Rafael André Henriques Ferreira

Bachelor of Computer Science and Engineering

Tracking Context in Conversational Search: From Utterances to Neural Embeddings

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering

Adviser: Dr. João Miguel da Costa Magalhães,
Associate Professor, NOVA University of Lisbon

Co-adviser: Dr. David Semedo,
Assistant Researcher, NOVA University of Lisbon

Examination Committee

Chair: Dr. Vasco Miguel Moreira Amaral, NOVA University of Lisbon
Rapporteur: Dr. Bruno Emanuel da Graça Martins, Instituto Superior Técnico
Member: Dr. João Miguel Costa Magalhães, NOVA University of Lisbon

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

February, 2021

Tracking Context in Conversational Search:
From Utterances to Neural Embeddings

This document was created using the (pdf)LaTeX processor, based on the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Para os meus pais.

Acknowledgements

First and foremost, I would like to express my gratitude to my advisor Dr. João Magalhães for all of the support, encouragement, and expertise, along with his availability during the course of this thesis.

Second, I would like to thank my co-advisor, Dr. David Semedo, for all his help and valuable knowledge, and all of the people in the Nova-Search group for their feedback.

Furthermore, I would like to thank Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa for not only giving me the tools to complete this thesis but also for the lessons that it taught me to become a better person and professional. Additionally, I want to thank the project GoLocal ref. CMUP-ERI/TIC/0046/2014 and all its participants for funding this project.

I am also thankful to all my friends that accompanied me during not only these five years but also way far back when we didn't even know what we were going to do.

Finally, I would like to thank my family for all their support and especially my parents for all the patience, encouragement, and help during all my life. ☺

Abstract

The use of conversational assistants is becoming increasingly more popular among the general public, pushing the research towards more advanced and sophisticated techniques. Hence, there are currently a number of research opportunities to extend the comprehension and applicability of these tasks in everyday systems.

These conversational assistants are capable of performing various tasks, such as chitchatting, internal device functions (e.g., setting up an alarm), and searching for information. In the last few years, the interest in conversational search is increasing, not only because of the generalization of conversational assistants but also because conversational search is a step forward in allowing a more natural interaction with the system. To build a system such as this, many components need to work together, since in a conversation, the importance of context is paramount to retrieve the best answers to the user's questions.

In this thesis, the focus was on developing a conversational search system that aims to help people search for information in a natural way. In particular, this system must be able to understand the context where the question is posed, tracking the current state of the conversation and detecting mentions to previous questions and answers. We achieve this by using a context-tracking component based on neural query-rewriting models. Another crucial aspect of the system is to provide the most relevant answers given the question and the conversational history. To achieve this objective, we used state-of-the-art retrieval and re-ranking methods and expanded their architecture to use the conversational context.

The results obtained with the system developed achieved state-of-the-art when compared to the baselines present in TREC Conversational Assistance Track (CAsT) 2019.

Keywords: Conversational Search, Multi-turn Question Answering, Conversational Context, Information Retrieval, Query Rewriting, Ranking, Natural Language Processing

Resumo

O uso de assistentes conversacionais está a tornar-se cada vez mais popular entre o público em geral, levando à investigação de técnicas mais avançadas e sofisticadas. Consequentemente, existem atualmente várias oportunidades de investigação para estender a compreensão e aplicabilidade destas tarefas em sistemas do quotidiano.

Estes assistentes são capazes de efetuar várias tarefas como, por exemplo: ter uma conversa informal, efetuar funções internas ao dispositivo (e.g. colocar um alarme), e pesquisar por informação. Nos últimos anos, o interesse em pesquisa conversacional tem estado a aumentar, não só pela generalização dos assistentes conversacionais, mas também devido a ser um passo em frente para permitir uma interação mais natural com o sistema. Para construir um sistema deste tipo, vários componentes têm de trabalhar em conjunto, uma vez que numa conversa o contexto é da maior importância para recuperar as melhores respostas para as perguntas do utilizador.

Nesta tese, o foco foi desenvolver um sistema de pesquisa conversacional para ajudar as pessoas a pesquisar por informação de uma forma natural. Em particular, este sistema tem de ser capaz de compreender o contexto onde a questão é colocada, fazendo *tracking* do estado atual da conversa e detetando menções a perguntas e respostas anteriores. Com esse objetivo, desenvolvemos um componente de *tracking* de contexto baseado em modelos neuronais de reescrita de perguntas. Outro aspeto crucial deste sistema é fornecer as respostas mais relevantes dada uma pergunta e o histórico da conversa. Para alcançar este objetivo, utilizámos modelos do estado-da-arte em recuperação de informação e *re-ranking* e expandimos estas arquiteturas de modo a utilizarem o contexto da conversa.

Os resultados obtidos com o sistema desenvolvido atingiram resultados do estado-da-arte quando comparados às *baselines* submetidas no TREC Conversational Assistance Track (CAsT) 2019.

Palavras-chave: Pesquisa Conversacional, Perguntas e Respostas Multi-turn, Contexto da Conversa, Recuperação de Informação, Reescrita de Perguntas, Ranking, Processamento de Linguagem Natural

Contents

List of Figures

List of Tables

Acronyms

AT	Adversarial Training
BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Understudy
CAR	Complex Answer Retrieval
CAsT	Conversational Assistance Track
CNN	Convolutional Neural Network
ConvBERT	Conversational BERT
Conv-KNRM	Convolutional Kernel-based Neural Ranking Model
CoQA	Conversational Question Answering
DCG	Discounted Cumulative Gain
DeepCT	Deep Contextualized Term Weighting
DRMM	Deep Relevance Matching Model
FFNN	Feed-Forward Neural Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HAE	History Answer Embedding
HAM	History Attention Mechanism
HRED	Hierarchical Recurrent Encoder-Decoder
IR	Information Retrieval
KD	Knowledge Distillation
K-NRM	Kernel-based Neural Ranking Model

KB	Knowledge Base
LMD	Language Model Dirichlet
LMJM	Language Model Jelinek-Mercer
LSTM	Long-Short Term Memory
MAP	Mean Average Precision
MLD	Masked Language Model
MRR	Mean Reciprocal Rank
MS MARCO	Microsoft Human Generated MAchine Reading COmprehension Dataset
NDCG	Normalized Discounted Cumulative Gain
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
NSP	Next Sentence Prediction
PosHAE	Positional History Answer Embedding
QA	Question-Answering
QuAC	Question Answering in Context
REST	Representational State Transfer
RM	Relevance Model
RNN	Recurrent Neural Network
RoBERTa	Robustly optimized BERT approach
ROUGE	Recall Oriented Understudy for Gisting Evaluation
RRF	Reciprocal Rank Fusion
SQuAD	Stanford Question Answering Dataset
T5	Text-to-Text Transfer Transformer
TPU	Tensor Processing Unit
TREC	Text REtrieval Conference
WaPo	Washington Post
WER	Word Error Rate

Chapter 1

Introduction

1.1 Context and Motivation

Conversational search systems are an emerging research topic and a step forward from traditional search engines, allowing a more natural interaction with an intelligent agent. The potential of these kinds of systems is very high, with applications to various domains, such as e-commerce, medical question answering, and others. Conversational assistants, such as Siri, Alexa, and Bixby, have been around for a few years, with major improvements since their creation. However, their ability to support conversational search is up to this time still limited, not supporting or basing the context on named entity recognition [5]. These limitations need to be addressed with stronger context-tracking models that understand the underlying context of the conversation in order to retrieve relevant information.

Nowadays, the emergent interest in conversational search, as evidenced in the recent international conferences ECIR 2020[1], SIGIR 2020[2], and ACL 2020[3], as well as the recent efforts to construct adequate datasets (e.g. TREC CAsT [10]), enables the opportunity to explore the different facets of the creation and evaluation of conversational search systems.

We also add that this thesis was developed in the context of the project *GoLocal - From Monitoring Global Data Flows to Recommendation Based on Context*[4], where one of the goals is to research robust context-aware natural language processing (NLP) models to work in open domains.

[1] https://ecir2020.org/
[2] https://sigir.org/sigir2020/
[3] https://acl2020.org/
[4] https://www.fct.pt/apoios/projectos/consulta/vglobal_projecto.phtml.pt?idProjecto=139253&idElemConcurso=8677

1.2 Problem Definition and Objective

Following Jurafsky and Martin [24], conversational systems can be divided into three main categories:

- **Chatbots** - The agent tries to mimic a real user, conversing as seamlessly as possible, with the objective of keeping the user engaged.
- **Task Completion** - Information is provided to the agent to complete a specific task. Typically follows pre-determined steps or uses a slot filling pattern to achieve the goal.
- **Question Answering (QA)** - The agent returns direct and succinct answers to the user's queries, analyzing, and retrieving knowledge from various sources to provide the best answers to the user.

We depart from chatbots, which are mainly used for chitchat, and diverge from task completion systems, that generally have a limited scope of actions, by focusing on an open-domain, retrieval-based, conversational question-answering scenario where the questions can be about any topic.

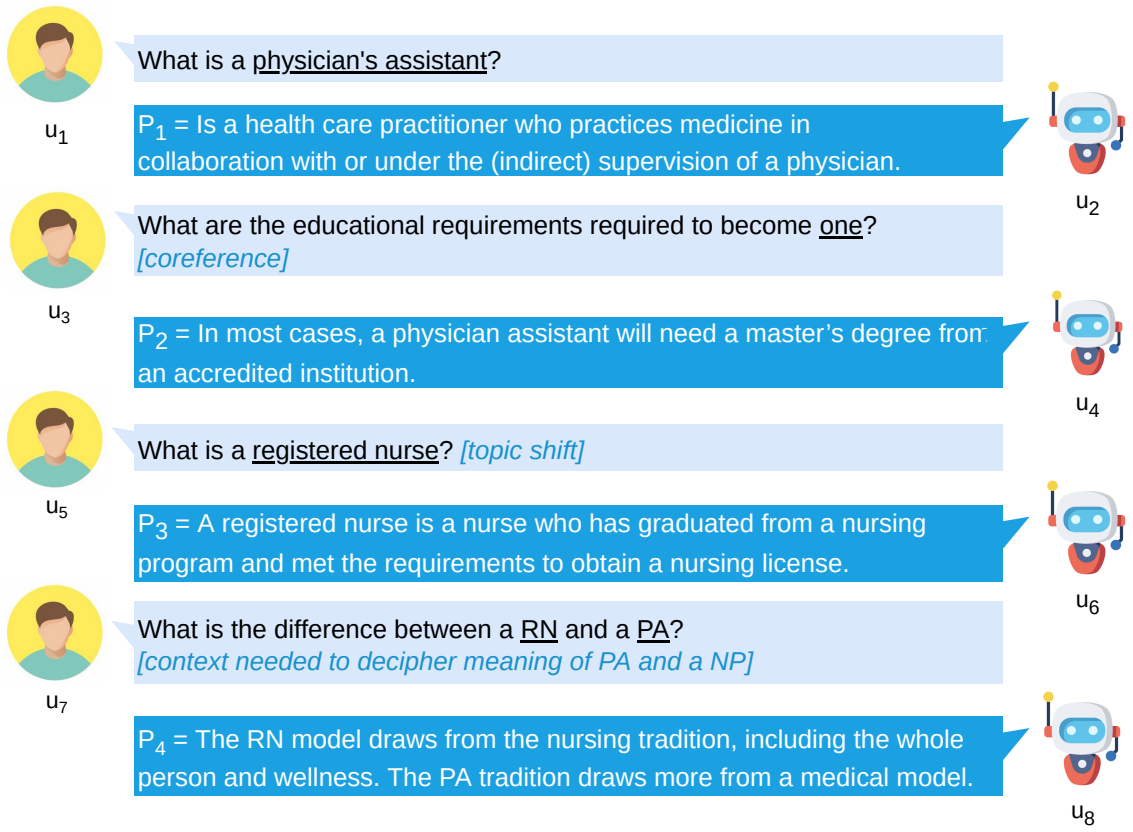


Figure 1.1: Example conversation between a user and a conversational search system.

This thesis is focused on search and answer selection (QA) in a conversational search setting. This more challenging scenario departs from typical single-turn QA and search systems, where the system only needs to answer the current question of the user without any notion of previous questions/answers. An example of a conversation between a user and a conversational search system is represented in figure 1.1. As it can see in the example, conversational search has various nuances that need to be addressed in order to provide a relevant response to the user's queries. In particular, we highlight the use of state and context from previous questions and answers. This is evidenced in figure 1.1 in the second question, where the user uses the pronoun "one" to refer to "physician's assistant" mentioned in the previous query. Another important remark is the system's ability to detect context shifts, as seen in question 3, answering the current question, which is about a different topic. As a final example, question 4 shows a complex task, where the system needs to resolve the ambiguities in the question, using the conversational context to resolve the meaning of "RN" and "PA", which correspond to "registered nurse" and "physician's assistant", respectively. All these subtleties need to be addressed for effective conversational search, allowing the user to search for information about open topics in a quick and natural way.

The formal definition of conversational search provided in [10] is: given a series of natural language utterances (u) about a given topic T:

$$T = \{u_1, \dots u_i, \dots u_n\}, \tag{1.1}$$

the task is to identify relevant passages P_i for each turn (user utterance) u_i that satisfies the information needs in round i given the conversation context, i.e., the previous conversation turns $u_1, \dots, u_{i-1}$.

As stated, the conversational context assumes a central role, therefore, **the objective of this thesis is to research methods that can improve the search results by tracking the context present in (i) the conversation utterances in natural language and (ii) in the neural embeddings of a conversational neural-state tracker**.

1.3 The "Anatomy" of a Conversational Search Agent

To tackle the conversational search task, various challenges need to be addressed to provide a correct final result. Figure 1.2 represents a simple overview of the system developed, where the main components are the following:

- **Knowledge-Base** - The knowledge-base contains all the information that the agent knows and can use to answer questions. In the context of this thesis, the knowledge-base corresponds to passages from the Wikipedia and Web.
- **Retrieval Model** - This component is responsible for the first search on the knowledge-base (index), comprised of possibly millions of documents. It needs to be fast and to achieve a high recall to provide the re-ranker with as many relevant documents

as possible. This was accomplished using well-established information retrieval models.

- **Re-Ranker** - After the initial retrieval step, it is obtained a smaller list of documents, in the order of a few thousand. These documents are then re-ordered by a more computationally complex algorithm that aims to put in the top the most relevant documents, resulting in a better ranking. This was addressed using state-of-the-art neural models that have an understanding of the interactions between the words in the query and documents.
- **State Tracker** - This is the component responsible for handling the context and history of the conversation. In this component, two approaches can be followed: (i) query rewriting to convert conversational queries to context-independent queries, and (ii) extend the previously mentioned re-rankers to create context-based ranking architectures that use the context present in the conversation.

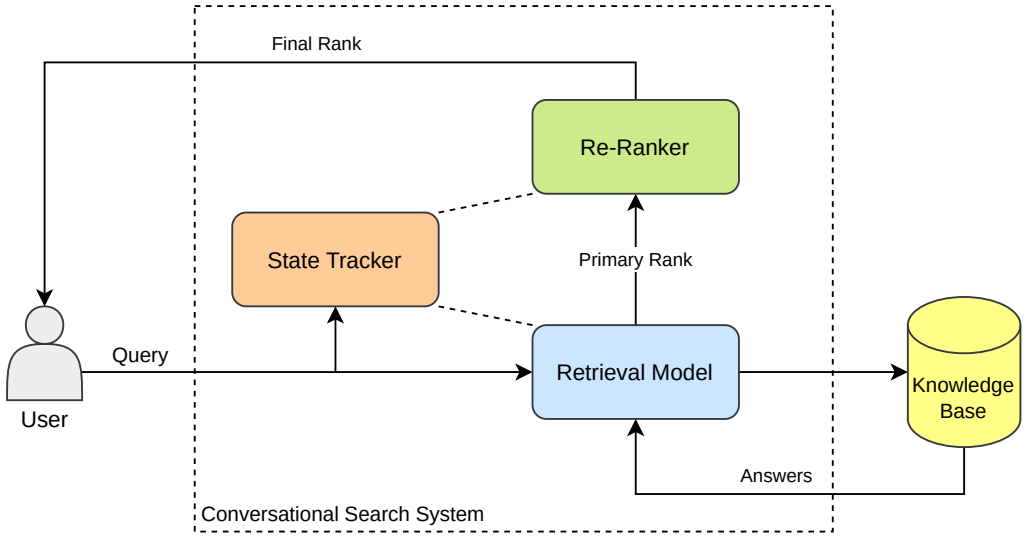


Figure 1.2: Overview of a conversational search system architecture. The dashed lines indicate that the components can retrieve and provide information to the state tracker.

1.4 Contributions

As a result of the work performed in this thesis, we highlight the following contributions:

- A thorough investigation of how the conversational context can be explored as a query rewriting problem (chapter 4) and as a neural state tracking problem (chapter 5).
- We propose two new architectures based on RNNs [2, 20] and Memory Networks [54] that bring state-of-the-art neural re-ranking models to the conversational search domain (chapter 5).

- A systematic and highly comprehensive experimental evaluation of each of the conversational search system components (chapter 6).
- The methods developed achieved state-of-the-art results when compared to the baselines submitted to TREC CAsT 2019 [10] (chapter 6).

In terms of scientific dissemination, the following submissions were done in the context of this thesis:

- Submission to TREC CAsT 2020 [9] with the name: "*NOVA at TREC 2020 Conversational Assistance Track*", and authors: Rafael Ferreira, David Semedo, and João Magalhães.
- Full paper accepted in ECIR 2021[5] titled: "*Open-Domain Conversational Search Assistant with Transformers*" [16], and authors: Rafael Ferreira, Mariana Leite, David Semedo, and João Magalhães.
- Submission under review: "*Knowledge-driven Answer Generation for Conversational Search*", with authors: Mariana Leite, Rafael Ferreira, David Semedo, and João Magalhães.

1.5 Document Structure

The remaining of the document is divided in the following chapters:

- Chapter 2, Related Work - Introduces the concepts related to conversational search and presents a discussion of the techniques, models, and algorithms currently used by the scientific community.
- Chapter 3, Indexing and First-Stage Retrieval - Presents traditional indexing strategies and information retrieval models, as well as recent neural document expansion techniques.
- Chapter 4, Conversational Context as Query Rewriting - Describes various query rewriting techniques that use the conversational context to convert a conversational query into a context-independent query.
- Chapter 5, Conversational Context-Aware Neural Ranking - Details the developed neural ranking models and the extensions performed to them to make use of the conversational context in order to re-rank the results retrieved by a typical information retrieval model.
- Chapter 6, Evaluation - Provides an extensive evaluation and discussion of the results obtained by the various components of the system in a conversational search scenario.

[5] https://www.ecir2021.eu/accepted-papers/

- Chapter 7, Conclusions and Future Work - Presents the conclusions of this thesis and avenues for future work.

