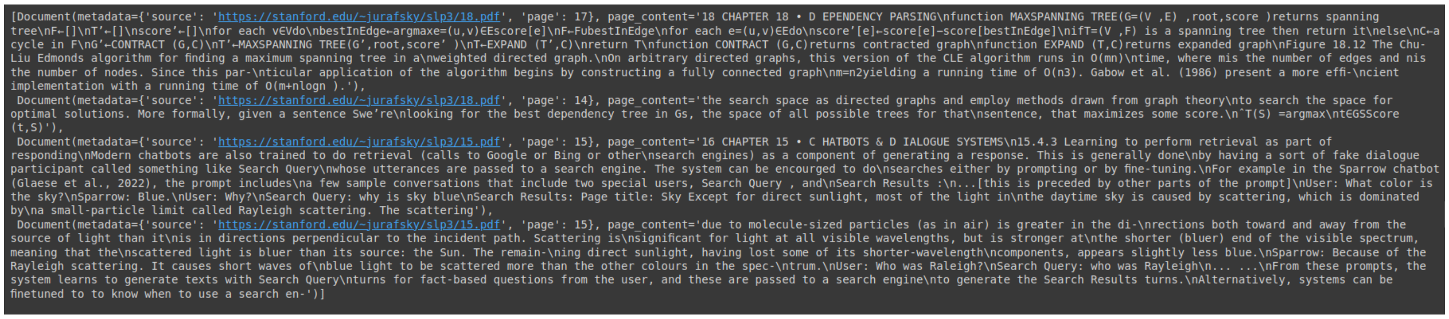
To evaluate this retriever, three queris have been given:

$$\text{Query}_1 = \text{Speech acts in chatbots}$$

$$\text{Query}_2 = \text{Binary Search Tree(BST)}$$

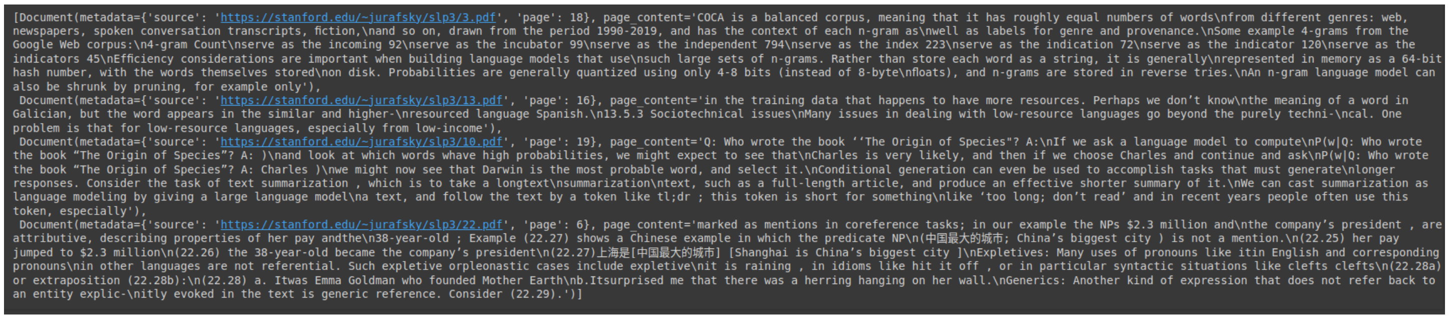
$$\text{Query}_3 = \text{Who is the president of Bolivia country?}$$

Now based on these queries, the retrieved resulst has been illustrated in Figure 4,Figure 5 and Figure 6: According



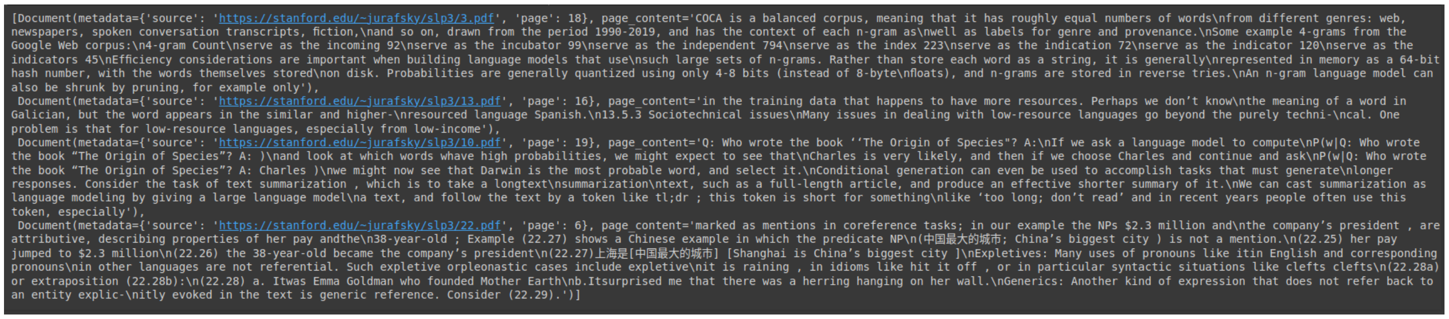
```
[Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/18.pdf', 'page': 17}, page_content='18 CHAPTER 18 • D EPENDENCY PARSING\nfunction MAXSPANNING TREE(G=(V ,E) ,root,score )returns spanning
tree\nF←[]\nT’←[]\nscore’←[]\nfor each v∈Vdo\nbestInEdge←argmaxe=(u,v)∈Escore[e]\nF←FubestInEdge\nfor each e=(u,v)∈Edo\nscore’[e]←score[e]−score[bestInEdge]\nifT=(V ,F) is a spanning tree then return it\nelse\nC←a
cycle in F\nG’←CONTRACT (G,C)\nT’←MAXSPANNING TREE(G’,root,score’ )\nT←EXPAND (T’,C)\nreturn T\nfunction CONTRACT (G,C)returns contracted graph\nfunction EXPAND (T,C)returns expanded graph\nFigure 18.12 The Chu-
Liu Edmonds algorithm for finding a maximum spanning tree in a\nweighted directed graph.\nOn arbitrary directed graphs, this version of the CLE algorithm runs in O(mn)\ntime, where mis the number of edges and nis
the number of nodes. Since this par-\nticular application of the algorithm begins by constructing a fully connected graph\nm=n2yielding a running time of O(n3). Gabow et al. (1986) present a more effi-\ncient
implementation with a running time of O(m+nlogn ).'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/18.pdf', 'page': 14}, page_content='the search space as directed graphs and employ methods drawn from graph theory\nto search the space for
optimal solutions. More formally, given a sentence Swe’re\nlooking for the best dependency tree in Gs, the space of all possible trees for that\nsentence, that maximizes some score.\nˆT(S) =argmax\nt∈GSScore
(t,S)'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 15}, page_content='16 CHAPTER 15 • C HATBOTS & D IALOGUE SYSTEMS\n15.4.3 Learning to perform retrieval as part of
responding\nModern chatbots are also trained to do retrieval (calls to Google or Bing or other\nsearch engines) as a component of generating a response. This is generally done\nby having a sort of fake dialogue
participant called something like Search Query\nwhose utterances are passed to a search engine. The system can be encourged to do\nsearches either by prompting or by fine-tuning.\nFor example in the Sparrow chatbot
(Glaese et al., 2022), the prompt includes\na few sample conversations that include two special users, Search Query , and\nSearch Results :\n...[this is preceded by other parts of the prompt]\nUser: What color is
the sky?\nSparrow: Blue.\nUser: Why?\nSearch Query: why is sky blue\nSearch Results: Page title: Sky Except for direct sunlight, most of the light in\nthe daytime sky is caused by scattering, which is dominated
by\na small-particle limit called Rayleigh scattering. The scattering'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/15.pdf', 'page': 15}, page_content='due to molecule-sized particles (as in air) is greater in the di-\nrections both toward and away from the
source of light than it\nis in directions perpendicular to the incident path. Scattering is\nsignificant for light at all visible wavelengths, but is stronger at\nthe shorter (bluer) end of the visible spectrum,
meaning that the\nscattered light is bluer than its source: the Sun. The remain-\ning direct sunlight, having lost some of its shorter-wavelength\ncomponents, appears slightly less blue.\nSparrow: Because of the
Rayleigh scattering. It causes short waves of\nblue light to be scattered more than the other colours in the spec-\ntrum.\nUser: Who was Raleigh?\nSearch Query: who was Rayleigh\n... ...\nFrom these prompts, the
system learns to generate texts with Search Query\nturns for fact-based questions from the user, and these are passed to a search engine\nto generate the Search Results turns.\nAlternatively, systems can be
finetuned to to know when to use a search en-')]
```