Chapter 2

Related Work

In this chapter, we present the essential concepts related to conversational search and the current methodologies and algorithms used in the various components of such systems. In the first section, we introduce the concept of dialogue systems, and in the second, we present the common datasets and evaluation metrics. The third section introduces the current state-of-the-art Transformer models used to compute contextual word embeddings. In the fourth, we introduce indexing techniques and methods used to perform the first-stage retrieval step. The fifth section introduces various re-ranking algorithms, followed by the sixth section, where we present conversational state tracking methods. The subsequent section introduces several conversational systems, and the final section shows a critical summary of this chapter.

2.1 Dialogue Systems

The objective of a dialogue system is to converse with the user. We can categorize these systems into three main categories: chatbots, task completion, and question answering systems [24].

Chatbots are one of the simplest forms of dialogue systems. There exist two categories of chatbots: rule-based and corpus-based. A rule-based chatbot is characterized by pattern-action rules, meaning that it uncovers patterns in the utterance and applies specific pre-defined actions. A corpus-based chatbot, instead of hand-built rules, uses large amounts of data to derive its answers.

Task completion dialogue systems, on the other hand, have the goal of helping the user complete a specific task. These types of systems can extract from a sentence what is called a frame that represents the types of interactions the system allows. Each frame is

composed of a collection of slots, where each can take a specific value. The objective of the system is to determine the intent of the user (frame to use) and then fill the slots. The more recent task-based dialogue systems use dialogue acts [24], where the utterances of a user or agent are considered actions that can change the state of both the user and the system, and so, the state of the conversation. These systems have a more advanced frame-based architecture called dialogue-state [24]. This architecture uses natural language understanding (NLU) to extract slot fillers, a dialogue state tracker to monitor the current state, and a dialogue policy to decide which action to execute next.

Question answering dialogue systems have the goal of providing concise answers to typically factoid questions. From an architectural point of view, these systems can be of two types [24]: Knowledge-based (KB) or Information Retrieval (IR) based. In a KB architecture, the system answers the natural language questions by mapping them to a query on a structured database (the knowledge base). The most simple formulation for this is converting a query to a triple, such as (subject, predicate, object), and then perform the query over the database. With an IR-based architecture, the goal is to retrieve the answer to a question from a collection of documents. To achieve this, it can first apply a query formulation task, creating a list of tokens to send to the information retrieval system. After the query formulation step, the system performs the document/passage retrieval, resulting in a set of documents/passages ranked by their relevance to the query.

This thesis, in particular, focuses on a retrieval-based architecture for conversational search. Conversational search differentiates itself from simple QA because of its multi-turn setting, where various queries are performed in a sequence with the model retrieving from a large collection, a ranked list of answers in each turn.

Ethics also need to be considered when designing these types of systems. It is important to understand that the data being used can include certain biases, so a careful evaluation of the data and the implications of the system being deployed needs to be conducted. An illustrative example is Microsoft's Tay chatbot. This chatbot was shutdown in less than a day because it began posting messages of inappropriate content due to learning with "toxic" user interactions in social media [35].

2.2 Evaluation of Conversational Agents

2.2.1 Datasets

Datasets are one of the most important aspects when evaluating and building a conversational search agent. The recent advances made in this area are largely due to the proliferation and availability of these large-scale datasets. Each one has its own set of characteristics, ranging from the type of text it has (articles, news, conversations, fictional, and narrative) to the number of speakers and the way the data is collected. Although being large-scale, these datasets must also have quality and be aligned with the task at

hand because having a large dataset that doesn't fit the specifications of our task is not as useful.

In this section, we will explain the content, scale, way the data is collected, and the usefulness of some of the more relevant datasets used in retrieval and conversational settings.

2.2.1.1 Retrieval Datasets

A retrieval dataset is characterized by a set of queries, along with a typically large set of documents/passages from which to retrieve an answer.

The TREC Washington Post Corpus (WaPo) [22] is a dataset provided by the American News Agency Washington Post. This dataset contains 608.108 news articles and blog posts that range from a period of 5 years, from January 2012 to August 2017. The articles are available in JSON format and were broken into multiple paragraphs, each one with a unique identifier.

TREC CAR (Complex Answer Retrieval) [13] is a dataset that contains a corpus of over 20 million paragraphs collected from a 2016 snapshot of Wikipedia. From this snapshot, the researchers discarded templates, talk pages, portals, disambiguation, redirect, and category pages, as well as articles with categories that indicate people, organizations, music, books, films, events, and lists. The paragraphs were then deduplicated and given a unique identifier to create a passage ranking task. The task of passage raking is defined as given a topic outline (query), which is defined as the concatenation of a Wikipedia article title with the title of one of its sections, retrieve a ranking of relevant passages, where the ground truth are the actual paragraphs of that section.

The MS MARCO [36] dataset or Human Generated MAchine Reading COmprehension Dataset is a dataset comprised of more than 1 million questions where each one has a human-generated answer. The questions are sampled from Bing's search logs, filtering out the non-question queries. These questions are not all well-formed, and so they can be ambiguous or contain typographical and other errors, requiring systems to be robust enough to understand and answer these questions. After having the set of queries, Bing's large-scale web index is used to retrieve relevant passages, totaling 8.8 million passages. In the final step, human editors annotate passages that contain useful information and compose well-formed natural language answers. This dataset is particularly useful to benchmark many machine reading comprehension problems and question-answering models.

The TREC Conversational Assistance Track 2019 (CAsT) [10] is also a retrieval-based dataset, but contrary to the ones presented before, it is focused on conversational passage retrieval. This dataset is the most aligned with our task, and we perform an extensive analysis in section 6.1. In summary, this dataset contains 80 conversational topics, where each one has, on average, 10 conversational questions (turns) about a specific topic. The

passage collection is composed of the datasets WaPo, TREC CAR, and MS MARCO totaling over 47 million documents. This retrieval task is more challenging than simple retrieval because of the need to track the context of the conversation to retrieve relevant information.

2.2.1.2 Question-Answering Datasets

When referencing question-answering (QA) datasets, we consider datasets where the aim is to extract from a single passage or document (no retrieval required) a span that answers a particular question.

The Stanford Question Answering Dataset, or SQuAD [49] for short, is a dataset that consists of questions constructed by crowdsourced workers on a set of Wikipedia articles. The answers to every question are a passage or span of text, which makes them easier to evaluate than free-form answers. Version 1.1 of the dataset contains more than 100.000 question-answer pairs on 536 articles. To collect the data, researchers used three steps. The passage curation step was used to retrieve high-quality articles. The question-answer collection step used crowdsourced workers from Amazon Mechanical Turk. In this step, each worker was tasked to ask 5 questions on the content of the paragraph and highlight the answers in the given paragraph. The final step is creating an additional answer collection to evaluate human performance. In this step, the workers were shown the questions and the paragraphs of an article and were told to select the shortest span in the paragraph that answered the question.

The newest version of SQuaD is the 2.0 [48]. In this version, the original dataset was augmented with 50.000 unanswerable questions, also written by crowdsourced workers, to look similar to answerable questions.

Since our focus is on a conversational task, where the user can make many questions about a topic in a context-dependent way, we also describe the conversational QA datasets QuAC [3] and CoQA [50].

The QuAC dataset [3] (Question Answering in Context) tries to mimic a real conversation, containing 14k crowdsourced dialogues with a total of 100k QA pairs. The dataset was collected in an interactive way, where two crowdsource workers play the roles of "student" and "teacher". The "student" poses a sequence of free-form questions about a Wikipedia article, where the "student" only sees the section's title and the first paragraph of the main article, not knowing the answers to the question prior asking them. The task of the "teacher" is to provide a span that answers the question, in case it exists. The "teacher" has access to the entire Wikipedia article, and after each question must provide the "student" with a list of dialog acts. There exist three dialog acts: (1) continuation (follow up, maybe follow up, or do not follow up), (2) affirmation (yes, no, or neither), and (3) answerability (answerable or no answer). These dialog acts are used so that "teachers" can guide the "students" to the more important aspects of the article. The dialog ends if 12 questions are answered, one of the partners ends the interaction, or more than two

unanswerable questions were asked.

CoQA [50] is another conversational question answering dataset. This dataset contains 127k question-answer pairs (average 15 turns), obtained from 8k conversations about text passages from diverse domains. These domains include children's stories, literature, middle and high school English exams, news, Wikipedia, Reddit, and science. The data collection process is similar to QuAC using crowdsourced workers, where one annotator takes the role of the questioner, and the other takes the role of the answerer. The questioner's role is to ask natural language questions. The answerer's role is to answer the questions and highlight the rationales (span of text supporting the answer). Contrary to QuAC, in CoQA, the questioner has access to the text where the answers are coming from.

In table 2.1 is presented a summary of the characteristics of the datasets analyzed. In particular, we emphasize the TREC CAsT dataset [10] for being the only one combining both retrieval and conversational data in a single dataset.

Table 2.1: Comparison of the various datasets.

Dataset	Retrieval	Conversational	Dialog Acts	Unanswerable
WaPo [22]	✓	✗	✗	✓
TREC CAR [13]	✓	✗	✗	✓
MS MARCO [36]	✓	✗	✗	✓
TREC CAsT [10]	✓	✓	✗	✗
SQuAD 1.1 [49]	✗	✗	✗	✗
SQuAD 2.0 [48]	✗	✗	✗	✓
QuAC [3]	✗	✓	✓	✓
CoQA [50]	✗	✓	✗	✓

2.2.2 Evaluation Metrics

Having metrics is important to measure the performance of the system and to compare it with others. In this section, we describe the most common metrics used to evaluate information retrieval and natural language generation systems.

Recall represents the fraction of documents that are relevant to the query that were retrieved successfully. So, a recall of 1 means that all documents that are considered relevant were retrieved. If the recall is 0, none of the documents retrieved is relevant to the query. The equation to calculate recall is equation 2.1.

$$recall = \frac{|\{relevant\ docs\} \cap \{retrieved\ docs\}|}{|\{relevant\ docs\}|}. \tag{2.1}$$

Precision can be viewed as the fraction of documents retrieved that are relevant to the user's information need. Also used is $P@K$, which represents the precision on the top K

results. A P@K of 1 tells us that all the documents in the top K are considered relevant, a precision of 0 tells us the opposite. The equation to calculate precision is equation 2.2.

$$precision = \frac{|\{relevant\ docs\} \cap \{retrieved\ docs\}|}{|\{retrieved\ docs\}|}. \tag{2.2}$$

The **F1 Score** represents the harmonic mean of precision and recall, so an F1 score of 1 indicates that the system achieved maximum precision and recall. The formula to calculate the F1 score is equation 2.3.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}. \tag{2.3}$$

Discounted Cumulative Gain (DCG) is a metric used to penalize when a highly relevant document is placed lower on a list of ranked results. The penalty is a graded relevance value that is reduced logarithmically, proportional to the position of the result (i), as presented in equation 2.4. Also used is the normalized DCG (nDCG), since search result lists may have different lengths depending on the query. This can be achieved by sorting all relevant documents in the corpus by their relative relevance, and thus produce the maximum possible DCG through position p called Ideal DCG (IDCG). The nDCG can be computed using equation 2.5.

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \tag{2.4}$$

$$nDCG_p = \frac{DCG_p}{IDCG_p}. \tag{2.5}$$

Mean Average Precision (MAP) for a set of queries is the mean of the average precision scores for each query. It is represented by equation 2.6, where Q represents the number of queries.

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}. \tag{2.6}$$

Mean Reciprocal Rank (MRR) is used to calculate the reciprocal of the rank at which the first relevant document was retrieved. The reciprocal rank is 1 if the first relevant document was retrieved in rank 1, 0.5 if retrieved in rank 2, and so on, following equation 2.7.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}. \tag{2.7}$$

BLEU (Bilingual Evaluation Understudy) [43] is an automatic machine translation evaluation metric that is used to evaluate text generation. BLEU calculates the correspondence between the system's output and a reference output. BLEU values range from 0 to 1, being 1 the maximum similarity between the system output and the reference output.

ROUGE (Recall Oriented Understudy for Gisting Evaluation) [29] appears as a modification of BLEU that works in the opposite way, focusing on recall. It compares how many n-grams in the reference output appear in the system's output.

Word Perplexity is often used for probabilistic language modeling. It represents the inverse probability of the test set, normalized by the number of words, meaning that a lower perplexity results in a better model [24]. Equation 2.8 represents this metric, where w represents a word, and N is the number of words in the dataset.

$$PP(W) = P(w_1 w_2 \dots w_N)^{(-\frac{1}{N})} = \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}. \tag{2.8}$$

Word Error Rate (WER) represents the number of words in the output that the model has predicted incorrectly, divided by the total number of words in the reference output. The equation for WER is 2.9, where S, D, and I are the number of substitutions, deletions, and insertions that the model predicted, and N is the number of words in the reference output.

$$WER = \frac{S + D + I}{N}. \tag{2.9}$$

2.3 Pre-trained Transformer Models

Embedding is the technique where words or sentences from a vocabulary are mapped to a vector of real numbers. These vectors provide a way of performing calculations using words, being able to, for example, calculate the similarity between two words or perform peculiar calculations such as, $vec(\text{“}Madrid\text{”}) - vec(\text{“}Spain\text{”}) + vec(\text{“}France\text{”}) \approx vec(\text{“}Paris\text{”})$, where *vec()* is a function that transforms the word into its vector representation [34].

A **Recurrent Neural Network** (RNN) is a generalization of a feed-forward neural network that has a notion of "memory". It is called recurrent because it applies the same function to all inputs, but the output of the current input is dependent on the previous computation. Considering these characteristics, RNNs can be used to generate sentence embeddings where a text is viewed as a sequence of words. To generate these embeddings, the text of the sentence is mapped to a dense, low-dimensional semantic vector by sequentially processing each word and mapping the subsequence up to that word into another vector that is viewed as the hidden state [17]. An RNN and its extensions based on gated RNNs, such as LSTMs (Long-Short Term Memory) [20] and GRUs (Gated Recurrent Unit) [2], are often used to allow persistence. They use the hidden state to remember some information about the full sequence. With this hidden state, an RNN can learn from the past, but in most cases, this only works correctly if the gap between the relevant information and the place where it is needed is small.

Although useful, RNNs do not capture sufficient context to handle complex Natural Language Processing (NLP) tasks [59]. So, a newer generation of models pre-trained in

a large corpus, such as BERT [12] and others [31, 47, 65] are now used. These types of models are capable of not only generating sentence embeddings but also capture the context of the words in the sentence, being able to, for example, disambiguate the same word in different contexts, e.g., a "bat" can be a piece of sporting equipment used in baseball or a winged mammal.

The bulk of these new pre-trained models are based on the **Transformer** architecture, first presented in [56]. The Transformer can handle sequential data, such as text, but contrary to what happens in the RNNs, the Transformer does not require a sequence to be processed in order, allowing for parallelization and embeddings conditioned by all the words in the sentence. Adding to this, the Transformer also addresses the mentioned limitations of the RNNs by using an attention mechanism. This mechanism can access all the tokens in the input and attribute a weight (a measure of relevance) to each one, providing a better representation for each token because this representation is based on all the tokens in the input and not on a hidden state like an RNN. To be more specific, the Transformer uses an encoder-decoder architecture, however, our main interest is in the encoder component as represented in Figure 2.1.

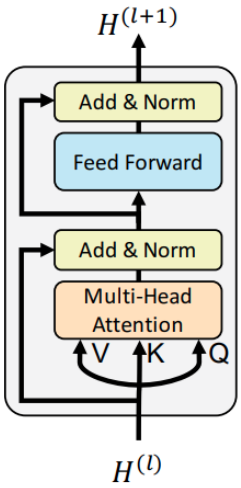


Figure 2.1: The Transformer encoder architecture [56].

The encoder consists of three main components: the attention mechanism, layer normalization, and a feed-forward neural network, where the most important and innovative part is the attention mechanism. In this component, the input is a sentence from which attention weights are calculated for every token. As seen in Figure 2.1, the attention module receives three inputs, Value (V), Key (K), and Query (Q), and learns three weight matrices W_V, W_K, and W_Q. Each token in the input sentence is then multiplied by these matrices, resulting in three corresponding vectors per token. To calculate the attention weights, the dot product of the query and key vectors for every token is used, divided by the square root of the size of the key vector (d_k) to stabilize the gradients during training. This representation is then passed to a *softmax* function to obtain the final weights.

In summary, this procedure is represented in equation 2.10:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \tag{2.10}$$

So, the multi-head attention component represented in Figure 2.1 is composed of multiple different sets of the previously mentioned matrices that the researchers showed focus on different aspects of the input [56]. Here we presented just a short description of the encoder architecture, for more detail, refer to the original paper [56].

2.3.1 Autoencoder Models

Autoencoder language models are characterized by their ability to reconstruct the original data from corrupted input. An example of these types of models is the Bidirectional Encoder Representations from Transformers or BERT [12] for short. This model was first presented by Devlin et al. [12] as a new language representation model. This model was designed to pre-train deep bidirectional representations of unlabelled text by jointly conditioning on the left and right context in all layers, meaning that a word is conditioned on the words to its left and right. Because of this characteristic, this model can work on various tasks, such as question answering and language inference, without significant changes in the architecture.

BERT implements a multi-layer bidirectional architecture of transformer encoders, following the principles and ideas first presented in [56], especially in the attention components. In this architecture, the number of layers is denoted as L, the hidden size as H, and the number of self-attention layers as A. The most used models are the BERT BASE (L=12, H=768, A=12, Total Parameters=110 Million), and BERT LARGE (L=24, H=1024, A=16, Total Parameters=340 Million).

The input representation can be a single sentence or a pair of sentences like a question-answer pair. The first token of every sequence is a special token called [CLS] (classification) that aims to be a representation of the entire sentence. To use sentence pairs, these are packed together into a sequence with another special token, [SEP] (separator), in between, along with the addition of a learned embedding to every token, indicating if it belongs to sentence A or B (segment embeddings). On top of those embeddings, a position embedding is used that indicates the absolute position of the token in the sequence. So, denoting the input embeddings as E, the final input representation for a token is the sum of the token, segment, and position embeddings, as represented in Figure 2.2.

Regarding the training of the model, there are two phases: the pre-training phase and the fine-tuning phase that can be seen in Figure 2.3. In the pre-training phase, the model is trained on unlabelled data over two unsupervised tasks:

- **Masked Language Model (MLD)** – to train the deep bidirectional representation, 15% of the input tokens are masked at random and then predicted [12] (denoising objective). To mask the tokens a special symbol [MASK] is used 80% of the time,

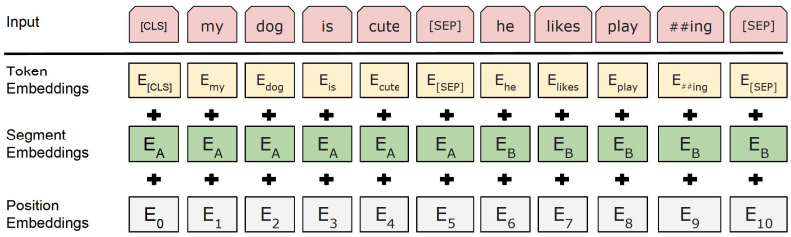


Figure 2.2: BERT input representation for a pair of sequences [12].