Figure 4: First Query Response



```
[Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/3.pdf', 'page': 18}, page_content='COCA is a balanced corpus, meaning that it has roughly equal numbers of words\nfrom different genres: web,
newspapers, spoken conversation transcripts, fiction,\nand so on, drawn from the period 1990-2019, and has the context of each n-gram as\nwell as labels for genre and provenance.\nSome example 4-grams from the
Google Web corpus:\n4-gram Count\nserve as the incoming 92\nserve as the incubator 99\nserve as the independent 794\nserve as the index 223\nserve as the indication 72\nserve as the indicator 120\nserve as the
indicators 45\nEfficiency considerations are important when building language models that use\nsuch large sets of n-grams. Rather than store each word as a string, it is generally\nrepresented in memory as a 64-bit
hash number, with the words themselves stored\non disk. Probabilities are generally quantized using only 4-8 bits (instead of 8-byte\nfloats), and n-grams are stored in reverse tries.\nAn n-gram language model can
also be shrunk by pruning, for example only'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/13.pdf', 'page': 16}, page_content='in the training data that happens to have more resources. Perhaps we don’t know\nthe meaning of a word in
Galician, but the word appears in the similar and higher-\nresourced language Spanish.\n13.5.3 Sociotechnical issues\nMany issues in dealing with low-resource languages go beyond the purely techni-\ncal. One
problem is that for low-resource languages, especially from low-income'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/10.pdf', 'page': 19}, page_content='Q: Who wrote the book ‘‘The Origin of Species"? A:\nIf we ask a language model to compute\nP(w|Q: Who wrote
the book “The Origin of Species”? A: )\nand look at which words whave high probabilities, we might expect to see that\nCharles is very likely, and then if we choose Charles and continue and ask\nP(w|Q: Who wrote
the book “The Origin of Species”? A: Charles )\nwe might now see that Darwin is the most probable word, and select it.\nConditional generation can even be used to accomplish tasks that must generate\nlonger
responses. Consider the task of text summarization , which is to take a longtext\nsummarization\ntext, such as a full-length article, and produce an effective shorter summary of it.\nWe can cast summarization as
language modeling by giving a large language model\na text, and follow the text by a token like tl;dr ; this token is short for something\nlike ‘too long; don’t read’ and in recent years people often use this
token, especially'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/22.pdf', 'page': 6}, page_content='marked as mentions in coreference tasks; in our example the NPs $2.3 million and\nthe company’s president , are
attributive, describing properties of her pay andthe\n38-year-old ; Example (22.27) shows a Chinese example in which the predicate NP\n(中国最大的城市; China’s biggest city ) is not a mention.\n(22.25) her pay
jumped to $2.3 million\n(22.26) the 38-year-old became the company’s president\n(22.27)上海是[中国最大的城市] [Shanghai is China’s biggest city ]\nExpletives: Many uses of pronouns like itin English and corresponding
pronouns\nin other languages are not referential. Such expletive orpleonastic cases include expletive\nit is raining , in idioms like hit it off , or in particular syntactic situations like clefts clefts\n(22.28a)
or extraposition (22.28b):\n(22.28) a. Itwas Emma Goldman who founded Mother Earth\nb.Itsurprised me that there was a herring hanging on her wall.\nGenerics: Another kind of expression that does not refer back to
an entity explic-\nitly evoked in the text is generic reference. Consider (22.29).')]
```

Figure 5: Second Query Response



```
[Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/3.pdf', 'page': 18}, page_content='COCA is a balanced corpus, meaning that it has roughly equal numbers of words\nfrom different genres: web,
newspapers, spoken conversation transcripts, fiction,\nand so on, drawn from the period 1990-2019, and has the context of each n-gram as\nwell as labels for genre and provenance.\nSome example 4-grams from the
Google Web corpus:\n4-gram Count\nserve as the incoming 92\nserve as the incubator 99\nserve as the independent 794\nserve as the index 223\nserve as the indication 72\nserve as the indicator 120\nserve as the
indicators 45\nEfficiency considerations are important when building language models that use\nsuch large sets of n-grams. Rather than store each word as a string, it is generally\nrepresented in memory as a 64-bit
hash number, with the words themselves stored\non disk. Probabilities are generally quantized using only 4-8 bits (instead of 8-byte\nfloats), and n-grams are stored in reverse tries.\nAn n-gram language model can
also be shrunk by pruning, for example only'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/13.pdf', 'page': 16}, page_content='in the training data that happens to have more resources. Perhaps we don’t know\nthe meaning of a word in
Galician, but the word appears in the similar and higher-\nresourced language Spanish.\n13.5.3 Sociotechnical issues\nMany issues in dealing with low-resource languages go beyond the purely techni-\ncal. One
problem is that for low-resource languages, especially from low-income'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/10.pdf', 'page': 19}, page_content='Q: Who wrote the book ‘‘The Origin of Species"? A:\nIf we ask a language model to compute\nP(w|Q: Who wrote
the book “The Origin of Species”? A: )\nand look at which words whave high probabilities, we might expect to see that\nCharles is very likely, and then if we choose Charles and continue and ask\nP(w|Q: Who wrote
the book “The Origin of Species”? A: Charles )\nwe might now see that Darwin is the most probable word, and select it.\nConditional generation can even be used to accomplish tasks that must generate\nlonger
responses. Consider the task of text summarization , which is to take a longtext\nsummarization\ntext, such as a full-length article, and produce an effective shorter summary of it.\nWe can cast summarization as
language modeling by giving a large language model\na text, and follow the text by a token like tl;dr ; this token is short for something\nlike ‘too long; don’t read’ and in recent years people often use this
token, especially'),
 Document(metadata={'source': 'https://stanford.edu/~jurafsky/slp3/22.pdf', 'page': 6}, page_content='marked as mentions in coreference tasks; in our example the NPs $2.3 million and\nthe company’s president , are
attributive, describing properties of her pay andthe\n38-year-old ; Example (22.27) shows a Chinese example in which the predicate NP\n(中国最大的城市; China’s biggest city ) is not a mention.\n(22.25) her pay
jumped to $2.3 million\n(22.26) the 38-year-old became the company’s president\n(22.27)上海是[中国最大的城市] [Shanghai is China’s biggest city ]\nExpletives: Many uses of pronouns like itin English and corresponding
pronouns\nin other languages are not referential. Such expletive orpleonastic cases include expletive\nit is raining , in idioms like hit it off , or in particular syntactic situations like clefts clefts\n(22.28a)
or extraposition (22.28b):\n(22.28) a. Itwas Emma Goldman who founded Mother Earth\nb.Itsurprised me that there was a herring hanging on her wall.\nGenerics: Another kind of expression that does not refer back to
an entity explic-\nitly evoked in the text is generic reference. Consider (22.29).')]
```

Figure 6: Third Query Response

to the results, since our chunks are related to just NLP, the retrieved results for first query is acceptable.In contrast, for the last two queries, we have not any reasonable related documents(cause our chunks are from Jurafsky book.

5 Part Four: Router Chain

5.1 Together AI

After sign up in Together AI, we can use 'Llama-3-70b-chat-hf'.

5.2 Load Llama

Utilizing following code, we will load the model.

```
llm = ChatTogether(
    model="meta-llama/Llama-3-70b-chat-hf",
    temperature= 0
    )
```

We have set the API_KEY at third cell:

```
os.environ["TAVILY_API_KEY"] = "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"
os.environ["TOGETHER_API_KEY"] = "3
   ↪ b6df4b272e648dfe1e16c91018e4e4f790580648e132a21757d474dd3a1d18b"
```

As description said, temperature is set to zero.

5.3 Create Router Chain

We will create a prompt for our model in order to just specify the model to choose between three routing options, 'VectorStore', 'SearchEngine' or 'None'.The created prompt is:

```
router_prompt_template = (
    "You are an expert in routing user queries to either a VectorStore,
       ↪ SearchEngine, or a Fallback Message.\n"
    "Your VectorStore contains data specifically about natural language processing
       ↪  (NLP). If the given query is about NLP or contains data related to NLP,
       ↪  choose VectorStore.\n"
    "If the query is not about NLP but is related to computer science or contains
       ↪ data about computer science, choose SearchEngine.\n"
    "If the query does not contain any data about NLP or computer science, do not
       ↪ choose any tool and return the string None.\n"
    "Give me only and only the name of the tool you chose and nothing more. If
       ↪ there are no chosen tools, give me back the string None.\n"
    "{output_instructions}\n"
    "query: {query}"
)
```

After creating prompt, we should use chat prompt template to chat model.(This will standardize for LangChain)

```
prompt = ChatPromptTemplate.from_template(
    template=router_prompt_template,
)
```

5.4 Add Parser to Router Chain

Now it is time to proceed create a custom parser.This custom parser will map LLM's output to a framework. That tool name is required to be filled.

```python
from typing import Literal
class ChosenTool(BaseModel):
    tool_name: Literal['None', "VectorStore", "SearchEngine"] = Field(description=
        ↪ "the tool that was chosen by LLM in question routing stage")
question_router_parser = PydanticOutputParser(pydantic_object=ChosenTool)
question_router_parser.get_format_instructions()
```

After define a "ChosenTool" class, an instance was created from Pydanic object.
Now chain simply is:

```python
question_router = prompt | llm | question_router_parser
res = question_router.invoke({"query": "My name is Pouya. who are you?",
                              "output_instructions": question_router_parser.
                                ↪ get_format_instructions()
 })
```

The output instruction must be in form of parser's output instructions.Three queries has been made to evaluate our router chain:

$$\text{Query}_1 = \text{My name is Pouya. who are you?}$$

$$\text{Query}_2 = \text{What is dependecy parsing?}$$

$$\text{Query}_3 = \text{What is Binary Search Tree?}$$

The respond for these queries are "ChosenTool(toolname='None') ", "ChosenTool(toolname='VectorStore')" and "ChosenTool(toolname='SearchEngine')" which prove that model correct;y respond to us.

5.5 Why Temperature is Set 0?

Temperature is designed to control either we want to diverse response or high-quality response. If temperature is set to zero, we want to get a high-quality respond while setting it response will be variate each time LLM generate an answer.
Since our task is easy enough where llm must choose one option among three, we do not want variation.Instead we want high-quality response to achieve more accurate predictions.

6 Part Five: Search Engine Chain

6.1 Tavily API

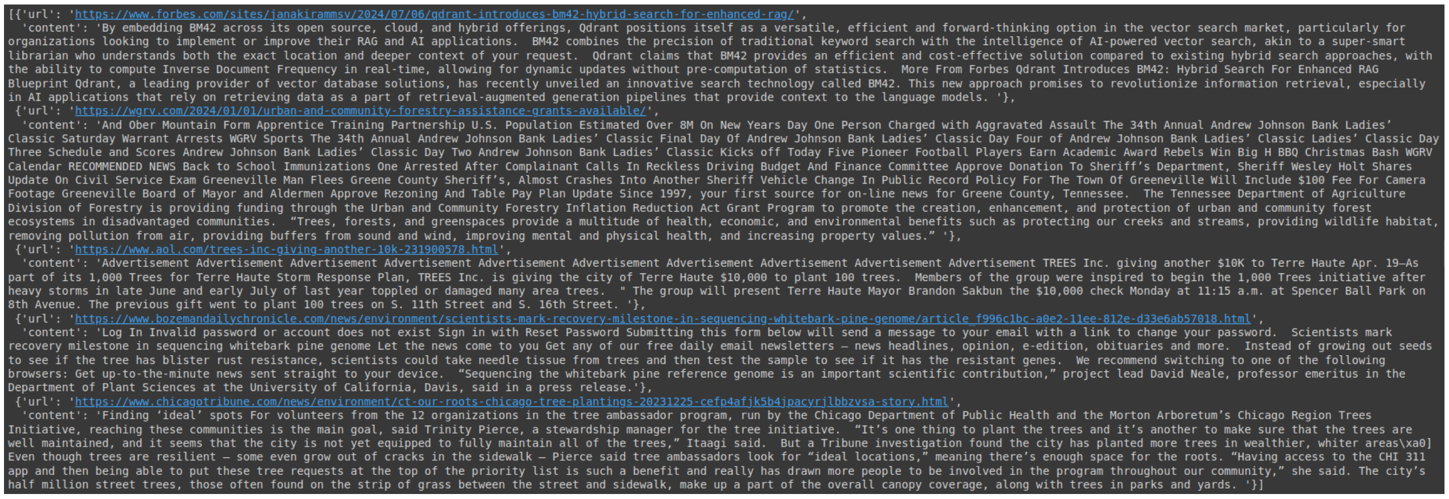
After signing up into Tavily API, we can use this API as a search engine.