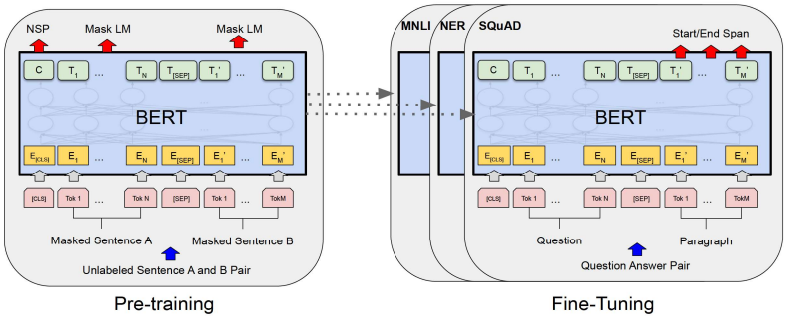


Figure 2.3: BERT's architecture for pre-training and fine-tuning [12].

but this creates a mismatch between training and fine-tuning, so the researchers also replace the tokens with a random token (10%) or by the actual input token (10%).

- **Next Sentence Prediction (NSP)** - researchers pre-trained a binarized next sentence prediction task [12]. Meaning that when choosing the sentences A and B for each pre-training example, 50% of the time B is the sentence that follows A, and 50% of the time B is a random sentence from the corpus. The aim of the model is to classify if the second input sentence is, in fact, the sentence that follows the first one, using the embedding generated by the model for the [CLS] token.

In the fine-tuning phase, the model is initialized with the pre-trained parameters, and then all of them are fine-tuned using labeled data from downstream tasks. The self-attention mechanism in the Transformer grants BERT the ability to model these downstream tasks. This allows performing simple modifications to model different tasks, by putting the inputs and outputs into BERT and fine-tuning all the parameters end-to-end. When compared to pre-training, the fine-tuning task is relatively inexpensive.

In Liu et al. [31] is presented a revised version of the original BERT model called Robustly optimized BERT approach or RoBERTa for short. The main difference in this model resides in the training phase. In RoBERTa, 160 GB of data are used for training, while the original BERT used only 16 GB. During training, the masked language model used in RoBERTa differentiates itself from BERT by using a dynamic mask that is generated every time a sequence is fed to the model, while in BERT the mask is static, using the same mask for each training instance in every epoch. In RoBERTa the next sentence prediction

task was removed, so each input is packed with full sentences sampled contiguously from one or more documents (total length <= 512 tokens), allowing inputs to cross document boundaries if needed. Another modification to the original training algorithm was the use of larger mini-bathes and learning rates, achieving an improvement in perplexity for the masked language modeling objective and end-task accuracy, making it also easier to parallelize via distributed data-parallel training [31]. The vocabulary used in this work contains 50k sub-word units, without any additional preprocessing or tokenization of the input. In comparison, BERT uses a vocabulary of 30k sub-words. All of these changes applied during training allowed the RoBERTa model to achieve state-of-the-art results in various tasks, surpassing the equivalent BERT model in most tasks [31].

In a more recent paper [47] is presented a study on the transfer learning techniques (large pre-train models), a new model called Text-to-Text Transfer Transformer (T5 for short), and a new dataset named Colossal Clean Crawled Corpus (C4) that was used to pre-train this model. Our interest, in particular, is on the T5 model. This model is based on the encoder-decoder architecture [56] and achieved state-of-the-art results in various tasks. The model has a simple input and output structure that uses strings, being considered a text-to-text model. The pre-training task is similar to BERT using a denoising objective, but the amount of data used to train the model is much larger thanks to the C4 dataset (803 GB). By using this architecture, the model can be used in a variety of tasks such as machine translation, question answering, and classification. The model also exists in various sizes (number of parameters), with the largest one having 11 billion parameters, requiring modern accelerators, such as TPUs to train. To be comparable to BERT, T5 also has two versions called BASE and LARGE with a similar size to BERT.

2.3.2 Autoregressive Models

A problem with BERT is that there exists a difference between the training step and the fine-tuning step. This happens because the training step includes the masking of parts of the input using the [MASK] token that does not appear in the fine-tuning step. Another limitation of BERT is that it assumes independence between the masked tokens, although this is not always the case.

Autoregressive language modeling aims to estimate the probability distribution of a text using an autoregressive model. Typically these types of models are only trained to encode uni-directional context (either forward or backward), while models like BERT are built to model bi-directionality. However, in [65], a new model called XLNet is created that resolves the aforementioned limitations of the BERT model, using a generalized autoregressive method that can encode bi-directional context. In particular, the bidirectionality is achieved by maximizing the log-likelihood of a sequence, by taking into account all possible permutations of the factorization order. This technique is called Permutation Language Modeling and is the innovation that allows the model to capture the bidirectional context. Yang et al. [65] compared XLNet to an equivalent version of BERT

and showed that XLNet improves the results in most tasks, achieving state-of-the-art in some of them.

2.4 Indexing and First-Stage Retrieval

2.4.1 Indexing

Indexing is the process of creating the pool of data, from which the first-stage retrieval model will search for the relevant information given a query.

2.4.1.1 Preprocessing in Natural Language

Preprocessing data is a simple modification that can often have a significant impact on the results obtained. Some of the most notable methods are the following:

Capitalization is the technique that involves changing the capitalization of words. Typically an approach of reducing all the words to lower case is used. This approach increases the number of matches, but it also neglects some words that may be confused if we use the same case, like in the case of "US" (United States), that is transformed into "us".

Tokenization is the process of splitting text into meaningful parts, such as splitting text into sentences, or sentences into words. After the tokenization, an algorithm can be applied to each unit. The most typical tokenization is the punctuation and white-space tokenizer, but more complex strategies can also be applied.

Stop words are the most common words in a language or words that do not provide relevant information and so can be filtered before processing the text. There are various lists of stop words, so different lists can be used that may affect the quality of the results. Dai and Callan [7] showed that stop word removal improves results of traditional methods based on term frequency (e.g., BM25 [51]), but that models like BERT [12] achieve better results without stop word removal.

Stemming and Lemmatization. Stemming is the process of reducing inflected or derived words to their root form or word stem. For example, the words "likes", "liked", and "liking" are stemmed to the word "like". Lemmatization has a similar function as stemming but can capture more complex examples by using a dictionary to lookup word stems. For example, "good" is the lemma of the word "better". These techniques can also increase the number of matches but at the cost of the precision of the terms.

2.4.1.2 Index Expansion

One of the problems with traditional indices is that the term frequency is a typical method to determine the importance of a term in a document (tf) or query (qtf). This simplistic approach might not be enough if the distribution of the terms is flat or the sentence is short, making it difficult to assess whether a term is relevant to the meaning of the

text. Index expansion tries to solve this by expanding the documents in the index with words or sentences that can make the document more easily discoverable by retrieval algorithms. This method is used to improve retrieval efficiency without significantly increasing retrieval time.

Dai and Callan [6] propose a Deep Contextualized Term Weighting (DeepCT) framework that learns to map BERT's contextualized text representations to context-aware term weights for passages that can be used by typical first-stage retrieval algorithms. So, DeepCT can be used to create a query-independent index that stores the weights of the terms in passage-long documents. The weights of the terms are obtained by training the BERT model to predict if a term in the passage is likely to appear in a relevant query. The main difference between DeepCT's index and a typical inverted index is that the term weights are based on the term frequency generated by DeepCT. This process is performed offline, so it does not add any latency to the system. The experimental results showed that DeepCT improves the accuracy of first-stage retrieval algorithms and the accuracy/efficiency tradeoff for later-stage re-rankers, achieving a better accuracy with fewer candidate documents. In summary, the main advantage of DeepCT over classic term weighting approaches is that it finds the most central words, regardless of the term frequency. A problem with this approach is that non-central words that may be frequent are suppressed.

Another possible approach to increase retrieval efficiency, instead of repeating the central terms like in DeepCT, is to expand the documents with terms that represent their content. This approach can make the most relevant terms more salient and also diminish the vocabulary mismatch problem that occurs when users use terms in the query that are different from the ones in the document. In [41] is proposed doc2query, a method of predicting queries for a document, that is then expanded with those predictions. This method uses a sequence-to-sequence transformer model, that given a document, outputs possible questions that the document can answer. The model is trained using a dataset consisting of pairs of queries and documents, where the input is a document, and the output is a predicted query. After training, the model is used to predict queries that are appended to each document. These expanded documents are then indexed and retrieved by a standard retrieval algorithm. To evaluate the performance of this method, Nogueira et al. [41] used the MS MARCO [36] and the TREC CAR [13] datasets, achieving an increase in retrieval effectiveness of approximately 15% in both datasets, when compared to the BM25 baseline, with the advantage of not significantly increasing the retrieval latency since this method is done prior indexing.

In Nogueira and Lin [39] is presented docTTTTTquery, a model that expands on the ideas of doc2query using the same setup but replacing the sequence-to-sequence transformer model with the sequence-to-sequence encoder-decoder model T5 [47]. This model outperforms the BM25 baseline and doc2query on the MS MARCO dataset, attributing the improvements over doc2query to the better language representation and pre-training that the T5 model has.

2.4.2 First-Stage Retrieval

First-stage retrieval is the initial step of fetching information from the dataset. This operation needs to be efficient and fast, but also retrieve relevant documents, achieving a high recall, in order to not compromise the next steps in the pipeline.

Although, in general, not returning the best ranking possible, this initial step is able to return a specified number of documents, in the order of a few thousand, from a large set of documents, in the order of millions, in a short amount of time. These documents can later be re-ranked by a more computationally complex algorithm to achieve a better ranking.

2.4.2.1 BM25

Okapi BM25 [51] is a ranking function based on a probabilistic model that estimates the relevance of a document given a query. The typical ranking function of BM25 is a bag-of-words function that ranks a set of documents based on the query terms appearing in each document. The usual scoring function used in BM25 is represented in equation 2.11.

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})}, \tag{2.11}$$

where D is a document, Q is a query containing keywords $q_1,\ldots q_n$, $f(q_i,D)$ is the term frequency of q_i in document D, $|D|$ is the length of the document in words, $avgdl$ is the average document length in the text collection from which documents are drawn, k_1 and b are parameters to allow optimization, and $IDF(q_i)$ is the inverse document frequency of the query term q_i, usually calculated using equation 2.12, where N is the number of documents and $n(q_i)$ the number of documents containing q_i.

$$IDF(q_i) = ln(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1). \tag{2.12}$$

Another version of the BM25 model is BM25F [51]. This model considers a document as a set of fields, for example, title, abstract, and body, and gives different degrees of importance to each one.

2.4.2.2 Query Likelihood Model

In a language modeling approach, the objective is to estimate a language model for each document and then rank these documents by the likelihood of the query, according to the language model. The smoothing part of the model refers to the adjustment of the maximum likelihood estimator of a language model. Zhai and Lafferty [67] showed that the smoothing of the maximum likelihood is directly related to retrieval performance.

Two query likelihood models often used are the Jelinek-Mercer method (LMJM) and the Bayesian smoothing using Dirichlet priors (LMD). In the case of LMJM, the method

involves the interpolation of the maximum likelihood model, with the collection model, using a coefficient λ to control the influence of each model as represented in equation 2.13.

$$p_\lambda(w|d) = (1-\lambda)p_{ml}(w|d) + \lambda p(w|C), \tag{2.13}$$

where $p_\lambda(w|d)$ represents the unigram language model given document d, $p(w|C)$ is the collection language model, and $p_{ml}(w|d)$ is the maximum likelihood estimate given by relative counts.

The LMD model assumes the language model as a multinomial distribution where the conjugate prior for Bayesian analysis is the Dirichlet distribution. With this, it is possible to derive equation 2.14, where $p_\mu(w_i|d)$ represents the unigram language model given document d, $p(w_i|C)$ is the collection language model, $c(w;d)$ is the count of word w in the document d, and μ is a parameter greater than zero.

$$p_\mu(w_i|d) = \frac{c(w_i;d) + \mu p(w_i|C)}{\sum_{w_j} c(w_j;d) + \mu}. \tag{2.14}$$

Zhai and Lafferty [67] also found that the performance is generally more sensitive to smoothing in longer queries, and that LMD performs best with shorter queries while LMJM performs best with longer queries.

2.5 Re-ranking Models

In this section, we introduce several re-ranking models that are used after the first-stage retrieval step and are more computationally expensive. The aim of these models is, by having a "relatively small" (a few thousand) set of documents or passages (when compared to the complete index), create a new rank that will better reflect the answers to the query. This new rank is expected to improve on the results of the first-stage retrieval rank.

2.5.1 Coordinate-Ascent

Coordinate ascent [33] is an algorithm that is frequently used in optimization problems to iteratively optimize a multivariate objective function. To do this, it repeatedly cycles through each parameter, maintaining the others fixed, optimizing the parameter in use. In an information retrieval setting, coordinate ascent is a *listwise* approach, where the scores of the pairs query-document are calculated as weighted combinations of the feature values. This method uses two hyperparameters: (1) the number of restarts from random weights and (2) the number of iterations.

2.5.2 Deep Relevance Matching Model

Deep Relevance Matching Model (DRMM) [19] is a model designed for relevance matching in an ad-hoc retrieval setting. This model addresses exact match signals (same terms

in query and document), query term importance (each term in the query has an importance weight), and other matching requirements (e.g., able to compare documents of different lengths). This model uses a deep architecture at query term level over the local interactions between query and document terms, a matching histogram mapping, a feed-forward matching network, and a term gating network. Figure 2.4 shows the model's architecture.

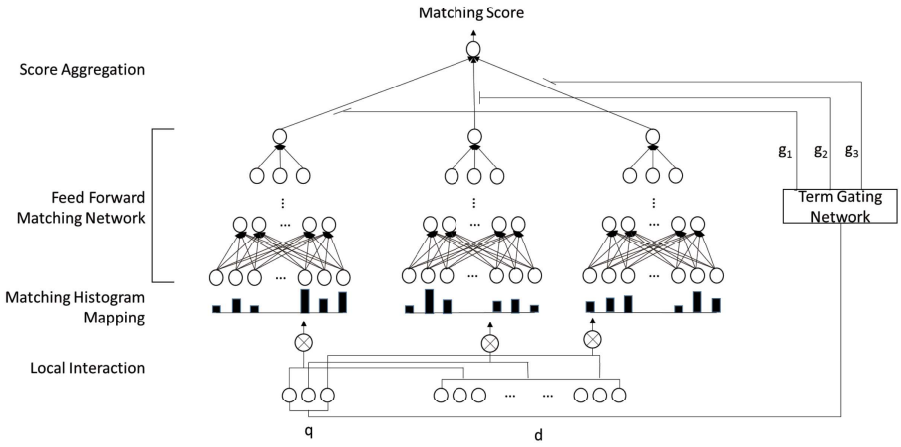


Figure 2.4: Architecture of the Deep Relevance Matching Model [19].

In this model, the first step is to build the local interactions between each pair of terms from a query and a document based on pre-computed term embeddings. After that step, each query term variable-length interactions are transformed into a fixed-length histogram. This histogram groups local interactions according to different levels of signal strength. Since the values are obtained using the cosine similarity, this leads to values between [-1,1] that are distributed in a set of disjoint bins. In this work, the bins are of fixed size, and the exact match ([1,1]) is treated as a separate bin. Then a feed-forward matching network is used to learn hierarchical matching patterns and produce a matching score for each query term, basing this on the previously constructed histogram. The final score is generated by aggregating the scores from each query term with a term gating network to compute the aggregation weights. This allows the model to explicitly model query term importance by controlling how much the relevance score of that query term contributes to the final relevance score. The gating function used was the *softmax* function in equation 2.15:

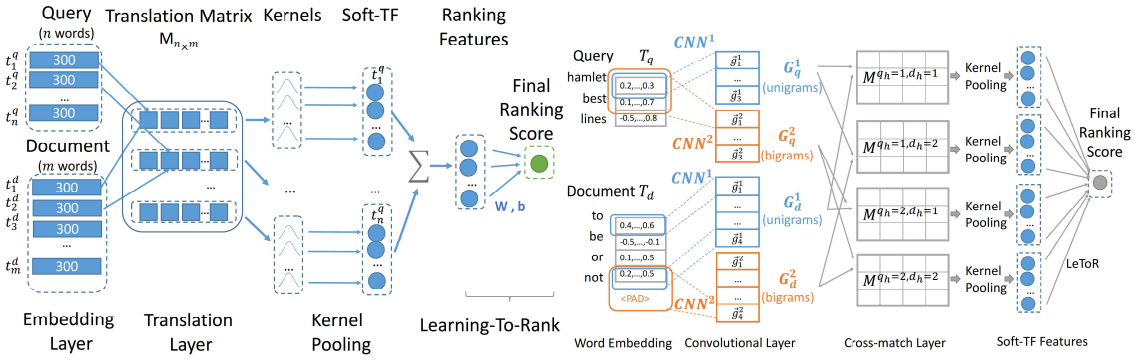
$$g_i = \frac{exp(w_g x_i^{(q)})}{\sum_{j=1}^{M} exp(w_g x_j^{(q)})}, \quad i = 1,\ldots,M \tag{2.15}$$

where w_g denotes the weight vector of the term gating network and $x_i^{(q)}$, $i = 1, \ldots, M$ denotes the i-th query term input.

The experimental results showed that DRMM outperforms traditional retrieval models like BM25 [51] and LMD [67], being later surpassed by KNRM [61] that we will discuss in the following section.

2.5.3 Kernel-based Matching Models

In neural approaches, such as DRMM [19], presented in the previous subsection, it is typical to use distributed representations like word2vec [34]. These representations can cause problems that can be seen in this example withdrawn from Xiong et al. [61]: *" Word2vec may consider 'Pittsburgh' to be similar to 'Boston', and 'hotel' to be similar to 'motel'. However, a person searching for 'Pittsburgh hotel' may accept a document about 'Pittsburgh motel', but probably will reject a document about 'Boston hotel' "*. The work presented in Xiong et al. [61] addresses these problems using a kernel-based neural ranking model (K-NRM). The architecture of the model can be seen in Figure 2.5(a).



(a) K-NRM architecture [61] (b) Conv-KNRM architecture [8]

Figure 2.5: K-NRM (left) and Conv-KNRM (right) architectures.

The K-NRM model uses distributed representations to represent query and document words and then uses their similarities to build a translation model. This translation model is built by using an embedding layer to map each word t to an L-dimension embedding $\vec{v}_t$. Then a translation layer constructs the translation matrix M, in which each element is the embedding similarity between a query word and a document word: $M_{ij} = cos(\vec{v}_{t_i^q}, \vec{v}_{t_j^d})$.