6.2 Example

In order to be familiar with the output of ”TavilySearchResults”, the following query has been passed to this function with setting max_result parameter to 5.

```python
tool = TavilySearchResults(max_results=5)
tool.invoke({"query": "what is Binary Search Tree"})
```

But I have no idea why this search engine will response so bad(I have multiple conversation with TA for this). Result has been shown in Figure7:



```
[{'url': 'https://www.forbes.com/sites/janakirammsv/2024/07/06/qdrant-introduces-bm42-hybrid-search-for-enhanced-rag/',
  'content': 'By embedding BM42 across its open source, cloud, and hybrid offerings, Qdrant positions itself as a versatile, efficient and forward-thinking option in the vector search market, particularly for
organizations looking to implement or improve their RAG and AI applications.  BM42 combines the precision of traditional keyword search with the intelligence of AI-powered vector search, akin to a super-smart
librarian who understands both the exact location and deeper context of your request.  Qdrant claims that BM42 provides an efficient and cost-effective solution compared to existing hybrid search approaches, with
the ability to compute Inverse Document Frequency in real-time, allowing for dynamic updates without pre-computation of statistics.  More From Forbes Qdrant Introduces BM42: Hybrid Search For Enhanced RAG
Blueprint Qdrant, a leading provider of vector database solutions, has recently unveiled an innovative search technology called BM42. This new approach promises to revolutionize information retrieval, especially
in AI applications that rely on retrieving data as a part of retrieval-augmented generation pipelines that provide context to the language models. '},
 {'url': 'https://wgrv.com/2024/01/01/urban-and-community-forestry-assistance-grants-available/',
  'content': 'And Ober Mountain Form Apprentice Training Partnership U.S. Population Estimated Over 8M On New Years Day One Person Charged with Aggravated Assault The 34th Annual Andrew Johnson Bank Ladies’
Classic Saturday Warrant Arrests WGRV Sports The 34th Annual Andrew Johnson Bank Ladies’ Classic Final Day Of Andrew Johnson Bank Ladies’ Classic Day Four of Andrew Johnson Bank Ladies’ Classic Ladies’ Classic Day
Three Schedule and Scores Andrew Johnson Bank Ladies’ Classic Day Two Andrew Johnson Bank Ladies’ Classic Kicks off Today Five Pioneer Football Players Earn Academic Award Rebels Win Big H BBQ Christmas Bash WGRV
Calendar RECOMMENDED NEWS Back to School Immunizations One Arrested After Complainant Calls In Reckless Driving Budget And Finance Committee Approve Donation To Sheriff’s Department, Sheriff Wesley Holt Shares
Update On Civil Service Exam Greeneville Man Flees Greene County Sheriff’s, Almost Crashes Into Another Sheriff Vehicle Change In Public Record Policy For The Town Of Greeneville Will Include $100 Fee For Camera
Footage Greeneville Board of Mayor and Aldermen Approve Rezoning And Table Pay Plan Update Since 1997, your first source for on-line news for Greene County, Tennessee.  The Tennessee Department of Agriculture
Division of Forestry is providing funding through the Urban and Community Forestry Inflation Reduction Act Grant Program to promote the creation, enhancement, and protection of urban and community forest
ecosystems in disadvantaged communities.  “Trees, forests, and greenspaces provide a multitude of health, economic, and environmental benefits such as protecting our creeks and streams, providing wildlife habitat,
removing pollution from air, providing buffers from sound and wind, improving mental and physical health, and increasing property values.” '},
 {'url': 'https://www.aol.com/trees-inc-giving-another-10k-231900578.html',
  'content': 'Advertisement Advertisement Advertisement Advertisement Advertisement Advertisement Advertisement Advertisement Advertisement Advertisement Advertisement TREES Inc. giving another $10K to Terre Haute Apr. 19—As
part of its 1,000 Trees for Terre Haute Storm Response Plan, TREES Inc. is giving the city of Terre Haute $10,000 to plant 100 trees.  Members of the group were inspired to begin the 1,000 Trees initiative after
heavy storms in late June and early July of last year toppled or damaged many area trees.  " The group will present Terre Haute Mayor Brandon Sakbun the $10,000 check Monday at 11:15 a.m. at Spencer Ball Park on
8th Avenue. The previous gift went to plant 100 trees on S. 11th Street and S. 16th Street. '},
 {'url': 'https://www.bozemandailychronicle.com/news/environment/scientists-mark-recovery-milestone-in-sequencing-whitebark-pine-genome/article_f996c1bc-a0e2-11ee-812e-d33e6ab57018.html',
  'content': 'Log In Invalid password or account does not exist Sign in with Reset Password Submitting this form below will send a message to your email with a link to change your password.  Scientists mark
recovery milestone in sequencing whitebark pine genome Let the news come to you Get any of our free daily email newsletters — news headlines, opinion, e-edition, obituaries and more.  Instead of growing out seeds
to see if the tree has blister rust resistance, scientists could take needle tissue from trees and then test the sample to see if it has the resistant genes.  We recommend switching to one of the following
browsers: Get up-to-the-minute news sent straight to your device.  “Sequencing the whitebark pine reference genome is an important scientific contribution,” project lead David Neale, professor emeritus in the
Department of Plant Sciences at the University of California, Davis, said in a press release.'},
 {'url': 'https://www.chicagotribune.com/news/environment/ct-our-roots-chicago-tree-plantings-20231225-cefp4afjk5b4jpacyrjlbbzvsa-story.html',
  'content': 'Finding ‘ideal’ spots For volunteers from the 12 organizations in the tree ambassador program, run by the Chicago Department of Public Health and the Morton Arboretum’s Chicago Region Trees
Initiative, reaching these communities is the main goal, said Trinity Pierce, a stewardship manager for the tree initiative.  “It’s one thing to plant the trees and it’s another to make sure that the trees are
well maintained, and it seems that the city is not yet equipped to fully maintain all of the trees,” Itaagi said.  But a Tribune investigation found the city has planted more trees in wealthier, whiter areas\xa0]
Even though trees are resilient — some even grow out of cracks in the sidewalk — Pierce said tree ambassadors look for “ideal locations,” meaning there’s enough space for the roots. “Having access to the CHI 311
app and then being able to put these tree requests at the top of the priority list is such a benefit and really has drawn more people to be involved in the program throughout our community,” she said. The city’s
half million street trees, those often found on the strip of grass between the street and sidewalk, make up a part of the overall canopy coverage, along with trees in parks and yards. '}]
```

Figure 7: Tavily’s Result

6.3 Create Search Engine Chain

In previous subsection, we have use this tool.First, we are going to create a parser where given a tavily’s result will change them to a document list. The parser function is:

```python
def document_parsing(tavily_outputs:List[Dict[str, str]]) -> List[Document]:
    documents = []
    for iterate in tavily_outputs:
        for tavily_output in iterate:
            document = Document(
                page_content = tavily_output['content'],
                metadata={'url': tavily_output['url']}
            )
            documents.append(document)
    yield documents
```

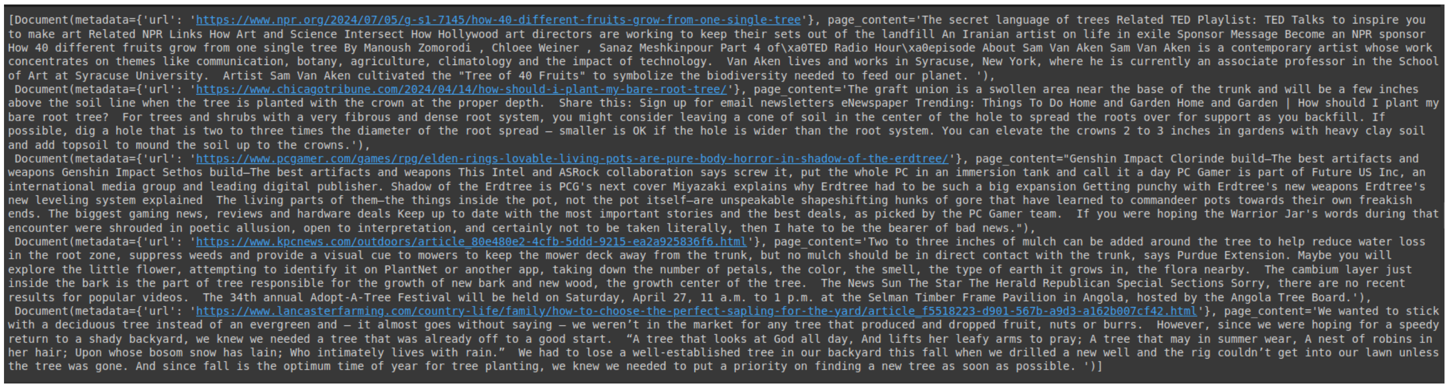
Since tavily's result is an iter._tree object, the first for loop has been used. List of Document object will be yield eventually.
In order to convert a function to be a 'runable' object where can be executed in the context of pipelines where tasks need to be chained, we use following code:

```
document_parsing = RunnableGenerator(document_parsing)
```

I think the hardest part of this computer assignmentwas indeed this part :).
Now we can create our search engine chain as follow:

```
search_engine_chain = tool | document_parsing
res = search_engine_chain.invoke({"query": "what is a Tree in computer science?"})
```

The output of this query has been illustrated in Figure 8: Attention: I have no idea, why Tavily is not respond quite



```
[Document(metadata={'url': 'https://www.npr.org/2024/07/05/g-s1-7145/how-40-different-fruits-grow-from-one-single-tree'}, page_content='The secret language of trees Related TED Playlist: TED Talks to inspire you
to make art Related NPR Links How Art and Science Intersect How Hollywood art directors are working to keep their sets out of the landfill An Iranian artist on life in exile Sponsor Message Become an NPR sponsor
How 40 different fruits grow from one single tree By Manoush Zomorodi , Chloee Weiner , Sanaz Meshkinpour Part 4 of\xa0TED Radio Hour\xa0episode About Sam Van Aken Sam Van Aken is a contemporary artist whose work
concentrates on themes like communication, botany, agriculture, climatology and the impact of technology.  Van Aken lives and works in Syracuse, New York, where he is currently an associate professor in the School
of Art at Syracuse University.  Artist Sam Van Aken cultivated the "Tree of 40 Fruits" to symbolize the biodiversity needed to feed our planet. '),
 Document(metadata={'url': 'https://www.chicagotribune.com/2024/04/14/how-should-i-plant-my-bare-root-tree/'}, page_content='The graft union is a swollen area near the base of the trunk and will be a few inches
above the soil line when the tree is planted with the crown at the proper depth.  Share this: Sign up for email newsletters eNewspaper Trending: Things To Do Home and Garden Home and Garden | How should I plant my
bare root tree?  For trees and shrubs with a very fibrous and dense root system, you might consider leaving a cone of soil in the center of the hole to spread the roots over for support as you backfill. If
possible, dig a hole that is two to three times the diameter of the root spread – smaller is OK if the hole is wider than the root system. You can elevate the crowns 2 to 3 inches in gardens with heavy clay soil
and add topsoil to mound the soil up to the crowns.'),
 Document(metadata={'url': 'https://www.pcgamer.com/games/rpg/elden-rings-lovable-living-pots-are-pure-body-horror-in-shadow-of-the-erdtree/'}, page_content="Genshin Impact Clorinde build—The best artifacts and
weapons Genshin Impact Sethos build—The best artifacts and weapons This Intel and ASRock collaboration says screw it, put the whole PC in an immersion tank and call it a day PC Gamer is part of Future US Inc, an
international media group and leading digital publisher. Shadow of the Erdtree is PCG's next cover Miyazaki explains why Erdtree had to be such a big expansion Getting punchy with Erdtree's new weapons Erdtree's
new leveling system explained  The living parts of them—the things inside the pot, not the pot itself—are unspeakable shapeshifting hunks of gore that have learned to commandeer pots towards their own freakish
ends. The biggest gaming news, reviews and hardware deals Keep up to date with the most important stories and the best deals, as picked by the PC Gamer team.  If you were hoping the Warrior Jar's words during that
encounter were shrouded in poetic allusion, open to interpretation, and certainly not to be taken literally, then I hate to be the bearer of bad news."),
 Document(metadata={'url': 'https://www.kpcnews.com/outdoors/article_80e480e2-4cfb-5ddd-9215-ea2a925836f6.html'}, page_content='Two to three inches of mulch can be added around the tree to help reduce water loss
in the root zone, suppress weeds and provide a visual cue to mowers to keep the mower deck away from the trunk, but no mulch should be in direct contact with the trunk, says Purdue Extension. Maybe you will
explore the little flower, attempting to identify it on PlantNet or another app, taking down the number of petals, the color, the smell, the type of earth it grows in, the flora nearby.  The cambium layer just
inside the bark is the part of tree responsible for the growth of new bark and new wood, the growth center of the tree.  The News Sun The Star The Herald Republican Special Sections Sorry, there are no recent
results for popular videos.  The 34th annual Adopt-A-Tree Festival will be held on Saturday, April 27, 11 a.m. to 1 p.m. at the Selman Timber Frame Pavilion in Angola, hosted by the Angola Tree Board.'),
 Document(metadata={'url': 'https://www.lancasterfarming.com/country-life/family/how-to-choose-the-perfect-sapling-for-the-yard/article_f5518223-d901-567b-a9d3-a162b007cf42.html'}, page_content='We wanted to stick
with a deciduous tree instead of an evergreen and – it almost goes without saying – we weren't in the market for any tree that produced and dropped fruit, nuts or burrs.  However, since we were hoping for a speedy
return to a shady backyard, we knew we needed a tree that was already off to a good start.  "A tree that looks at God all day, And lifts her leafy arms to pray; A tree that may in summer wear, A nest of robins in
her hair; Upon whose bosom snow has lain; Who intimately lives with rain."  We had to lose a well-established tree in our backyard this fall when we drilled a new well and the rig couldn't get into our lawn unless
the tree was gone. And since fall is the optimum time of year for tree planting, we knew we needed to put a priority on finding a new tree as soon as possible. ')]
```