Word pair interactions are then combined by a kernel-pooling layer that uses soft term frequencies to convert the translation matrix M to query-document ranking features ϕ(M):

$$\begin{aligned} \phi(M) &= \sum_{i=1}^{n} log\vec{K}(M_i), \\ \vec{K}(M_i) &= \{K_1(M_i), \ldots, K_K(M_i)\}, \end{aligned} \tag{2.16}$$

where $\vec{K}(\mathrm{M_i})$ applies K kernels to the i-th query word's row of the translation matrix, pooling it into a K-dimensional feature vector. The ranking features ϕ(M) are then combined by a ranking layer to produce the final ranking score in equation 2.17, where w and b are the ranking parameters to learn and $tanh()$ is the activation function:

$$f(q,d) = tanh(w^T \phi(M) + b). \tag{2.17}$$

Neural models like this require large amounts of training data. However, acquiring manual training labels is too expensive, so to circumvent this problem, Xiong et al. [61]

used user click data, which has shown that it can accurately predict manual labels. In terms of results, K-NRM showed significant improvements, being extremely effective at the top-ranking positions. According to the researchers, the kernel is the key to K-NRM results, converting the learning-to-rank loss to requirements on soft term frequency patterns, adjusting word embeddings in order to better separate relevant from irrelevant documents [61].

An extension of the work in K-NRM [61] is Conv-KNRM [8]. Conv-KNRM presents a convolutional kernel-based neural ranking model that uses a learned convolutional layer to form n-grams from individual word embeddings. The convolutional layer is used to project all n-grams to a common embedding space, allowing to match n-grams of different lengths (cross-matching), being this the main advantage of this method.

In Conv-KNRM, the words are embedded in continuous vectors and then a CNN is employed (convolutional neural network) to create n-gram embeddings from adjacent word embeddings. This CNN applies convolutional filters by going over the text like a sliding window. As in the case of K-NRM, soft-matches play an important role. These soft-matches are calculated using the similarity between n-gram embeddings. After this step, a kernel-pooling layer and a learning-to-rank layer are used to combine the n-gram soft-matches resulting in a final ranking score. In Dai et al. [8], it is possible to see all the details of the approach and Figure 2.5(b) represents the model's architecture.

The results of Conv-KNRM showed the advantages of soft-matching n-grams in relevance ranking, outperforming K-NRM in most tasks [8].

2.5.4 BERT-based Ranking Models

Nogueira and Cho [37] present a re-implementation of BERT for query-based passage re-ranking, estimating a score s_i, which represents the relevance of a passage d_i to the query q. The authors feed BERT with the query as sentence A and the passage as sentence B, truncating the question to 64 tokens. The passage of text is also truncated, such that the concatenation of the query, passage, and separator tokens have a maximum of 512 tokens. The researchers used a BERT LARGE model as a binary classifier, using the [CLS] vector, which is present in all sequences, as input to a single layer neural network, to obtain the probability of the passage being relevant. This probability is computed for each passage, and the final result is obtained by re-ranking the passages according to these probabilities. The fine-tuning of the BERT model is done using cross-entropy loss, as represented in equation 2.18.

$$L = -\sum_{j \epsilon J_{pos}} log(s_j) - \sum_{j \epsilon J_{neg}} log(1 - s_j), \tag{2.18}$$

where s_j is the likelihood of the passage being relevant, J_{pos} is the set of indexes of the relevant passages, and J_{neg} is the set of indexes of non-relevant passages in the top-1.000 documents retrieved with BM25 [51].

This model was evaluated in two passage ranking datasets: MS MARCO [36] and TREC-CAR [13], and at the time achieved state-of-the-art results on both datasets by a large margin, surpassing the models previously described in this section.

In [40] is proposed a multi-stage document ranking algorithm based on [37]. The proposed method is comprised of 3 stages that are used in a pipeline fashion, as represented in Figure 2.6.

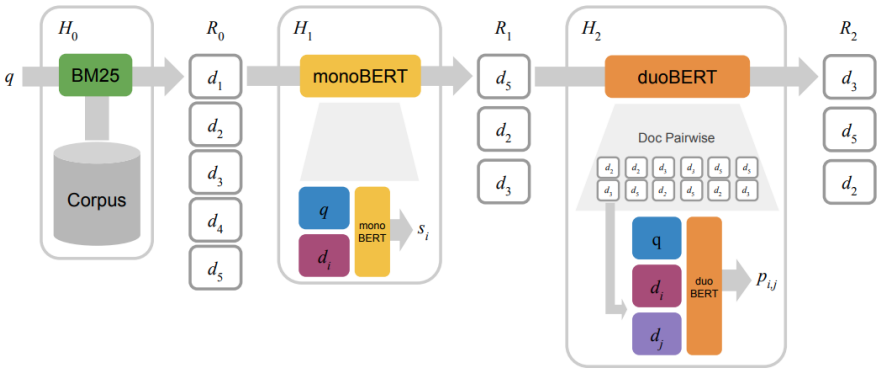


Figure 2.6: Architecture of the Multi-Stage Document Ranking BERT [40].

The first step is the initial retrieval. This step uses an inverted index and a BM25 [51] scoring function, being recall the main focus of this step (typically retrieved 1000 documents). The second step is called monoBERT, and it uses BERT as a *pointwise* re-ranker (considers a single document at a time), using a binary relevance classifier. In monoBert is employed the same procedure as in [37]. The input is a query and a document, and the model outputs the probability of the document being relevant to the query. Then these documents are re-ranked using this probability to prune the results (resulting in 20-50 documents). After this, these documents are passed to another BERT model called duoBERT, which is characterized as a *pairwise* approach (considers a pair of documents at a time). The duoBERT takes as input a query, a document d_i, and a document d_j, and then the [CLS] vector is used as input to a single layer neural network to obtain the probability $p_{i,j}$, of d_i being more relevant than d_j. At inference time, the pairwise scores are aggregated so that each document receives a single score. The final step is to re-rank the list of documents according to these scores.

2.6 Conversation State Tracking

In this section, we introduce several state tracking models, which are used to maintain the state of the conversation. The state tracking task is one of the most important aspects of conversational systems because it gives context to the current conversation turn. To achieve this, it is necessary to have a notion of the information seen so far and the ability to understand when this information is needed, using it when a relevant situation arises.

2.6.1 Hierarchical Recurrent Encoder-Decoder

The Hierarchical Recurrent Encoder-Decoder (HRED) was initially proposed by Sordoni et al. [53] for query suggestions. In that approach, the history of past queries was considered as a sequence at two levels: a sequence of words for each query and a sequence of queries. To achieve that behavior, two RNNs were used to encode, one at word level and another at query level. And a third RNN was used to decode an answer based on the output of the encoder at sequence level.

In Serban et al. [52] is tackled the task of building a conversational dialogue system based on large dialogue corpora using generative models (models that produce responses auto-generated word-by-word). To achieve this, Serban et al. [52] proposed an extension to the original HRED model in [53] by analogously considering a sequence of words for each query, and a sequence of queries in the original model, as a sequence of tokens, and a sequence of utterances, in a dialogue system. In a dialogue, the encoder RNN maps each utterance to an utterance vector representing the hidden state. The RNN at a higher-level keeps track of the past utterances (context) by processing each utterance vector iteratively. The next utterance prediction is performed by a decoder RNN, which takes the hidden state of the context RNN and produces a probability distribution over the tokens in the next utterance. The encoder, context, and decoder RNNs all make use of the GRU architecture [2]. Figure 2.7 represents the specified architecture.

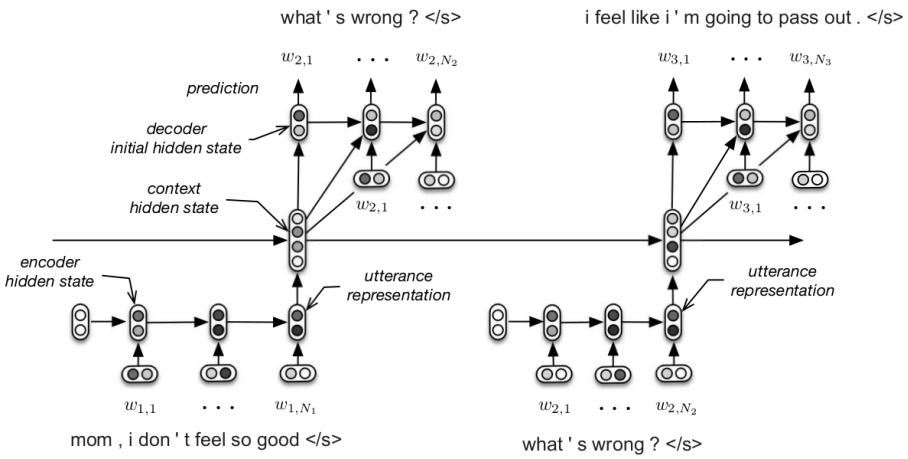


Figure 2.7: Computational graph of the HRED architecture for a dialogue composed of three turns [52].

In the same work, Serban et al. [52] also experimented with a bidirectional HRED. This model is particularly useful for utterances that are long and have a more complex syntactical articulation. In addition to the chain going forward, the bidirectional model has a chain going backwards through the utterance tokens (reversing the tokens in the utterance). From the results obtained, the bidirectional HRED appeared to retain information from previous utterances better.

In a conversation question answering scenario, we can leverage HRED's sentence and

conversation level architectures to model the conversational history.

2.6.2 Memory Networks

Because of the small memory and gradient vanishing problems of RNNs, Weston et al. [59] present a class of models called memory networks. Memory networks address the RNNs limitations by providing an easy way to read and write part of a long-term memory component. Typical knowledge base approaches for QA use a two-phase strategy of first applying information retrieval or extraction to find the answers and then inference. In opposition, in memory networks, the extraction of information to answer a question is performed "on-the-fly" over the memory.

Considering the architecture, a memory network consists of a memory, m, and four components:

- **I** (Input feature map) – parses, does coreference resolution, and encodes the input into an internal feature representation.
- **G** (Generalization) – this component makes use of a function that can be as simple as storing the memory as is, or more complex, being able to update old memories, organize the memory (in the context of a huge memory), and forget less useful memories.
- **O** (Output feature map) – produces a new output (in the feature representation space), given the new input and the current memory state. It can also be used to perform inference.
- **R** (Response) – converts the output into the desired response format.

In the basic model, component I takes an input text in its original form, and component G stores it in the next available memory slot. The O module produces output features by finding k supporting memories given x (a pair of sentences). The function that scores the pair, s_O, depends on the k value and can be generalized for a k=1 as in equation 2.19, where $i = 1,\ldots,N$ are the memories available in memory m.

$$o_{k=1} = O_1(x,m) = \underset{i=1,\ldots,N}{\arg\max}\, s_O(x,m_i), \tag{2.19}$$

With k=2, the second iteration takes into account the value of the previous, resulting in equation 2.20.

$$o_{k=2} = O_2(x,m) = \underset{i=1,\ldots,N}{\arg\max}\, s_O([x,m_{o_1}],m_i). \tag{2.20}$$

For the response production in component R is used equation 2.21, where W is the set of all words in the dictionary and S_R is the function that scores the match. This equation is the simplest of all in terms of providing an answer, being responsible for outputting a

single word, out of all the words seen by the model. This can be used to answer simple questions, where the answer is only one word.

$$r = arg\max_{w \epsilon W} s_R([x, m_{o_1}, m_{o_2}], w), \tag{2.21}$$

The training of the memory network is done in a supervised manner, where the inputs and responses are labeled. This training uses margin ranking loss and stochastic gradient descent.

To evaluate, Weston et al. [59] used three different experiments: (1) a large scale QA dataset, (2) a simulated world QA setting, where there are some statements referencing places, objects, and people, and the model has to answer questions about these statements, and (3) a combination of the previous two experiments. The results showed that the model was able to answer both general knowledge questions and specific statements referencing to previous dialogue, outperforming standard RNNs.

In Sukhbaatar et al. [54], was developed an end-to-end architecture based on memory networks for the question-answering problem. This approach uses a recurrent attention mechanism to read the memory that can be successfully trained via back-propagation. This approach is similar to the one in [59], but the *argmax* operations in each layer are replaced by a continuous weighting from the *softmax*. This method requires less supervision, surpassing other baselines with the same level of supervision, such as the original memory networks by Weston et al. [59].

Other applications of memory networks to model conversational context include Visual Dialog [11], which uses both images and text, and the Wizard of Wikipedia [14]. In Visual dialog, a memory network is used to keep track of previous question-answer embeddings generated by an LSTM about an image. In Wizard of Wikipedia, memory networks are used in conjunction with BERT embeddings to maintain the conversational context, with the objective of simulating a conversation.

2.6.3 Conversational Query Rewriting

Another way of maintaining the conversational context is by using a query rewriting method to convert conversational queries to context-independent queries. This rewriting is particularly important in a retrieval system because a conversational query generally lacks all of the information needed to search for the correct information, as a result of the use of omissions and references to the context of the conversation.

To address the identified problems in [15] was created the CANARD dataset by manually rewriting the conversational queries in QuAC [3] to form context-independent queries to train models in the conversational query rewriting task. In particular, Elgohary et al. [15] used a sequence-to-sequence model with an attention and copy mechanism, which is fed with the full conversational history and the query to be rewritten.

Similarly in [58], it is fine-tuned a BERT model on the binary term classification task that aims to add to the current query terms from previous turns. The model was

trained using distant supervision labels, which reduced the amount of human-curated data needed for this task. In particular, after obtaining the BERT embeddings a linear layer is applied to classify if a token in the history is relevant for the query, being concatenated to the query if the model considers the term relevant.

As a final example, in [30], a T5 [47] model is fine-tuned on the CANARD dataset to generate context-independent queries. As in the previous approaches, the input to the model is the full conversational history, and the target is the query rewritten. This model achieved state-of-the-art results in the conversational query rewriting task that uses the CANARD dataset [15]. The researchers attributed these results to the better language representation and pre-training that the T5 model has over the BERT model [30].

2.7 Conversational Systems

2.7.1 Conversational Question Answering Systems

The task of a conversational QA system can be defined as given a passage p, the k-th question q_k in a dialog, and the conversation history H_k, answer q_k by predicting an answer span a_k within p. In this section, we present some systems and the approach that they use to solve the conversational question answering task.

2.7.1.1 FLOW for Conversation Question Answering

In Huang et al. [21] is presented FlowQA, a model consisting of two main components: a neural model for single-turn machine comprehension, and FLOW, a mechanism used to encode the conversational history, to use the intermediate representations created in previous turns in the conversation. The full architecture is presented in detail in [21], but summarizing is composed of a question and context encoding step, followed by a reasoning step, which includes various integration-flow layers and attention (first on the question and after on the context), being the final step the answer prediction. In our task, the interest is primarily on the integration-flow layers. Each of these layers is composed by a context integration layer and a FLOW component. In the context integration layer, the current context representation C_i^h for each question i in layer h is passed into a Bi-LSTM layer [20]. In the FLOW component, the output of the context integration layer is reshaped and passed through a GRU [2]. The final output of the integration-flow layer is the concatenation of the integration layer and the FLOW outputs. So, in the end, we have a representation of both the context and its interactions with the current query.

In Yeh and Chen [66] is introduced an extension to FLOW [21] called FlowDelta. This model is used to explicitly model information gain during dialogues. In FlowDelta, researchers assume that the difference between previous hidden representations indicate a change in flow. An example used in FlowDelta [66] is considering 3 consecutive questions Q_{k-2}, Q_{k-1}, Q_k, in which all discuss the same event, but Q_{k-2} is about another topic. The researchers expect the hidden state $\{h_{k-1,j}, h_{k-1,j+1}, \ldots, h_{k-1,l}\}$ of the span in turn *k-1* to be

dissimilar to the hidden state in the turn *k-2* because their topics are different. To model this, FlowDelta changes the computation of FLOW in Huang et al. [21] to also use the difference between turn embeddings.

2.7.1.2 Pre-trained Models for Historical Question and Answer Embeddings

With the emergence of the large pre-trained language models of Section 2.3, researchers have studied ways of incorporating these models into conversational question answering systems. An example of this is the historical answer embedding (HAE) model from [45]. Qu et al. [45] use a BERT BASE [12] model, packaging the question and passage into a sequence, obtaining the respective embeddings. To handle the conversational history, the researchers give tokens extra embedding information by including a history answer embedding (HAE) layer. In this layer two unique answer embeddings are learned that indicate if a token is part of the historical answers or not.

In [46] is proposed an extension to HAE that uses a positional history answer embedding method (PosHAE), and a history attention mechanism (HAM) that attends with different weights to each historical turn. To perform the positional history answer embeddings, researchers used BERT to encode the question q_k, the passage p, and the conversational history H_k into a contextualized representation, where k represents the current turn in the conversation. So, given a training instance represented by the triple (q_k,p,H_k), the model generates *k-1* variations, where each one contains the same question and passage, but with a different turn from the history. Before passing that information to BERT it is also added a positional history answer embedding called PosHAE [46]. This method embeds the history answer a_i, into the passage p, extending the work done in [45] with positional information. Figure 2.8 demonstrates the encoding step. The input for the history attention mechanism (HAM) is the token-level and sequence-level representations for all the variations of the same training instance. This history attention mechanism is a single-layer feed-forward network that learns an attention vector $D \in \mathbb{R}^h$, where h is the hidden size of the token representation. This attention vector is then used to map a sentence representation s_k^i to a *logit*, that is the input to a *softmax* function that computes the probabilities across all sequences generated by the same instance. The experiments

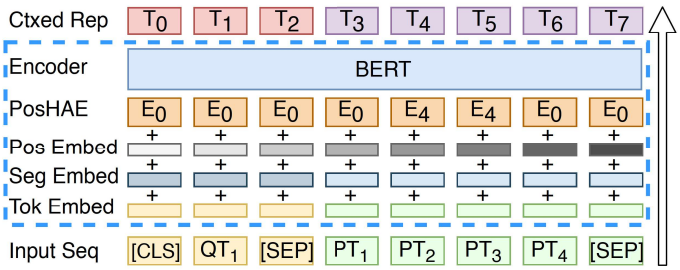


Figure 2.8: Encoding step using PosHAE. QT_i/PT_i denote question/passage tokens. This figure can be of training example (q_6,p,H_6^2). E_4 and E_0 are the history embeddings for tokens in and not in H_6^2, respectively [46].

performed in the QuAC dataset [3] showed that BERT+PosHAE improved significantly on the results of HAE, [45] attributing these improvements mainly to the history attention mechanism (HAM) [46].

In Ohsugi et al. [42] is proposed a method that consists of two main steps: contextual encoding and answer span prediction. In the contextual encoding step, a BERT model is used to independently obtain paragraph representations that are conditioned by the current question and each of the previous questions and answers. This process has three parts: (1) the current question is encoded, (2) each question of the history is encoded, and (3) each of the previous answers is encoded. After the encoding part, a function is used to extract features from the results given by BERT, which correspond to the segment of the paragraph in the final hidden states. In the answer span prediction step, the features extracted are passed through a Bidirectional-GRU (BiGRU) generating $M^{(1)}$ that is used to calculate the start index of the span using a linear layer and the *softmax*. To calculate the end index, $M^{(1)}$ is passed through another BiGRU generating $M^{(2)}$, which is also used as input to another linear layer and *softmax*. Figure 2.9 represents the complete approach, also including the answer type prediction, necessary for the task where the model was evaluated.

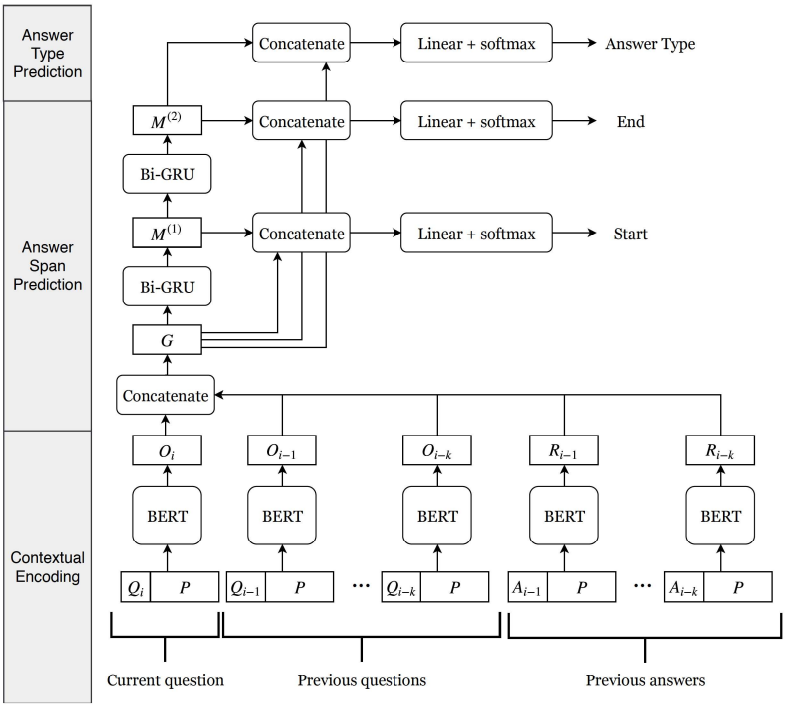


Figure 2.9: Complete model by Ohsugi et al. [42].

In Ju et al. [23] is presented a conversational QA system that is comprised of a RoBERTa [31] model, Adversarial Training (AT), and Knowledge Distillation (KD). For our task, the most relevant part of this architecture is the incorporation of context in the

RoBERTa model. In more detail, the input to the RoBERTa model is built as:

$$\text{“}[CLS]\ Q_k^*\ [SEP]\ C\ [SEP]\text{”}\,, \tag{2.22}$$

where C is the context of the question (i.e., a passage), and Q_k^* is the concatenation of the history of question-answer pairs (turns) and the current question. To fine-tune the model was used a rationale tagging multitask by using a fully connected layer where the objective is to label each token of the paragraph as 1 if it should be included in the rationale and 0 if not. Another detail is that since the research was conducted using the CoQA dataset [50], possible answers also include Yes/No/Unknown beyond just passage extraction, so an additional classification layer is built on top of RoBERTa to predict these types of answers. The details about the Adversarial Training and Knowledge Distillation parts of the system can be seen in the original paper [23]. At the time of writing, [23] is currently in first place in the CoQa dataset [50].

2.7.2 Conversational Search Systems

Conversational search systems, as formally defined in Section 1.2, are different from the conversational QA systems discussed in the previous section because of the retrieval step that needs to be performed over a knowledge-base. In particular, while the conversational QA systems extract a span from an already provided passage, in conversational search, there are possibly millions of passages from which to retrieve, making this a more challenging task.

In [4] is presented a conversational search system that uses the coreference resolution model from AllenNLP [27] to perform query rewriting, the BM25 [51] retrieval model with the pseudo-relevance feedback model RM3 [32] to perform the first-stage retrieval over the dataset, and a BERT model to perform the re-ranking [12]. The results of the re-ranker were particularly relevant, showing that with limited training data, in this case, the TREC CAsT 2019 dataset [10], it is difficult to train an effective BERT re-ranker model, and for this reason, it is better to use a fine-tuned retrieval model.

Voskarides et al. [57] propose a query expansion model based on a notion of word *centrality* (the most important words) and word *recency* (words that appear in the most recent turns). To calculate word *centrality*, the researchers build an undirected graph where the nodes are the words in the queries, and the edges are the cosine similarity between the nodes (words) using the pre-trained word embeddings from word2vec [34]. Word *recency* is calculated using an expression that accounts for the turn where the words appear [57]. So the final score for a word is the combination of the *centrality* and *recency* scores. Then the top-k words ranked by this score are added to the current query. To complement the query expansion technique, [57] also uses a BERT re-ranker following [37], explained in Section 2.5.4, to obtain a score for each query-passage pair. This score is then mixed with the score given by the LMD [67] retrieval model to obtain the final score for a passage.

In [63] is presented two different approaches to the conversational search task called: Historical Query Expansion and Historical Answer Expansion. In the first approach, the researchers use the BM25 [51] score for each word in the query to identify the most relevant words in the conversation. They consider two levels of relevance: session words, which are always added to each subsequent query, and query words, which are added only from the last three queries. The other approach, Historical Answer Expansion, uses the pre-trained BERT model from [37] to estimate the query-passage scores, which are then mixed with the scores from the previous turn using a parameter (λ). This provides a simple way of combining the score given by BERT to the current query-passage pair with the scores obtained in the previous turn. Regarding the results, the Historical Query Expansion method was the best performing system in TREC CAsT 2019 [10].

2.8 Critical Summary

In this section, we discuss all of the methods presented in this chapter.

The traditional first-stage retrieval algorithms, such as BM25 and query likelihood models, are already established in the literature, basing its scoring functions on the term frequency of the words in the document and index.

The construction of the index is still a matter of study as evidenced by recent works, such as DeepCT [6], doc2query [41], and docTTTTTquery [39], that aim to increase the performance of the well studied first-stage retrieval algorithms without incurring in an increase in retrieval time.

In terms of re-ranking, we introduced models that work with word embeddings [8, 19, 61]. However, the state-of-the-art methods involve contextual word embeddings, which can capture complex interactions between terms beyond simple term matching. These embeddings can be obtained by using pre-trained models such as BERT [12], RoBERTa [31], T5 [47], and XLNet [65], which can then be further fine-tuned on a specific task for better performance. These types of models are now the norm and marked a very important milestone in retrieval and natural language processing. The development of these large models is currently an active subject, being aided by the increase in computational power, and the availability of large collections of text, although the resources needed to develop these types of models are very demanding.

We also analyzed state tracking components that can model the conversational context in different ways. In specific, we discussed Hierarchical RNNs [52], Memory Networks [54, 59], that have the advantage of keeping all of the relevant parts of the context in memory, and query rewriting approaches that aim to rewrite the queries using the context present in the history [15, 30, 58].

With the development of specific datasets for the question-answering task [3, 49, 50], many systems were created that leveraged the advances in natural language processing, and in specific, the use of the pre-trained models mentioned earlier, to model the history of the conversation [23, 42, 45, 46]. However, these models are developed for a task where

the answer is in a single provided document, being only necessary to extract a span, while in a conversational search setting, there are possibly millions of documents.

We also discussed some conversational search systems [4, 57, 63], which showed that using the context from previous queries is very important to achieve good results.

This thesis aims to utilize the knowledge described in this chapter and increase the comprehension of these techniques in a conversational search setting, where their application is still an open research topic.

Chapter 3

Indexing and First-Stage Retrieval

3.1 Introduction

The first step in an information retrieval system, conversational or not, is to retrieve relevant documents to a particular query. This method for each query returns a ranked list of results with the ones the model considers the most relevant at the top.

Regarding the indexing stage, there are simple changes that can be applied to the original data to improve retrieval performance, such as stemming and stop word removal. Another option is document expansion techniques. Here the aim is to expand documents with words or sentences that are relevant for that particular document. Depending on the chosen algorithm, this can be a very expensive technique, however, this calculation is performed before indexing the documents, having little impact on retrieval time in a live system.

When designing a simple retrieval system, without a second step for re-ranking, the focus is to have a model that is able to place the more relevant documents first, but since in our approach we want to apply a theoretically superior and computational more expensive re-ranking model, we instead focus on retrieving the largest amount of relevant documents possible. Consequently, our main focus in the first-stage retrieval step is to achieve the highest recall.

In this chapter, we explore several indexing strategies, document expansion methods, and retrieval models. The choices of the different methods and algorithms presented in this chapter are the result of the study of the related work and thorough experimentation.

The results obtained with the methods present in this chapter are comprehensively examined in Chapter 6, where their robustness and performance are analyzed across different metrics. Figure 3.1 represents a simple view of when each of the methods is applied.

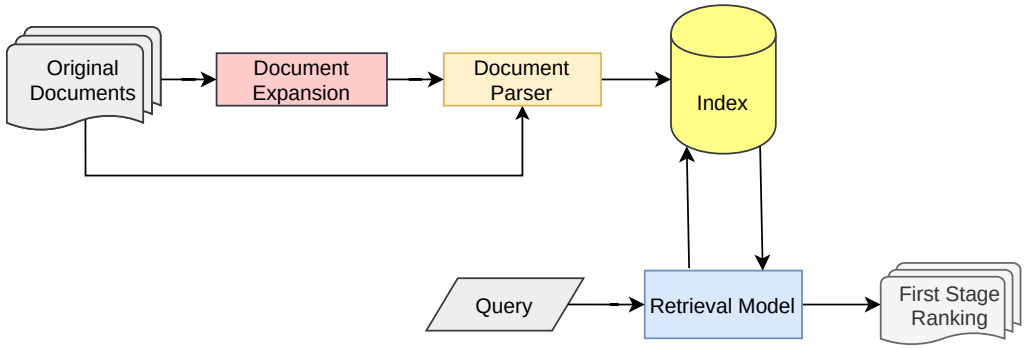


Figure 3.1: Simple view of the architecture demonstrating where in the pipeline the document indexing and retrieval methods are used.

3.2 Indexing and Document/Passage Expansion

3.2.1 Document-Passage Parser

The first step is to decide the best way to index the data. This data, comprised of various documents/passages, can be stored as-is, or we can apply a simple but yet effective modification to the text to improve the performance of the retrieval algorithms and diminish the vocabulary mismatch between the user's queries and the document's words. This process is performed offline and is comprised of straightforward techniques like tokenization, stemming, and stop word removal. As a control experiment, we index the documents with all the words lowercased and using a standard white-space and punctuation tokenizer. Since the documents to be indexed are passages, with a relatively short length, the influence of stop words can be noticeable and introduce noise when searching for relevant documents. To avoid this, we used two different stop word removal lists. The first one is from Indri[1] and the second one is from Lucene _english_stopwords[2]. From the analysis of both lists, we saw that Indri has a more extensive list of stop words (418 words) when compared to the stop words list from Lucene (35 words), providing a way of assessing the impact of stop words in the performance of the retrieval model.

We also experimented with the use of a stemmer to increase the number of matches between the query and document words by converting words to their root form. The stemmer used was the KStem[3]. The KStem is characterized by reducing the word to its stem but avoiding conflating variants that have different meanings, such as "memorial" and "memorize" that reducing to "memory" would overlap their representations.

Finally, our last indexing strategy utilizes both stop word removal and stemming. Table 3.1 shows an example of a query before and after each of the previously described method is applied.

[1] https://github.com/igorbrigadir/stopwords/blob/master/en/indri.txt
[2] https://github.com/apache/lucene-solr/blob/master/lucene/analysis/common/src/java/org/apache/lucene/analysis/en/EnglishAnalyzer.java#L46
[3] http://lexicalresearch.com/kstem-doc.txt

Table 3.1: Effects over a query using the different indexing strategies.

Original Query	**When and where is the next World Cup being played?**
Standard Tokenizer	when and where is the next world cup being played
Stop Words Indri	world cup played
Stop Words Lucene	when where next world cup being played
Stemmed	when and where is the next world cup being play
Stop Words Lucene and Stemmed	when where next world cup being play

3.2.2 Document/Passage Expansion

An alternative approach to improve retrieval efficiency is to expand documents with terms that represent their content. This expansion can help with problems such as vocabulary mismatch, which occurs when a user utilizes terms in the query that are different from the ones in the document. It can also increase the term frequency of the more important terms, influencing the score of the document calculated by the retrieval models described previously. Document expansion also has the advantage of not incurring in a significant increase in query time, since the expansion is done prior to indexing.

In our work, we used two neural document expansion techniques, doc2query [41], and its improved version docTTTTTquery [39]. Both of these techniques predict queries that can be asked given a document. Following the same approach as in [39, 41], we used these techniques to predict k queries for each document, and then before indexing, we concatenate these queries to the text of the original document, separating them by a special token. The special token is used to identify where the original document ends, which can be relevant for later steps. As a result of the expansion, the more important terms (appearing in the predicted queries) have their term frequencies increased, and since the documents are expanded with queries, there also exists the possibility of achieving a greater match between the terms in the queries, and the terms in the document.

Table 3.2 shows an example of 5 queries predicted for a document using both document expansion techniques described. From this table, we observe that the queries predicted using docTTTTTquery seem to be more diverse, capturing the subject of the document in a better way as also evidenced in the original work [39].

3.3 Retrieval Models

The responsibility of the retrieval models is to fetch documents given a query from a collection of possibly millions of documents in an efficient and fast fashion. Retrieval models are generally simple and unsupervised, basing their ranking in the query terms in relation to their document-term frequency, collection-term frequency, and length of the documents in the collection.

Choosing a good retrieval model is important for the effectiveness of our approach, so we conducted a study over three different models frequently used in information retrieval.

Table 3.2: Example of 5 predicted queries for a MS MARCO document using doc2query and docTTTTTquery.

Document	The Manhattan Project and its atomic bomb helped bring an end to World War II. Its legacy of peaceful uses of atomic energy continues to have an impact on history and science.
Predicted Queries doc2query	1. why was the manhattan project created 2. what was the manhattan project 3. why was the manhattan project important 4. why was manhattan an important factor 5. what was the result of the manhattan
Predicted Queries docTTTTTquery	1. what were a major contributions to the manhattan effort 2. what impact did the manhattan project have on history 3. what is the manhattan project impact on world 4. what helped to end world war ii 5. why did the manhattan project help end world war ii

The first model is BM25 [51], a probabilistic model described in detail in Section 2.4.2.1. The second is LMD (Language Model Dirichlet), and the third is LMJM (Language Model Jelinek-Mercer). These last two are query likelihood models that estimate a language model for each document, using different types of smoothing that change the way the ranking score is calculated for each document. In particular, in [67] it was found that LMD works best with short queries and LMJM with longer queries. The description of these models is provided in Section 2.4.2.2.

As stated before, these are classical information retrieval models that don't have any real understanding about the text present in the query and document, besides term matching, but have proved to be effective and are currently used in production systems like ElasticSearch[4]. In our work, as in others [8, 19, 37, 61], these simple models are used to retrieve a small set of documents from a collection of millions, that can then be re-ranked by a more complex neural model.

3.4 Summary

Figure 3.2 demonstrates where each of the techniques described are applied. It also shows the execution pipeline, from when the query is issued, to the moment where we obtain a first list of ranked documents.

The top half of the Figure describes the document expansion techniques and indexing strategies that we used to create the index. From the original collection of documents, we can apply any or none of the document expansion techniques. These algorithms, as explained before, are computationally expensive and aim at improving recall by increasing the frequency of the more relevant terms and by decreasing the vocabulary mismatch between the words in the user's query and the words in the documents.

[4] https://www.elastic.co

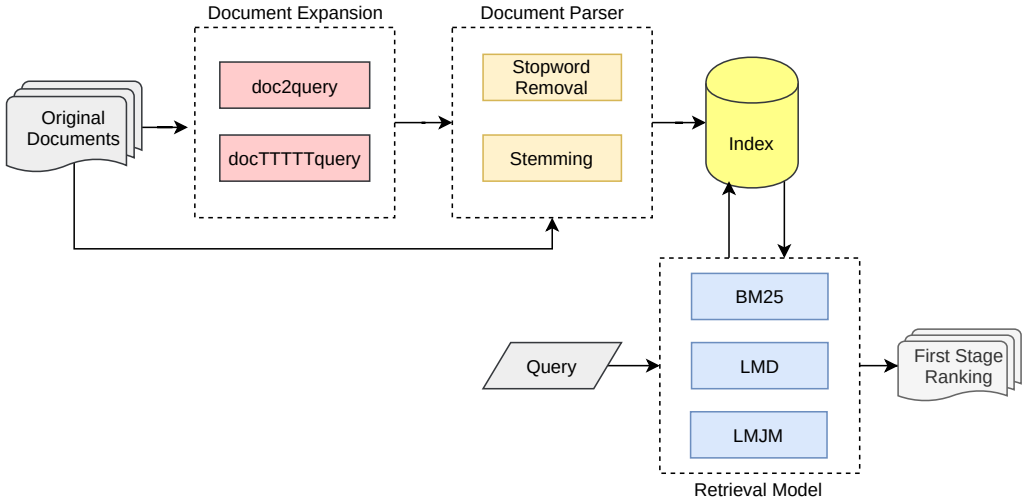


Figure 3.2: Indexing and document expansion approaches (top half). Retrieval models for first-stage retrieval architecture (bottom half).

The indexing strategy is defined by applying simple modifications to the content of the documents. We can perform stop word removal with different stop word lists and utilize different stemming algorithms to increase the number of matches between query and document terms. All of these operations are performed offline (before query time), so they don't influence the retrieval time in a significant way. With all of the documents parsed using these methods, we can proceed to create the index.

The bottom half of Figure 3.2 shows the steps to obtain an initial ranking. This is a straightforward process since we only need to pass the query through a retrieval model. As explained, we considered the typical information retrieval models BM25, LMD, and LMJM to query the index. In the end, we obtain a list of ranked documents ordered by their score that can be given to the user directly. Although it can be used in practice, this list is obtained by models that lack the comprehension of text needed to achieve the best results. So, to improve performance, at the expense of an increase in retrieval time, we can apply more complex neural re-ranking algorithms.

In our work, since we are building a conversational system, the use of the original query is often not sufficient to answer the user's information need, so various query rewriting methods will be introduced in Chapter 4.

Chapter 4

Conversational Context as Query Rewriting

4.1 Introduction

The retrieval models and architecture proposed in Chapter 3 can be enough for many information retrieval needs but, due to the conversational aspects of this work, the query can be in a format that doesn't have all of the information needed to answer the user's query. An example of this is the conversation presented in table 4.1. In turn 2, the system needs to understand that "its" refers to "Lisbon's" (explicit coreference). Another even more challenging example is presented in turn 3 since the monuments retrieved should be in Lisbon, despite no direct evidence in the current query text (implicit coreference). In a regular information retrieval system, it is almost impossible to retrieve relevant information for turns 2 and 3 of the conversation because there is no notion of the context of the conversation. To solve this problem one of the most crucial tasks is to rewrite the queries to make use of context.

Table 4.1: Conversation example about a specific topic, in this case, the city of Lisbon.

Turn	Conversational Query	Non-conversational Query
1	How is the climate in Lisbon?	How is the climate in Lisbon?
2	Tell me about its origins.	Tell me about Lisbon's origins.
3	What monuments should I visit?	What monuments should I visit in Lisbon?

To perform this query rewriting task, one of the first things that needs to be tackled is to perform coreference resolution to disambiguate entities, and include context from previous turns to search for the correct information. Still in the queries, it is also possible to perform query expansion, which means adding new terms to the query that can help

retrieve relevant documents. To achieve this, we utilized the pseudo-relevance feedback model RM3 [26].