Figure 8: Result of Search Engine

well. When I tried it at first, the response was very good.After finishing project, I have notice responses are not relevant to my query. I have even tried multiple combination of 'TavilySearchResults'. For example I have set the include_domain to geeksforgeeks, acm, arxiv, wikipedia and codeforces but it always find irrelevant responses!

7 Part Six: Relevancy Checker Chain

7.1 Prompt

We have written a precise prompt where LLM to evaluate a list of documents and determine each document is either ’relevant’ or ’irrelevant’ based on the given query. We have also emphasised that it should return a list indicating relevance status to each document.

```
relevancy_prompt_template = (
    "You are an expert in determining the relevancy of documents to a given query.
       ↪  "
    "Given a query and a list of documents, respond with a list where each element
       ↪  corresponds to the relevancy of each document based on the given query.
       ↪  "
    "If a document is relevant to the query, respond with the word 'relevant'. "
    "If a document is not relevant, respond with the word 'irrelevant'. "
    "Provide the responses as a list of words containing only either 'relevant' or
       ↪  'irrelevant', in the same order as the provided documents list.\n"
    "{output_instructions}\n"
    "Query: {query}\n"
    "Documents: {documents}"
)
```

7.2 Model

The model is same LLM we used in previous sections.

7.3 Post-Process

The LLM’s output should map to a framework. Then we have implemented following parser:

```
class ChosenWord(BaseModel):
    word: List[Literal['relevant', 'irrelevant']] = Field(description="the words
       ↪ that was chosen by LLM in relevancy check stage")

relevancy_parser = PydanticOutputParser(pydantic_object=ChosenWord)
print(relevancy_parser.get_format_instructions())
```

It has been implemented like the parser in router parser.

7.4 Why Relevancy Chain Is Necessary?

To answer this question, for more information there would be another chain that is complementary for this.That chain’s duty is re-writing the query.
We often query on search engines like google in order to find a what we want. So if search engine just return 5 most relevant documents based on our query, it might not answer our question.Moreover, if user’s query was bad thus retrieval results from vector store would be also bad.

So we can get help of LLM to just re-write the user query muliple times to user get what they wanted.(But we did not implement this complementary re-writing chain)
Two example provide for you to simulate what I just said earlier.

```python
relevancy_check_chain = relevancy_prompt | llm | relevancy_parser
docs = search_engine_chain.invoke({"query": "what is a chat bot?"})
res = relevancy_check_chain.invoke({
    "query": "my name is pouya.",
    "documents": docs,
    "output_instructions": relevancy_parser.get_format_instructions()
})
```

The relevancy chain's results are:

$$\text{result} = \text{ChosenWord}(\text{word} = ['\text{irrelevant}','\text{irrelevant}','\text{irrelevant}','\text{irrelevant}','\text{irrelevant}'])$$

The second example:

```python
res = relevancy_check_chain.invoke({
    "query": "what is a chat bot?",
    "documents": docs,
    "output_instructions": relevancy_parser.get_format_instructions()
})
```

The chain results:

$$\text{result} = \text{ChosenWord}(\text{word} = ['\text{relevant}','\text{relevant}','\text{relevant}','\text{relevant}','\text{relevant}'])$$

8 Part Seven: Fallback Chain

8.1 Prompt

We have crafted a prompt where specify the LLM, their limitation to answer the queries. We just modify prompt in workshop based on our needs:

```
fallback_prompt = ChatPromptTemplate.from_template(
    (
        "You are an expert in Natural Language Processing and computer science
           ↪ created by the NLP Staff Course at the University of Tehran.\n"
        "Do not respond to queries that are not related to Natural Language
           ↪ Processing or computer science.\n"
        "If a query is not related to Natural Language Processing or computer
           ↪ science, acknowledge your limitations.\n"
        "Provide concise responses to only NLP or computer science-related queries
           ↪ .\n\n"
        "Current conversation:\n\n{chat_history}\n\n"
        "human: {query}"
    )
)
```

This will answer just the queries are related to NLP or CS.

8.2 Model

For this chain we consider to have same model but with a higher temperature since this is not a classification task to have just a precise and predictable answer,instead we want to have more variation and diverse answers by LLM.

```
fall_back_llm = ChatTogether(
    model="meta-llama/Llama-3-70b-chat-hf",
    temperature= 0.2
    )
```

8.3 Parser

The string output parser will eventually add to the last component of chain.(We used same logic implemented in workshop for chat histories.)

```
fallback_chain = (
    {
        "chat_history": lambda x: "\n".join(
            [
                (
                    f"human: {msg.content}"
                    if isinstance(msg, HumanMessage)
                    else f"AI: {msg.content}"
                )
                for msg in x["chat_history"]
            ]
        ),
        "query": itemgetter("query") ,
    }
    | fallback_prompt
    | fall_back_llm
    | StrOutputParser()
)
```

9 Part Eight: Generate With Context Chain

9.1 Prompt

This prompt should inform the LLM to just answer the query based on provided context and also this LLM must not use any knowledge outside the given context.

```
generate_with_context_template = (
    "You are a knowledgeable assistant specializing in providing accurate and
       ↪ context-specific answers. "
    "You must answer the query below based on the provided context. "
    "If the provided context is not relevant to the query, DO NOT answer based on
       ↪ your own knowledge"
    "Do not use any information or knowledge outside of the given context to
       ↪ answer the query.\n\n"
    "Context: {context}\n\n"
    "Query: {query}"
)
generate_with_context_prompt = ChatPromptTemplate.from_template(
   ↪ generate_with_context_template)
```

9.2 Model

Model are as same as previous section(temperature 0).