Figure 4.1 represents a simple view of where each of the methods is applied, and in Chapter 6, we present the results obtained with these methods.

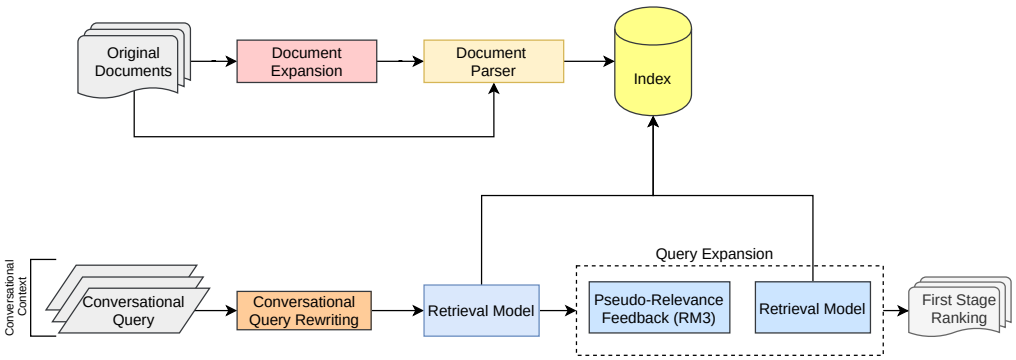


Figure 4.1: Simple view of the architecture demonstrating where in the pipeline the query rewriting and expansion methods are used.

4.2 Conversational Query Rewriting

The goal of conversational search is to provide a more natural interaction with the agent. In a normal conversation, people tend to be succinct, making use of coreferences to previous questions and answers, while also having a notion of the flow and context of the conversation at any given moment, as seen in table 4.1. Our system must be able to understand the various nuances of the conversation and rewrite the current question to make the context explicit.

In this section, we explain the implemented query rewriting techniques, both in isolation and in conjunction with each other, in order to address the context tracking requirements of conversational query rewriting.

4.2.1 Query Rewriting with Previous Queries

In conversational search, the incorporation of previous queries can be seen as a simple way of giving context to the current query. If the conversation follows the typical information-seeking pattern of first exploring a general concept and then following this with a more detailed search about that concept or a related one, we can use the previously mentioned terms to expand the query. With this in mind, we structure the three following approaches:

- **Prefixing** - Prefixes the first query issued by the user to the current query. We use the first query since it is non-conversational and it is usually the starting point for the subsequent queries. One of the problems of this approach is that in the event of a topic shift during the conversation, the first query may introduce noise.

- **Full-Union** - Performs the union of the current query with all previous queries, creating longer queries as the conversation advances. It is expected that most of the text in the query will be noise, but all the relevant concepts needed to specify the context will be present. This approach is meant to show the importance of getting the right context for the current query instead of using the whole context.

- **Union** - Performs the union of the current query with each of the previous queries separately, performing n queries (that are easily parallelizable) depending on the turn depth t following equation 4.1:

$$n = \begin{cases} 1, & if \;\; t \leq 2 \\ t-1, & if \;\; t > 2 \end{cases} \quad t \geq 1, \tag{4.1}$$

  After this, we make a union of the results of all the queries and keep the top k documents retrieved with the highest retrieval scores, making this a fusion algorithm. With this approach, we expect to obtain the most relevant documents for each combination of queries, removing the documents that appear lower on the rank for each query.

An example of each of the described approaches is presented in table 4.2.

Table 4.2: Example of incorporation of previous turns terms. The history of queries represents the queries issued so far by the user (turn depth 3). The other rows represent 3 different approaches of using previous turns to rewrite the current query.

History of queries:	1. Tell me about the Bronze Age collapse? 2. What is the evidence for the Bronze Age collapse? 3. What are some of the possible causes?
Prefixing	1. Tell me about the Bronze Age collapse? What are some of the possible causes?
Full-Union	1. Tell me about the Bronze Age collapse? What is the evidence for the Bronze Age collapse? What are some of the possible causes?
Union	1. Tell me about the Bronze Age collapse? What are some of the possible causes? 2. What is the evidence for the Bronze Age collapse? What are some of the possible causes?

4.2.2 Coreference Resolution

Coreference resolution is applied before performing any expansion on the query or search in the dataset. We aim to resolve the coreferences in order to search for the correct information. To achieve this, we used the toolkit provided by AllenNLP [18]. This toolkit has many different functionalities, such as named entity recognition, reading comprehension, coreference resolution, and others. In our work, we utilized the coreference resolution module that uses the implementation from Lee et al. [27] but replacing GloVe embeddings with BERT embeddings.

To encode the input, Lee et al. [27] used a vector representation for each word derived from trained word embeddings and a 1-dimensional Convolutional Neural Network

(CNN) over the characters. This representation is then passed through a bidirectional LSTM [20] to compute the vector representation for the span. After this, an attention mechanism is applied over the words in each span, to obtain a representation of the full span. After obtaining the span representation, a scoring architecture based on feed-forward neural networks (FFNN) is used to calculate $s_m(i)$, the unary score of span i being a mention, and $s_a(i,j)$, the pairwise score of span j being an antecedent of span i. The final score for the span $s(i,j)$ is given by $s_m(i) + s_m(j) + s_a(i,j)$ if j is an entity mention and is coreferent to i, or 0 if j is not a mention or is not coreferent to i.

In the AllenNLP [18] toolkit, the coreference resolution algorithm described in the previous paragraph receives sentences as input, and outputs a list of mentions that indicate which spans in the input belong to the same mention. With the notion that the first mention is the most descriptive of all, we developed two approaches:

- **Coref** - In this first approach, all the spans with the same mentions are replaced by the first one. We analyzed the results of this approach and noticed that the algorithm sometimes attributes the same mention to concepts that are not related (coreferent).

- **Coref-Pronouns** - To mitigate the problem identified in the previous approach, we only replace the mention if it is needed, i.e., when pronouns are present in the mention. Using this approach, mentions that give some information about the subject remain unchanged.

Figure 4.2 shows the mentions detected by the coreference resolution model (same color same entity) on a sentence. Table 4.3 shows the application of the two described approaches at the same sentence.

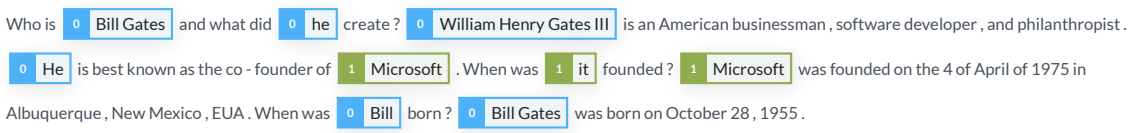


Figure 4.2: Mentions detected by AllenNLP coreference resolution algorithm [27].

4.2.3 Conversational Query Rewriting using the Text-To-Text Transfer Transformer (T5) Model

As explained in Section 2.3, the T5 model is a large, pre-trained, encoder-decoder model, with a simple text-to-text input-output format that can be fine-tuned in various tasks, such as machine translation and question answering.

In our work, we use the characteristics of T5 to rewrite the queries in a conversational format to form non-conversational queries. To achieve this, the model must be capable of resolving the coreferences and provide the necessary context to queries that

Table 4.3: Results of applying the developed coreference resolution methods.

Coref	Q1: Who is Bill Gates and what did Bill Gates create? A1: Bill gates is an American businessman, software developer, and philanthropist. He is best known as the co-founder of Microsoft. Q2: When was Microsoft founded? A2: Microsoft was founded on the 4 of April of 1975 in Albuquerque, New Mexico, EUA. Q3: When was Bill Gates born? A3: Bill Gates was born on October 28, 1955.
Coref-Pronouns	Q1: Who is Bill Gates and what did Bill Gates create? A1: William Henry Gates III is an American businessman, software developer, and philanthropist. He is best known as the co-founder of Microsoft. Q2: When was Microsoft founded? A2: Microsoft was founded on the 4 of April of 1975 in Albuquerque, New Mexico, EUA. Q3: When was Bill born? A3: Bill Gates was born on October 28, 1955.

are not specific enough. To accomplish this, our approach follows the one presented by Lin et al. [30]. This approach consists of fine-tuning the T5 BASE model using the CANARD dataset [15]. CANARD is a dataset created by rewriting the questions in QuAC [3] to form non-conversational questions. These questions rewrites were done by human crowdsourced workers to form context-independent natural questions that preserve the structure of the original query. This resulted in a dataset with 31.538, 3.418, and 5.571 query rewrites for training, development, and test sets, respectively.

To train the T5 model it is necessary to provide an input sequence and a target sequence given as strings. With this notion, we constructed the input using two different formats, where i is the current turn, q is a query, p is a retrieved passage, and *[CTX]* and *[TURN]* are special separator tokens. *[CTX]* is used to separate the current query from the context (previous queries and passages), and *[TURN]* is used to separate the historical turns. In particular, we created two input formats:

- **Utterance-Separated** - Uses as input sequence the concatenation of the history of queries using a special token *[TURN]* to separate every historical utterance. After this, we prefix the current conversational query to the history, separating them with the special token *[CTX]*:

$$\text{“}q_i\ [CTX]\ q_1\ [TURN]\ p_1\ [TURN]\ q_2\ [TURN]\ p_2\ [TURN]\ \ldots\ q_{i-1}\ [TURN]\ p_{i-1}\text{”}. \quad (4.2)$$

- **Turn-Separated** - Uses the same strategy as *Utterance-Separated* but it adds the special token *[TURN]* every two historical utterances (query-answer pair):

$$\text{“}q_i\ [CTX]\ q_1\ p_1\ [TURN]\ q_2\ p_2\ [TURN]\ \ldots\ q_{i-1}\ p_{i-1}\text{”}. \quad (4.3)$$

The target in both input formats is the question rewritten, and the model is trained using the standard maximum likelihood [47].

Examples of the inputs, targets, and predicted queries are presented in table 4.4. The first two examples are from CANARD, and the last two are from TREC CAsT [9], the conversational search dataset that we introduced in Section 2.2.1.1, and that we will

evaluate in more detail in Section 6.1. The way the input is created only affects the CANARD dataset because in TREC CAsT 2019 the historical utterances don't depend on the responses of the system. As we can see, T5 is capable of resolving ambiguous queries in most situations, however, it sometimes mistakes similar entities when multiple are involved, as evidenced in [15] and in table 4.4 example 3, where the model predicts "throat cancer" instead of "lung cancer". We can also note that this model is more robust than simple coreference resolution, as we can see in example 4, where it includes the words "Bronze Age Collapse", even though there is no particular pronoun or coreference mention (implicit coreference).

Comparing the T5 model to the previously described AllenNLP coreference resolution model, we can see that both are able to resolve explicit coreferences, i.e., when a particular pronoun or name is used to refer to a previously mentioned entity. The main advantage of this trained T5 model over AllenNLP's model is the ability to perform implicit coreference and context resolution, which is very important in a conversational search system.

Table 4.4: Example of T5 query rewriting: inputs, targets, and predictions.

	CANARD
Original Query	What was his agreement with McMahon?
T5 Input Query *(Utterance-Separated)*	What was his agreement with McMahon? [CTX] Superstar Billy Graham [TURN] Return to WWWF (1977-1981) [TURN] Why did he return to the WWWF? [TURN] an agreement with promoter Vincent J. McMahon Senior
T5 Predicted Query	What was Superstar Billy Graham's agreement with McMahon?
Target Query	What was Billy Graham agreement with promoter Vincent J. McMahon?
Original Query	What was his agreement with McMahon?
T5 Input Query *(Turn-Separated)*	What was his agreement with McMahon? [CTX] Superstar Billy Graham. Return to WWWF (1977-1981) [TURN] Why did he return to the WWWF? An agreement with promoter Vincent J. McMahon Senior
T5 Predicted Query	What was Superstar Billy Graham's agreement with McMahon?
Target Query	What was Billy Graham's agreement with McMahon?
	TREC CAsT 2019
Original Query	What are its symptoms?
T5 Input Query	What are its symptoms? [CTX] What is throat cancer? [TURN] Is throat cancer treatable? [TURN] Tell me about lung cancer.
T5 Predicted Query	What are lung cancer's symptoms?
Target Query	What are throat cancer's symptoms?
Original Query	What are some of the possible causes?
T5 Input Query	What are some of the possible causes? [CTX] Tell me about the Bronze Age collapse? [TURN] What is the evidence for the Bronze Age collapse?
T5 Predicted Query	What are some of the possible causes of the Bronze Age collapse?
Target Query	What are some of the possible causes for the Bronze Age collapse?

4.3 Query Expansion With Pseudo-Relevance Feedback

Query expansion is the technique of adding new terms to a query. Some methods work on expanding the query based on pseudo-relevance feedback (blind feedback), meaning that they don't need the active participation of the user. These methods can be applied to

both conversational and non-conversational systems to retrieve documents that may be relevant but were not retrieved or were lower in the rank when using the original query.

To select the terms to expand the initial query, we first search the index using the original query provided by the user or the query rewritten using any of the previously mentioned methods. After this, we obtain a list of documents ordered by their relevance score, and we consider the top k documents as relevant. With these k documents, we use the most common n terms to expand the query. The weights of each term in the original query and documents are calculated using one of the relevance feedback models that we describe in the remainder of this section.

In [32], the authors demonstrated that RM3 is one of the most effective relevance feedback methods and also one of the more robust to parameter setting. Before introducing RM3, we first describe one of its parts, RM1 [26]. RM1 assumes independence between query words and relevant document words. Equation 4.4 shows how to calculate RM1, where Θ represents the set of document models, M_d represents a document's model, $p(w|M_d)$ represents the relevance of a term w, and $p(q_i|M_d)$ represents the relevance of a query term q_i:

$$p_{RM1}(w|\Theta) = \sum_{M_d \epsilon \Theta} p(w|M_d)p(M_d)\prod_{i=1}^{m} p(q_i|M_d). \tag{4.4}$$

RM3 is then given by equation 4.5, where α is the parameter that interpolates the original query model $p(w|M_Q)$ with the relevance model, which in this case is RM1:

$$p_{RM3}(w|Q) = (1-\alpha)\cdot p(w|M_Q) + \alpha \cdot p_{RM1}(w|\Theta). \tag{4.5}$$

It is important to note that query expansion, although used to improve results, makes retrieval slower due to the need for a second search over the index and can introduce noise or drift the query to another topic, hurting overall performance.

An example of a query expanded with RM3 is provided in table 4.5. We can see that the expansion encountered relevant terms such as "megalodon", but it also introduced other terms that can drift the query to another topic like "whale", "blue", and "million".

Table 4.5: RM3 expansion of a query using α=0.2, 20 feedback documents and 10 feedback terms. The numbers indicate the weight given to each term.

Original Query	What is the largest shark ever to have lived on earth?
Rm3 Query	(shark)$^{0.1583}$ (largest)$^{0.1555}$ (earth)$^{0.1282}$ (ever)$^{0.1247}$ (what)$^{0.1142}$ (have)$^{0.1142}$ (live)$^{0.1142}$ (whale)$^{0.0305}$ (blue)$^{0.0145}$ (million)$^{0.0136}$ (megalodon)$^{0.0131}$ (ago)$^{0.0103}$ (animal)$^{0.0081}$

4.4 Summary

In this chapter, we focused on the bottom half of Figure 4.3, which shows the query rewriting and query expansion methods outlined in this chapter.

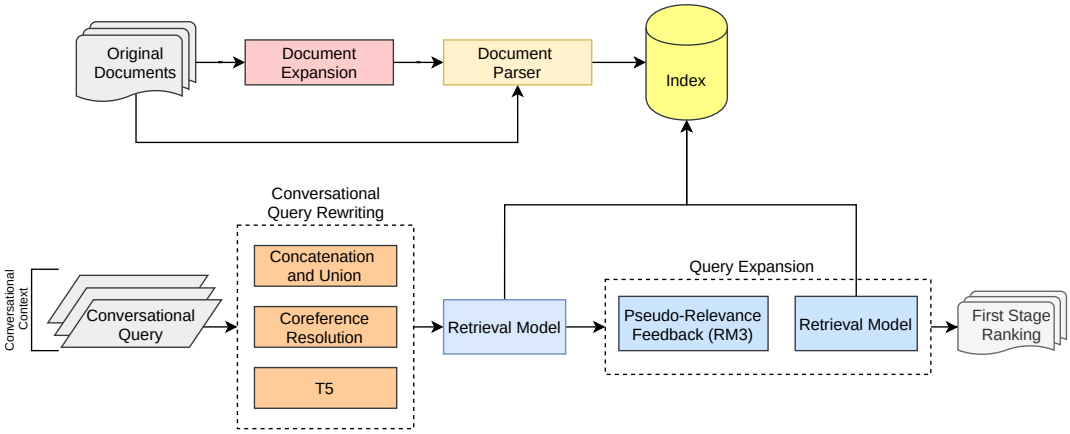


Figure 4.3: Indexing and document/passage expansion approaches (top half). Retrieval, query rewriting and expansion for first-stage retrieval (bottom half).

Describing in more detail the pipeline, the query can be done in a natural language format and can be conversational, making use of context from previous turns. When in a conversational setting, the query is passed through a query rewriting process. In this process, we can apply one or more of the techniques described earlier, such as concatenating previous turns, executing coreference resolution, or using the trained T5 model to provide context to the current query. The use of previous queries is a simple way of resolving context at the expense of a greater amount of noise. The use of a coreference resolution algorithm, such as AllenNLP can perform explicit coreference resolution by replacing pronouns as presented in table 4.1, turn 2, but lacks the ability to add implicit context like in turn 3 of table 4.1, which can be done by using the trained T5 model. We also saw some examples of failed context resolution in table 4.4, which shows that these techniques are not perfect, failing to resolve coreferences or using the wrong mention.

Following the query rewriting step, we do the same process explained in Chapter 3 and perform a search in the index using the retrieval model of our choice. If we opt to use a query expansion technique, such as RM3, after the first search in the index, we perform a second search that uses the query, expanded with terms retrieved from the passages, with the weights computed by the relevance model. This expansion can retrieve more relevant passages, but it can also add irrelevant terms.

After all these steps, we obtain a list of ranked passages that can then be used as input to a re-ranking algorithm, such as the ones that will be explained in Chapter 5. With these extensions to the system's architecture, we are now able to use the system in both conversational and non-conversation scenarios by maintaining context through the rewrite of the user's queries. This is crucial, especially in a conversational search system, since there are possibly millions of passages, and a conversational query is bound to produce poor results if the context is not accounted for.

Chapter 5

Conversational Context-Aware Neural Ranking

5.1 Introduction

In this chapter, we delve into neural ranking models and describe our implementation of the various architectures, algorithms, and adaptions developed to bring these models to a conversational search scenario. In our work, we use the neural ranking models in a re-ranking task, that as the name suggests, happens after having an initial ranking. This initial ranking is typically performed by simpler term-matching-based models that are able to process large amounts of text in a faster fashion, such as the ones presented in Section 3.3.