9.3 Parser

String parser used in previous section also used here. The chain is:

```
generate_with_context_chain = generate_with_context_prompt | llm | StrOutputParser
   ↪ ()
```

We have given two example to evaluate this chain and prompt.

```
query = "what is a chatbot?"
context = retriever.get_relevant_documents(query)
response =  generate_with_context_chain.invoke({"query": query, "context": context
   ↪ })
Markdown(response)
```

The response was:
"According to the provided context, a chatbot is a "conversational system designed to mimic the appearance of informal human conversation.""
The second query:

```
query = "what is a chatbot?"
context = "My name is Pouya"
response =  generate_with_context_chain.invoke({"query": query, "context": context
   ↪ })
```

```
Markdown(response)
```

The response is:
Nice to meet you, Pouya! Unfortunately, I don’t have any information about chatbots within our conversation so far. Since I’m only allowed to answer based on the provided context, I won’t be able to define what a chatbot is. Maybe you can provide more context or information about chatbots, and I’ll do my best to help?
As you can see the queries are answered properly.(But since we have filter docs node, thus relevant documents will reach to this node.Moreover, query ”My name is Pouya” will reach to Fallback node.Therefore this query and context are not realistic and we just wanted to evaluate this part so we used them.)

10 Part Nine: Preparation Graph With LangGraph

10.1 State

Each node will change that state.Moreover, using a dictionary as a state will keep track of the data required by the various nodes in graph:

```python
class AgentState(TypedDict):
    query: str
    chat_history: list[BaseMessage]
    generation: str
    documents: list[Document]
```

10.2 Router Node

The duty of this node is to get a query and based on the router's chain defined in section 5(Part 4) it decide what edge it should go:

```python
def router_node(state: dict):
    query = state["query"]
    try:
        respond = question_router.invoke({"query": query,
                                      "output_instructions": question_router_parser.
                                          ↪ get_format_instructions()
                                                })
    except Exception:
        print('Exception in getting respond')

    try:
        chosen_tool = respond.tool_name.lower()
    except Exception:
        return "LLMFallback"

    if chosen_tool == 'none':
        print("-- No tool called --")
        return 'LLMFallback'
    if chosen_tool == 'vectorstore':
        print("-- Routing to VectorStore --")
        return "VectorStore"

    if chosen_tool == 'searchengine':
        print("-- Routing to SearchEngine --")
        return "SearchEngine"
```

This code will try to get respond from question router chain. Based on decision router made, one of these three edges will be returned:

1. If chosen tool is 'none': The edge 'LLMFallback' will be taken.

2. If chosen tool is ’vectorstore’, ’VectorStore’ edge will be taken.
3. If chosen tool is ’searchengine’, ’SearchEngine’ edge will be taken.

10.3 Vector Store Node

This node will retrieve documents and find related chunks.

```python
def vector_store_node(state: dict):
    query = state["query"]
    return {"documents" : retriever.invoke(input=query)}
```

10.4 Search Engine Node

This node will get a query from user and find the the related documents from internet using search engine chain.

```python
def search_engine_node(state: dict):
    query = state["query"]
    return {"documents" :search_engine_chain.invoke({"query": query})}
```

10.5 Filter Docs Node

From given a query and a list of Documents object, this node will use relevant status list obtained by relevancy check chain to filter given documents.

```python
def filter_docs_node(state: dict):
    query = state["query"]
    documents = state["documents"]
    releated_list = relevancy_check_chain.invoke({
        "query": query,
        "documents": documents,
        "output_instructions": relevancy_parser.get_format_instructions()
        })
    filtered_docs = []
    for index,word in enumerate(releated_list.word):
        temp = word.lower()
        if temp == 'relevant':
            filtered_docs.append(documents[index])
    return {"documents": filtered_docs}
```

10.6 Fallback Node

Using chat history and query, this node will generate final answer for user.

```python
def fallback_node(state: dict):
    query = state["query"]
    chats = state["chat_history"]
    generation = fallback_chain.invoke({
        "query": query,
        "chat_history":chats
    })
    return {"generation": generation}
```

Generate With Context Duty of this node is generate an answer based on given query and provided context(Where context are filtered documents provided by filter docs node).

10.7 Workflow

Workflow of this graph: Router node has three edges based on the decision this node will make, some edge will be taken:

```
workflow = StateGraph(AgentState)

workflow.add_node("vector_store", vector_store_node)
workflow.add_node("fallback", fallback_node)
workflow.add_node("generate_with_context", generate_with_context_node)
workflow.add_node("filter_docs", filter_docs_node)
workflow.add_node("search_engine", search_engine_node)
workflow.set_conditional_entry_point(
    router_node,
    {
        "LLMFallback": "fallback",
        "VectorStore": "vector_store",
        "SearchEngine": "search_engine",
    },
)
```

Now we know that if filtered docs is empty, we should go to search engine node, otherwise should go to generate with context node.So in order to create a conditional edges based on the information above, we will utilize the following code:

```
def should_continue(state: dict):
    documents = state.get("documents", [])
    if len(documents) == 0:
        print("-- filtered was empty --")
        return "SearchEngine"
    print("-- filtered was not empty --")
    return "Generate"
```

This will determine next node from filter with following code:

```
workflow.add_conditional_edges(
    "filter_docs",
    should_continue,
    {
        "SearchEngine": "search_engine",
        "Generate": "generate_with_context"
    }
)
```

Now it is time to proceed into make edges:

```
workflow.add_edge("vector_store", "filter_docs")

workflow.add_edge("search_engine", "filter_docs")

workflow.add_edge("generate_with_context", END)

workflow.add_edge("fallback", END)
```

Now we are finished, let's see our graph is correctly implemented or not.

```
app = workflow.compile(debug=False)
plot = app.get_graph().draw_mermaid_png()

with open("plot.png", "wb") as fp:
    fp.write(plot)
from io import BytesIO
from PIL import Image
from IPython.display import display

img = Image.open(BytesIO(plot))
display(img)
```

The result of this code has been illustrated in Figure 9:

10.8 Chats

Now we can give some queries and talk with it:

```
response = app.invoke({"query": "who is the president of Iran?", "chat_history":
   ↪ []})
Markdown(response["generation"])
```

The result is shown Figure 10: The second chat:

```
response = app.invoke({"query": "what is a chat bot ?", "chat_history": []})
Markdown(response["generation"])
```

The answer has been illustrated in Figure 11: The last chat:

```
response = app.invoke({"query": "what is a Tree in computer science?", "
   ↪ chat_history": []})
Markdown(response["generation"])
```

The respond is depicted in Figure 12: But note that, sometimes search engine sometimes do not work correctly and we will stuck between search engine and filter docs since search engine do not work well.As we said in section 6(Part 5), it can be solved by a query re-writing node.

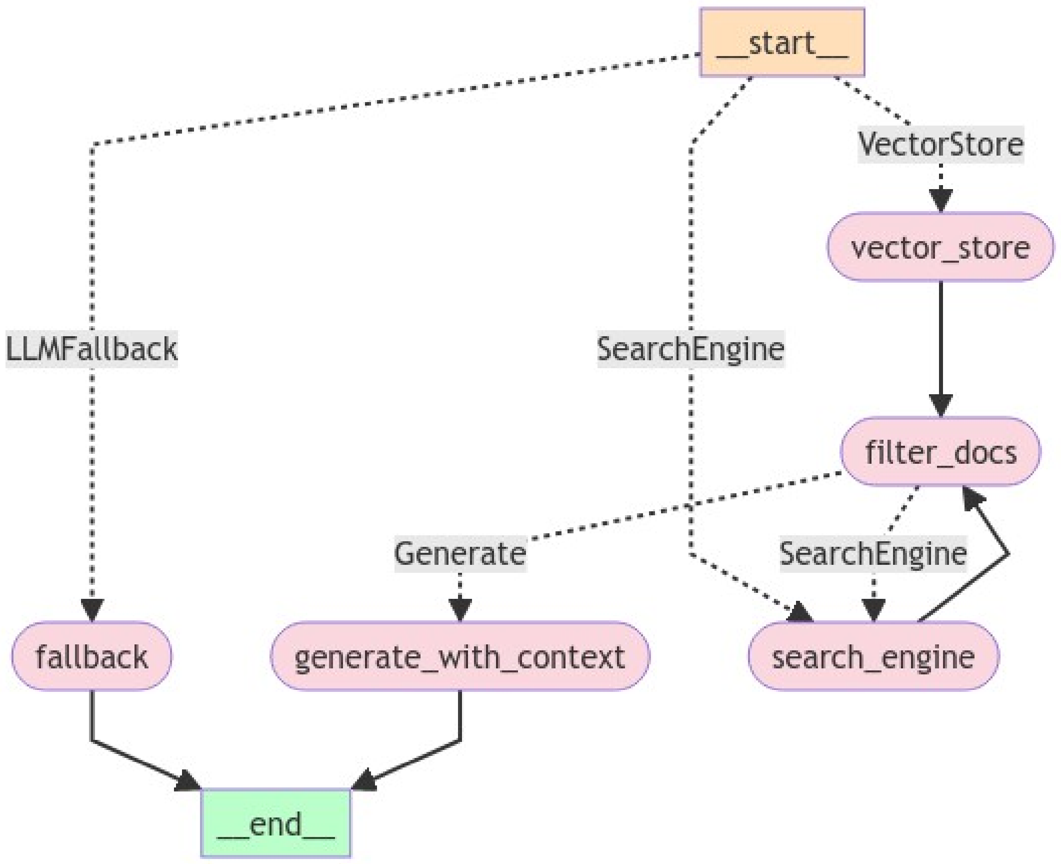
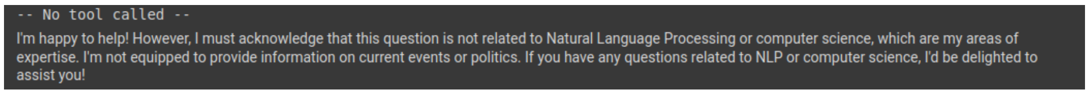


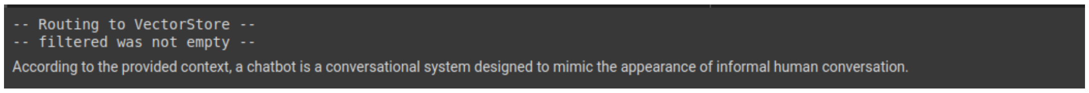
Figure 9: Graph



```
-- No tool called --
```

I'm happy to help! However, I must acknowledge that this question is not related to Natural Language Processing or computer science, which are my areas of expertise. I'm not equipped to provide information on current events or politics. If you have any questions related to NLP or computer science, I'd be delighted to assist you!

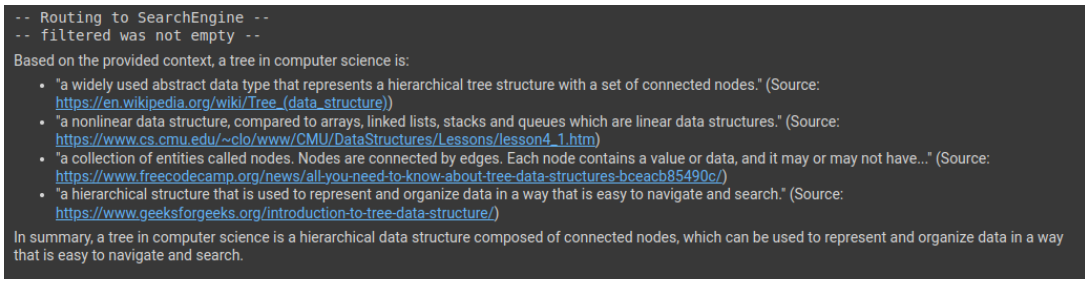
Figure 10: Chat 1



```
-- Routing to VectorStore --
-- filtered was not empty --
```

According to the provided context, a chatbot is a conversational system designed to mimic the appearance of informal human conversation.

Figure 11: Chat 2



```
-- Routing to SearchEngine --
-- filtered was not empty --
```

Based on the provided context, a tree in computer science is:

- "a widely used abstract data type that represents a hierarchical tree structure with a set of connected nodes." (Source: https://en.wikipedia.org/wiki/Tree_(data_structure))
- "a nonlinear data structure, compared to arrays, linked lists, stacks and queues which are linear data structures." (Source: https://www.cs.cmu.edu/~clo/www/CMU/DataStructures/Lessons/lesson4_1.htm)
- "a collection of entities called nodes. Nodes are connected by edges. Each node contains a value or data, and it may or may not have..." (Source: https://www.freecodecamp.org/news/all-you-need-to-know-about-tree-data-structures-bceacb85490c/)
- "a hierarchical structure that is used to represent and organize data in a way that is easy to navigate and search." (Source: https://www.geeksforgeeks.org/introduction-to-tree-data-structure/)

In summary, a tree in computer science is a hierarchical data structure composed of connected nodes, which can be used to represent and organize data in a way that is easy to navigate and search.

Figure 12: Chat 3

References

[1] Hugging Face. *MTEB Leaderboard*. `https://huggingface.co/spaces/mteb/leaderboard`. Accessed: 2024-07-07. 2024.

[2] LangChain. *Document Transformers*. `https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/`. Accessed: 2024-07-07. 2024.

[3] LangChain. *Ensemble Retriever*. `https://python.langchain.com/v0.1/docs/modules/data_connection/retrievers/ensemble/`. Accessed: 2024-07-07. 2024.

[4] LangChain. *Recursive Text Splitter*. `https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/recursive_text_splitter/`. Accessed: 2024-07-07. 2024.