With the new pre-trained neural language models, such as BERT [12] and others [31, 65] explained in Section 2.3, it is possible to generate contextual embeddings for a sentence and each of its terms. These embeddings can then be used as input to a model to perform re-ranking of the passages. This ranking is usually considered better because the model captures the context of the terms in the query and passage, as well as their interactions, being able to judge more thoroughly if a passage is indeed relevant to a query, not by term matching or frequency, but by the model's pre-trained term embeddings and fine-tuning on this particular task.

The models developed in this chapter use as a foundation a BERT model fine-tuned on a binary classification task [37]. This model achieved good results in non-conversational re-ranking tasks, where there is only one query and a set of passages. However, we believe that the context present in a conversation can be seen in both previous queries and answers, so we extended the model's architecture to also consider the conversational history. In particular, we extended BERT with the proven state-tracking components: RNNs [2, 20, 42] and Memory Networks [54, 59] to maintain a notion of the context of

the conversation while re-ranking. Figure 5.1 shows where in the conversational search pipeline the re-ranking models are used. The results obtained by these methods are presented in Chapter 6, where we perform extensive testing across various metrics to evaluate the effectiveness of each of the proposed architectures.

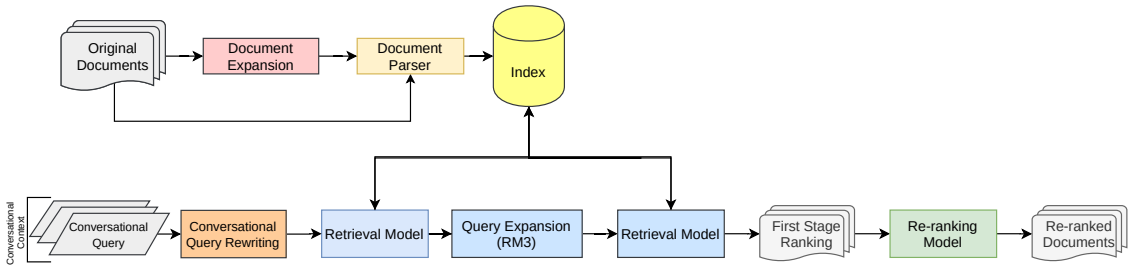


Figure 5.1: Simple view of the architecture demonstrating where in the pipeline the re-ranking model is used.

5.2 BERT Model for Passage Re-ranking

As explained in Section 2.3.1, BERT is a pre-trained language representation model trained on large amounts of data that generates embeddings that capture the context of words in a sentence. These representations alone can be useful for many tasks, such as classification and question-answering, however, the real power of these models comes from fine-tuning on a specific task, which, in turn, improves performance for that task.

In Nogueira and Cho [37], the BERT model is fine-tuned on the passage ranking task through a binary relevance classification task, where positive examples are relevant query-passage pairs, and negative examples are non-relevant query-passage pairs. In our work, we follow the same approach, as presented in Figure 5.2.

The query, q, is truncated to a maximum of 64 tokens and the passage, p, is also truncated so that the concatenation of the query, passage, and separator tokens has a maximum of 512 tokens. It is used a maximum of 512 tokens because this is the maximum number of tokens that BERT can handle as a single input. So, the input to BERT given a sequence of N tokens is given as:

$$emb = BERT(\text{“}[CLS]\ q\ [SEP]\ p\text{”}), \tag{5.1}$$

where $emb \in \mathbb{R}^{N \times H}$ (H is BERT embedding's size) is the embeddings matrix of all tokens, and *[CLS]* and *[SEP]* are special tokens in BERT's vocabulary, representing the classification and separation tokens, respectively. From emb we extract the embedding of the first token, which corresponds to the embedding of the *[CLS]* token, $emb_{[CLS]} \in \mathbb{R}^{H}$. This embedding is then used as input to a single-layer feed-forward neural network (FFNN), followed by a *softmax*, to obtain the probability of the passage being relevant to the query:

$$P(p|q) = softmax(\text{FFNN}(emb_{[CLS]})). \tag{5.2}$$

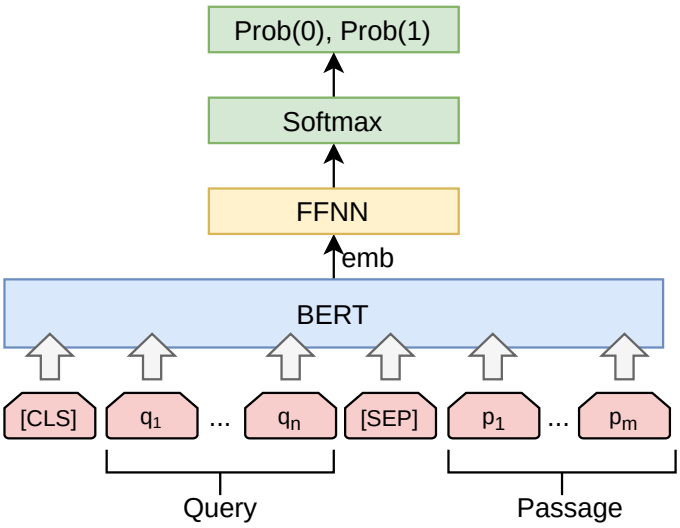


Figure 5.2: BERT re-ranker architecture. The input to BERT is the query concatenated with each one of the passages at a time, using the structure [CLS] q [SEP] p.

With $P(p|q)$ calculated for each passage p, given a query q, the final rank is obtained by re-ranking according to the probability of being relevant, i.e., the closer to a relevance probability of 1, the closer the passage will be to the top of the rank.

This method of fine-tuning BERT has shown successful results in re-ranking tasks, being able to surpass various baselines, achieving state-of-the-art in some of them [36, 37, 40].

An example of the input to the BERT model is presented in table 5.1 for the query "Why is blood red?". In the first passage, there are multiple references to "red" and to "blood", which for a typical retrieval model would be enough to consider this a relevant passage. However, if we read both passages carefully, we can conclude that only the second passage actually answers the question. These subtleties are caught by the BERT-based ranking model giving the second passage a higher probability of being relevant, and so it is put on the top of the rank.

Despite the very good performance of this method in re-ranking tasks, we reckon that in a conversational search setting, there are particular characteristics that could be used to improve the performance of this model, especially when considering the context in queries and answers from previous turns.

5.3 Conversational BERT for Passage Re-ranking

As we saw in the previous section, BERT can be fine-tuned on a passage re-ranking task with good results. Despite this, conversational search is more complex than just passage ranking since we are predominantly working in a multi-turn, sequence-based task. In this scenario, we have to not only manage the current query but also have a notion of the context where the query is posed. We believe that the context can be seen in both previous

Table 5.1: Example of the input and output of the BERT model in the relevance classification task for the query "Why is blood red?". The rank is calculated in the end by ordering the passages in decreasing order of Prob(1).

BERT Input in format "[CLS] q [SEP] p"	**Prob(1)**	**Rank**
[CLS] Why is blood red? [SEP] Red blood cells. The purpose of red blood cells in the blood is to carry oxygen to body cells. The hemoglobin in red blood cells binds to oxygen received from the lungs, and these cells are circulated in the blood to all cells to supply needed oxygen. Blood cells carry oxygen from the lungs to the body's tissues and take carbon dioxide from the tissues back to the lungs.	0.837	2
[CLS] Why is blood red? [SEP] Blood is red because it is made up of cells that are red, which are called red blood cells. But, to understand why these cells are red you have to study them on a molecular level. Within the red blood cells there is a protein called hemoglobin. Each hemoglobin protein is made up subunits called hemes, which are what give blood its red color.	0.993	1

queries and answers, so it is necessary to define a model that can leverage this information. Therefore, while re-ranking, we want the model to consider the importance of the passage to the current query, as well as the importance of the passage given the conversational context. An example of this is present in table 5.2. In turn 2, we want the model to focus on answering the query by retrieving passages that define "extinction event", but push to the top passages that discuss extinction events related to the conversational context, that in this case from the previous turn is the "Cretaceous-Paleogene extinction event".

Table 5.2: Example of the need for conversational context to improve search results. The passages are adapted from the corresponding Wikipedia articles.

Turn	**Query**	**Passages**
1	What is a Tyrannosaurus?	Tyrannosaurus is a genus of coelurosaurian theropod dinosaur. The species Tyrannosaurus rex, often called T-Rex, is one of the most well-represented of the large theropods. It was the last known member of the tyrannosaurids, and among the last non-avian dinosaurs to exist before the Cretaceous-Paleogene extinction event.
2	What is an extinction event?	An extinction-level event (also known as a mass extinction or biotic crisis) is a widespread and rapid decrease in the biodiversity on Earth. An example of this event, the Cretaceous-Paleogene, finished with 75% of all species extinct.

In a conversational search scenario, the conversations can be long, spanning multiple turns, wherein each one there are various passages. This makes the approach of simply concatenating the multiple rounds together infeasible because of the maximum input of the BERT model (512 tokens), as well as the dispersion of topics throughout the conversation (topic changes) that can incorrectly influence the model.

To expand the BERT model to this conversational passage ranking setting, we propose two different architectures based on the study of the state-of-the-art models and related work of Chapter 2:

- **ConvBERT RNN** - BERT is used to generate term and sentence embeddings. An RNN is used to model the conversational context.
- **ConvBERT MemNet** - BERT is used to generate term and sentence embeddings. Memory networks are used to model the conversational context.

5.3.1 ConvBERT RNN

RNNs are generally used to model sequences of data and have been applied in various domains because of their ability to analyze the current input conditioned on previously seen inputs by the use of a hidden state [2, 20, 52]. In ConvBERT RNN, we aim to combine this architecture with the generated BERT embeddings and utilize the RNN's hidden state to model the conversational context between conversational turns.

Using an RNN in a conversational scenario is studied in [53] by using a Hierarchical Recurrent Encoder-Decoder (HRED) architecture. In this architecture, explained in more detail in Subsection 2.6.1, there are two hierarchical RNNs, one that encodes the sentence and another that encodes the context of the conversation. Our model, ConvBERT RNN, or Conversational BERT using Recurrent Neural Networks, uses a similar structure but replaces the bottom RNN (sentence level) with a BERT model that generates the sentence embeddings. This way, we can leverage the better sentence representations given by BERT and use an RNN to maintain the conversational context.

Another paper that served as inspiration for the developed architecture is presented by Ohsugi et al. [42]. But unlike in [42], where the answer to all queries in the conversation is a span of text from a single passage, we are in a retrieval scenario, a more challenging environment, where there are possibly millions of passages. This makes topic shifts more frequent and diverse, since we can have queries unrelated to previous queries and passages, while in the single passage setting the queries are more grounded.

The complete ConvBERT RNN architecture can be seen in Figure 5.3. The Figure represents two turns of information seeking. In the first turn, the query and each of the retrieved passages is encoded by BERT using the same input structure explained in the previous section and represented in equation 5.1. The difference here is that we are training a new model from the start, so we can train it with different parts of the embeddings generated by BERT. In particular, we consider these embeddings:

- **[CLS] token of the last hidden state** - In theory, this token is able to represent the whole sequence since the model was fine-tuned on the relevance classification task using this embedding. This is also one of the most common ways of using BERT [12, 37, 44].

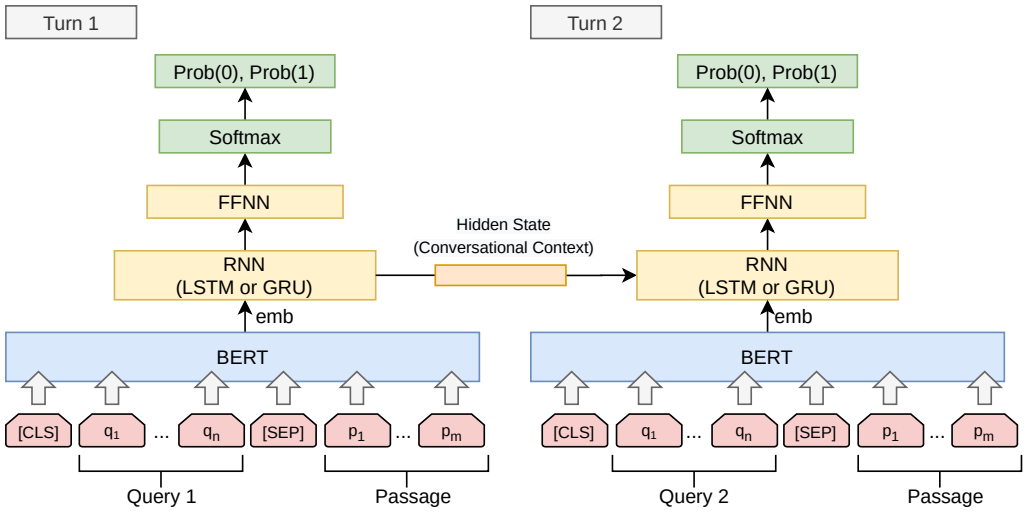


Figure 5.3: ConvBERT RNN architecture. The left side of the figure represents the first turn in the conversation and the right side represents the second turn. The input to BERT is the query concatenated with each one of the passages at a time, using the structure [CLS] q [SEP] p.

- **Pooled output of the [CLS] token** - This embedding corresponds to [CLS] token of the last hidden state further processed by a linear layer and a *tanh* activation function. This is also a usual way of using BERT, but it works best when the model is fine-tuned on the task at hand [37].

After this, we pass the chosen BERT [CLS] embedding ($emb_{[CLS]}$) through an RNN such as, an LSTM [20] or GRU [2] to obtain the representation of the current query-passage pair in turn i given the context (out_i), and the hidden state of the conversation (h_i), which will be passed to the subsequent turns:

$$out_i,\ h_i = RNN(emb_{[CLS]},\ h_{i-1}), \tag{5.3}$$

The representation of the query-passage pair (out_i) is subsequently used as input to a single-layer feed-forward neural network (FFNN) followed by a *softmax* function to calculate the probability of the passage being relevant:

$$P(p|q) = softmax(\ FFNN(\ out_i)\ ), \tag{5.4}$$

In the second turn, we apply the same procedure, but we also have access to the hidden state generated by the RNN in the previous turn (h_{i-1}). We use this hidden state to maintain the conversational context by passing the hidden state generated by the first passage in the rank to the next turn. The rationale for this is that the most relevant passage, and the one that the user first sees, is the one that appears in the first position of the ranked list of results. In an interactive setting, we could replace this hidden state with the passage the user clicked, using the user's feedback instead of the model's previous output.

To train this model, we need conversations in the form of conversational queries about a specific topic with corresponding passages labeled as relevant and non-relevant to the query. The input to the model in each turn will then be a query-passage pair, and the output is the probability of the passage being relevant to the query. We then proceed to train the model using the standard cross-entropy loss as in [37]. After training, at re-ranking time, we calculate for a query, and each of the passages retrieved by the retrieval model, the probability of the passage being relevant to the query. In the end, we re-rank the passages according to that probability. More details about the training and evaluation process of the model will be discussed in Section 6.4.3.1.

5.3.2 ConvBERT MemNet

As explained in Section 2.6.2, the memory network architecture can be used to mitigate the problems that arise from the use of RNNs, such as vanishing gradient problems [54, 59]. More important than that and specific to our task, there is also the problem that the previous queries and passages are always encoded in the RNN's hidden state, so previous queries and passages unrelated to the current query can add noise to the conversational context representation. This problem is addressed by the memory networks because it explicitly stores the embeddings of each turn separately.

Our approach with ConvBERT MemNet (Conversational BERT using Memory Networks) uses a single-layer, single-hop memory network [54] to model the conversational context. Ways of using memory networks for question-answering have proven to be successful for example in *Visual Dialog* [11], which uses both images and text, and in *Wizard of Wikipedia* [14]. In *Wizard of Wikipedia* in particular, memory networks are used with BERT embeddings as input to simulate a conversation.

Figure 5.4 shows the implemented architecture in the fourth turn of the conversation, storing the top query-passage embedding in each turn (3 embeddings stored memories).

The first step, and necessary in all turns of the conversation, is to use BERT to encode the query with each of the retrieved passages following the same input structure as ConvBERT RNN represented in equation 5.1.

After getting the [CLS] embeddings generated by BERT ($emb_{[CLS]}$), if we are in the first turn of the conversation, i.e., the memory is empty, we bypass the memory access and pass the embeddings directly to the feed-forward neural network followed by a *softmax* function to obtain the probability of the passage being relevant. This is the same expression used in [37] because we are not in a conversational scenario:

$$P(p|q) = softmax(\ FFNN(emb_{[CLS]})\ ). \tag{5.5}$$

If we are not in the first turn (the memory is not empty), we follow the approach of Figure 5.4. First, we perform the inner product between the current query-passage embedding, $emb_{[CLS]}$, and each of the memories (previous turns embeddings), m_i, and then calculate the *softmax* of these inner products to get the attention weights, a_i, for each

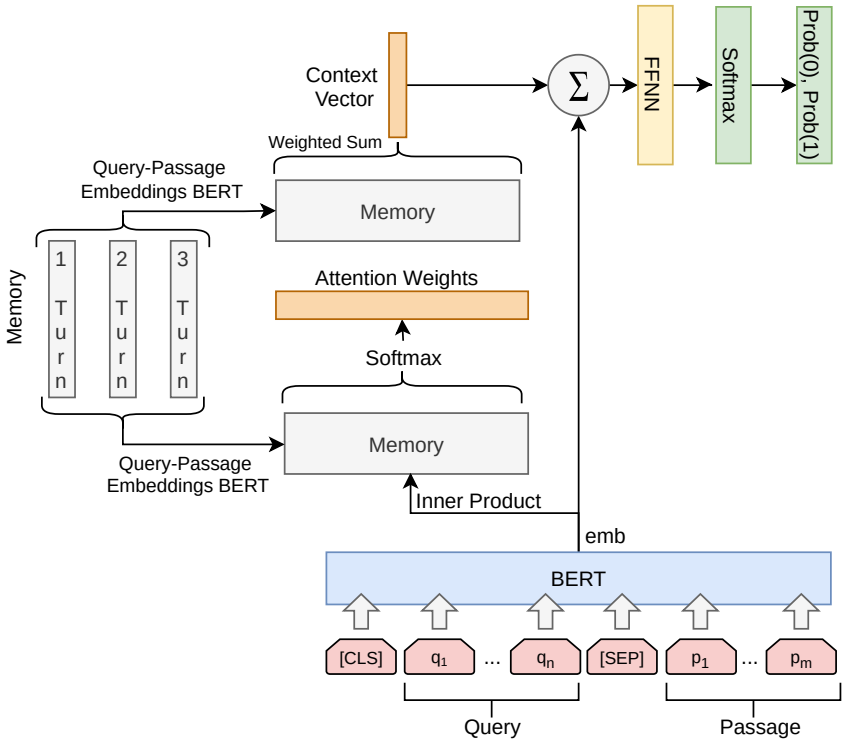


Figure 5.4: ConvBERT MemNet architecture in the fourth turn of the conversation, storing the top query-passage embedding in each turn (3 previous turns in memory). The input to BERT is the query concatenated with each one of the passages at a time, using the structure [CLS] q [SEP] p.

memory:

$$a_i = softmax(emb_{[CLS]}\, m_i^T), \tag{5.6}$$

With these attention weights, we calculate a weighted sum with the memories, which returns the context vector c:

$$c = \sum_i a_i m_i, \tag{5.7}$$

Following that, we perform the sum of the weighted memories with the current embedding, and pass this through a linear layer and a *softmax* function to get the probability of the passage being relevant:

$$P(p|q) = softmax(\ FFNN(c + emb_{[CLS]})\ ). \tag{5.8}$$

In this work, we store as memories the top passage given by the re-ranking algorithm in each turn. This is the same rationale used in ConvBERT RNN, but in the ConvBERT MemNet, the use of a dedicated memory provides a more flexible way of using the context. For example, instead of only using the embedding of the top passage retrieved as a memory, we can use several and adapt the weights given to each one depending on their rank. Also, in an interactive scenario, we could use as memories all of the passages previously clicked by the user, maintaining a complete history of the user's interactions with the system.

With this architecture, we mitigate the problems of the RNNs, while also having the advantage of keeping all of the previous sentence embeddings in memory in their original form, not losing any information. These embeddings can then be "used" or "ignored", depending on the attention weights given to each memory, so we believe that this architecture may be more robust to topic shifts.

The training procedure of ConvBERT MemNet is the same as in ConvBERT RNN, using conversations with labeled query-passage pairs. At re-ranking time, we also apply the same process as in ConvBERT RNN, ranking the passages using the score given by the model. As stated before, further details about training and evaluation will be provided in Section 6.4.3.1.

5.4 Summary

The pipeline of the conversational search system developed is shown in Figure 5.5. In this chapter, we focused on re-ranking models, and in specific, we presented different architectures that take advantage of the pre-trained language model BERT [12].

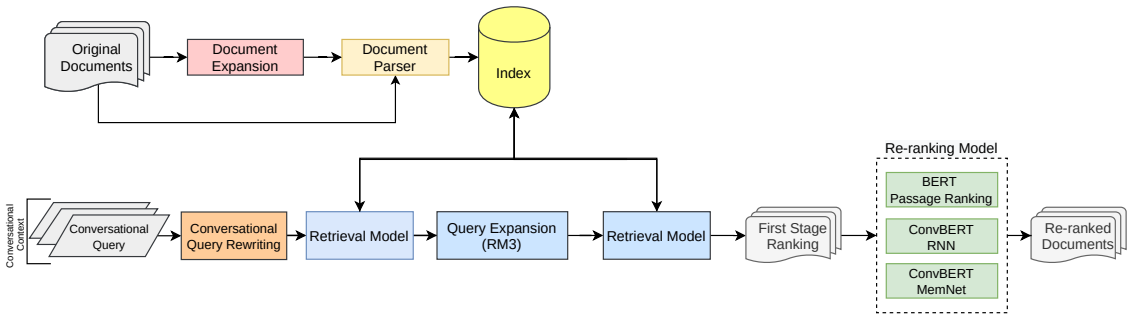


Figure 5.5: Indexing and document/passage expansion approaches (top half). Retrieval, query rewriting and query expansion and re-ranking (bottom half).

As explained before, the re-ranking models developed aim to "push" to the top the passages that are more relevant to a query, basing this analysis on the embeddings generated by BERT, instead of the simpler functions based on term matching and term frequency used by typical retrieval models. It is also important to recall that these models are complex and computationally expensive, being this the reason why they are only applied to a subset of passages retrieved from the index and not the full collection.

We followed the work done by Nogueira and Cho [37] and use the same model and input format to generate the embeddings. Despite the good results achieved by this model in the re-ranking task, it does not fully handle the challenges of conversational search scenarios because of the limitation of the input structure and the topic shifts that occur during a conversation. So, to overcome these limitations, we implemented extensions to BERT that use the generated embeddings to model the conversational context. Specifically, we explored RNN, and Memory Networks approaches:

- ConvBERT RNN uses a hierarchical structure with BERT generating the sentence embeddings and an RNN maintaining the conversational context that is propagated to subsequent turns.
- ConvBERT MemNet uses a memory to store previous turn embeddings, being able to access them in subsequent turns. This model, in contrast with ConvBERT RNN, can store each of the previous turns separately since the attention weights are calculated for each memory, making this a more flexible way of representing context.

With these extensions, we attempt to not only place the more relevant passages at the top but also place at the top the passages that better reflect the context of the conversation up to that point.

In the next chapter, we show the extensive evaluation conducted to assess the performance of the various methods described in this chapter and in Chapters 3 and 4.

Chapter 6

Evaluation

In this chapter, we present and discuss the results obtained using the various approaches described in the previous chapters. We start by defining the task and the dataset used, as well as present an analysis of the characteristics of this dataset. Then we display and discuss the results obtained for the first-stage retrieval, query rewriting, and re-ranking steps, using the different techniques. After this, we analyze the results of the different techniques, taking into account the specific characteristics of conversational search. To finalize, we compare our results to the baselines from 2019 and present a summary of the key findings.

6.1 TREC CAsT Dataset

In order to evaluate the performance of our system, we need a dataset that is suited to our task. In information retrieval, typical datasets include a list of queries and a list of corresponding documents that are classified as relevant or non-relevant [13, 36], normally in a non-exhaustive way (results pooling), i.e, not all documents were judged/annotated. Conversational search, on the other hand, is a recent topic in information retrieval, so the creation and amount of data dedicated to this task is still a matter of research.

To evaluate our system, we used the **TREC CAsT (Conversational Assistance Track) 2019 dataset** [9], that to the best of our knowledge is the only dedicated conversational search dataset created. TREC CAsT [10] is a track that begun in 2019 and in its first year, the aim was to create a benchmark for conversational search, focusing on candidate response (passage) retrieval. Recalling, the conversational search task is defined as: given a sequence of natural language conversational turns for a topic T, with utterances (u), for each $T = u_1,...u_i,...u_n$, the task is to find relevance passages P_i for each turn u_i, satisfying the user's information need for that turn with that context. An example of a topic is

provided in table 6.1, where it is possible to see the characteristics of conversational search, in particular **the need for context from previous turns to answer the current utterance**.

Table 6.1: Example of a topic from TREC CAsT 2019 [10].

Topic Number 31	
Description: A person is trying to compare and contrast types of cancer in the throat, esophagus, and lungs.	
Turn	**Utterances**
1	What is throat cancer?
2	Is it treatable?
3	Tell me about lung cancer.
4	What are its symptoms?
5	Can it spread to the throat?
6	What causes throat cancer?
7	What is the first sign of it?
8	Is it the same as esophageal cancer?

The complete passage collection is comprised of MS MARCO [36], TREC CAR [13], and WaPo [22] datasets, amounting to a total number of passages close to 47 million. Due to an error in the deduplication algorithm of WaPo, the final assessments are only restricted to MS MARCO and TREC CAR.

6.1.1 Conversation Topics and Relevance Judgments

The topics for TREC CAsT were created using a combination of TREC topics, MS MARCO conversational sessions [36], and the researchers' interests in order to exhibit complexity, diversity, and answerability. The conversational turns were manually created, generally starting with an introduction to the topic and then progressing to a more exploratory seeking of information. The questions were also created considering only previous questions and not system responses. These created conversations mimic the characteristics of a typical conversation, introducing coreference, omission, and comparisons between subtopics.

Researchers created 30 training topics and 50 evaluation topics, each with about 10 turns. Also provided are the queries rewritten (resolved) to include all of the information needed for the current turn, basically turning the task into a non-conversational task. These resolved queries can be used as an upper bound in terms of results when compared to the original (raw) conversational queries.

In total, 5 of the 30 training topics include a relevance assessment using a three-point relevance scale (0-not relevant, 1-relevant, 2-highly relevant), and since part of the data is from other datasets some have relevance labels associated, but for questions in a non-conversational form.

In terms of the evaluation data, 20 conversational topics were labeled until turn depth 8 on average. The assessment process for the evaluation set used a TREC style pooling, where it is created an assessment pool using the two runs marked with the highest priority from each participant and judged until pool depth 10 by NIST assessors. The relevance labels are a graded relevance that ranges from 0 (not relevant) to 4 (highly relevant). The exact meaning of each grade is provided in Figure 6.1.

(4) Fully Meets - The passage is the "perfect" single response for the turn. It focuses on the subject and contains little extra information.

(3) Highly meets - The passage answers the utterance and is focused on the answer (i.e., it is a satisfactory answer for a voice assistant). It may contain limited extraneous information.

(2) Moderately meets - The passage answers the utterance, but is focused on something related (i.e., it might initially be clear why a voice assistant picked this passage).

(1) Slightly meets - The passage includes some information about the turn but does not directly answer it. Users will find some useful information in the passage that may lead to the correct answer, perhaps after additional rounds of conversation (better than nothing).

(0) Fails to meet - The passage is not relevant to the question or is unrelated to the target query.

Figure 6.1: Relevance values scale according to TREC CAsT [10].

6.1.2 Evaluation Metrics

Regarding the evaluation metrics used in TREC CAsT, the researchers focus primarily on metrics that evaluate the earlier positions since, in a conversational scenario, one of the main interests is to provide the best answers first. With this in mind, the official metrics used to evaluate the participant systems were the nDCG@3 (normalized Discounted Cumulative Gain at 3), MAP (Mean Average Precision), and MRR (Mean Reciprocal Rank).

6.1.3 Dataset Analysis

To better understand the characteristics of conversational search and the differences between the training set and the evaluation set, we performed the analysis presented in table 6.2. The evaluation set contains the original conversational queries and the coreference resolved queries, which are the same but written in a non-conversational format.

From the analysis, we can see that there are 30 training topics and 50 evaluation topics, amounting to a total of 269 and 479 turns respectively. In the training set, 13 conversations (43.33%) out of the possible 30 were judged, and 20 (40.50%) out of the possible 50 were judged in the evaluation set. Concerning the turns in the training set, 120 (44.6%) out of the 269 were judged, and in the evaluation set 194 (40.50%) out of

Table 6.2: TREC CAsT dataset statistical analysis.

Parameter	Train Set	Evaluation Set		Total
		Conversational	Resolved	
# conversations	30	50	50	80
# judged conversations	13 (43.33%)	20 (40%)	20 (40%)	33 (41.25%)
# turns	269	479	479	748
# judged turns	120 (44.6%)	194 (40.50%)	194 (40.50%)	314 (41.98%)
# turns where context is needed	79 (65.83%)	125 (64.43%)	0 (0%)	204 (64.96%)
Avg. # turns	8.96 ± 1.45	9.58 ± 1.20	9.58 ± 1.20	9.35 ± 1.32
Avg. # terms per query	7.33 ± 2.05	7.14 ± 2.01	8.68 ± 2.47	7.21 ± 2.02
Avg. # judged docs per query*	19.99	151.28	151.28	101.11
Avg. # relevant docs per query*	5.33	41.85	41.85	27.89

*considering only judged turns

479. The average number of turns is close to 9 in the training set and close to 10 in the evaluation set.

It is also important to note that the training and evaluation sets were judged differently, which justifies the difference between the average number of judged docs per query, and the average number of relevant docs per query is so high between the two sets.

Another important aspect that characterizes conversational search is the smaller size of the queries, and this is evidenced by the data when comparing the non-conversational resolved queries (evaluation set resolved), with 8.68 terms per query, with the queries both in the training and in the conversational evaluation set with 7.33 and 7.14 terms per query respectively.

Conversational question classification. We also evaluated all of the judged queries in both sets and annotated if a question is conversational and only answerable having a notion of the previous context, **showing us that about 65% of queries are conversation**, demonstrating the importance of context in this dataset.

Questions types classification. After the previously described analysis, the focus shifted to classifying the question types. To do this, each of the queries in the evaluation set was categorized by 5 volunteers into one of the following categories: *Describe*, *List*, *Comparison and Connection*, *Yes/No*, or *Compositional* (represented in Figure 6.2). After this, we chose the mode of classification as the final label for the question.

Table 6.3 shows the distribution of queries in the complete evaluation set and by turn depth. The first evident aspect is the high agreement between annotators when classifying the type of query, which shows that this task is not particularly difficult for humans. In terms of the distribution of query types, we see that most queries are of type *Describe* by a large margin, followed by *List*, *Comparison and Connection*, *Yes/No*, and *Compositional*. The distribution of queries by turn depth is also interesting, showing us that although most of the time the queries are of type *Describe*, there is a much wider spread of types of queries after the first turn. Turns 11 and 12 have very few queries so their influence

should not be considered.

The difficulty of answering a question is not always conveyed by the type of question, but we can empirically think that most of the time the *Describe* queries will be easier to answer since concepts are more abundant than the other types. The most difficult queries, in theory, would be the *Comparison and Connection*, and *Compositional*, because in both it is necessary to have a notion of more than one concept in a single query.

Describe - Seeks a brief, general description or summary of the subject. When there is a lot of information about the subject, the highest-quality responses focus on the most well-known aspects of the subject. Example: Tell me about the Bronze Age collapse.

List questions - Seeks a passage that provides a list. When the system must make choices they should be popular, well-known, or with the most important elements first. Example: Who are The Avengers?

Comparison or Connection questions - Seeks passages containing comparisons or connections between the concepts discussed. It should include details of the relationships between the two, which are also the main focus of the topic. Example: How does Netflix compare to Amazon Prime Video?

Yes / No questions - Seeks a passage that answers a question and provides a brief justification or explanation to support the answer. Example: Is the National Coach Museum in Lisbon free?

Compositional questions - This type of question can be considered a combination of two questions. Asking about two aspects of one concept, such as when and why something happened. Example: What is the Galileo system and why is it important?

Figure 6.2: Type of questions according to TREC CAsT [9].

Table 6.3: Type of query distribution in the evaluation set.

	Describe	**Yes/No**	**List**	**Comparison and Connection**	**Compositional**	**# queries**
Total	112 (57.73%)	16 (8.24%)	35 (18.04%)	21 (10.82%)	10 (5.15%)	194
Agreement	92%	94%	83%	94%	82%	-
Type of Query by Turn						
1	15 (75%)	0 (0%)	3 (15%)	2 (10%)	0 (0%)	20
2	11 (55%)	2 (10%)	5 (25%)	1 (5%)	1 (5%)	20
3	13 (65%)	1 (5%)	4 (20%)	1 (5%)	1 (5%)	20
4	9 (45%)	3 (15%)	4 (20%)	1 (5%)	3 (15%)	20
5	8 (40%)	4 (20%)	4 (20%)	2 (10%)	2 (10%)	20
6	11 (55%)	1 (5%)	3 (15%)	3 (15%)	2 (10%)	20
7	14 (70%)	2 (10%)	2 (10%)	2 (10%)	0 (0%)	20
8	10 (50%)	1 (5%)	5 (25%)	4 (20%)	0 (0%)	20
9	10 (58.82%)	1 (5.88%)	3 (17.64%)	3 (17.64%)	0 (0%)	17
10	9 (75%)	1 (8.33%)	1 (8.33%)	0 (0%)	1 (8.33%)	12
11	1 (25%)	0 (0%)	1 (25%)	2 (50%)	0 (0%)	4
12	1 (100%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1

6.2 Indexing and First-Stage Retrieval Evaluation

The first-stage retrieval is the process used to get the first list of ranked passages. In this section, we optimize the indexing strategies, as well as the retrieval models.

Before experimenting with the different aspects of indexing and searching the data, it is important to discuss the information retrieval toolkits that were considered and their advantages and disadvantages. We experimented with two IR toolkits based on Apache Lucene[1]: ElasticSearch[2] and Anserini [64].

- **ElasticSearch** - is an open-source, RESTful, full-text search engine that allows a user to store large amounts of data and search in real-time. ElasticSearch is a commercial system with a large customer base and a well-documented Python library. ElasticSearch provides a great amount of features "out-of-the-box" in terms of analyzers, retrieval models, and search options. One of the disadvantages that we found in ElasticSearch is that it only allows one retrieval model to be in use at a time for each field, so in order to query the same index with different retrieval models, we would need to have the same data indexed multiple times, which in our case due to large scale of the datasets becomes computationally too expensive. Another missing feature from ElasticSearch is the lack of pseudo-relevance feedback models, such as RM3 [32].

- **Anserini** - is an information retrieval toolkit that aims to bridge the gap between research and practice by providing wrappers and extensions on top of Lucene to make it more intuitive. So, Anserini provides a platform that allows researchers to reproduce and test various baselines with minimum effort. From our experience, Anserini stands out by the flexibility that it gives to the user. It implements several ranking algorithms and allows to change between them on the spot. Another important aspect is the implementation of relevance feedback, in particular RM3, a model that has been seen to provide solid results [32]. Anserini also has a Python library to access its Java implementation but it is recent and lacks some features available in the Java version, particularly indexing. The documentation could also be improved upon, being necessary to evaluate the code itself in order to use some of its features.

After comparing both systems, we opted to use Anserini in our experiments, mainly because of interchangeable retrieval models and focus on reproducible results. Another reason for our choice was that it has been gaining traction in the research community, being utilized in various works that we studied such as [37, 39–41].

[1] https://lucene.apache.org
[2] https://www.elastic.co

6.2.1 Document-Passage Parser Results

As explained in Section 3.2.1, the first step is to decide the best approach to index the collection. In our case, since we will apply a re-ranking step to the first-stage retrieval results, when indexing, we aim to get the highest recall possible, collecting the greatest amount of relevant passages to give the re-ranker the greatest chances of success.

Because of the large size of the complete dataset (over 47 million passages), to test the different indexing strategies, we started by only indexing the MS MARCO dataset since this is the most represented in TREC CAsT's relevance judgments. To analyze the results, we used the *Raw* (conversational) queries available for both the training and evaluation sets, and the *Manual* queries, which correspond to the *Raw* queries with coreferences manually resolved by the track organizers, to form non-conversational queries. These *Manual* queries are only available for the evaluation set and can be seen as an upper bound to analyze system performance. For these preliminary experiments, we used the LMD (Language Model Dirichlet) retrieval model with default parameters (μ=1000).

Table 6.4 shows the recall at 1000 obtained by the different indexing strategies discussed in Section 3.2.1, on the index that contains only MS MARCO passages, removing all relevance judgments from TREC CAR and WaPo.

Table 6.4: Recall at 1000 for each method of indexing the MS MARCO dataset in the evaluation set with 1000 passages retrieved considering only relevance judgments from MS MARCO using LMD with μ=1000.

	Index Containing MS MARCO				
Queries	**Original Docs**	**Indri Stopwords**	**Lucene Stopwords**	**KStem**	**KStem and Lucene Stopwords**
Training:					
Raw	0.576	0.535	0.544	0.625	**0.655**
Evaluation:					
Raw	0.493	0.483	0.487	0.539	**0.548**
Manual	0.801	0.791	0.795	0.875	**0.884**

Analyzing table 6.4, we see that both sets are in line with respect to the results obtained using the different indexing strategies.

When comparing the results of the *Raw* and *Manual* queries in the evaluation set, we observe a big difference in recall. This happens because the *Raw* queries lack the resolution of context and so are unable to search for the correct information, demonstrating the importance of using context and the need for the query rewriting techniques developed.

Concerning the indexing strategies, we can see that only removing stop words doesn't improve recall since both stopword removal lists achieved a lower recall when compared to the original docs. The results also show that the less extensive list of stop words from Lucene achieves a higher recall than using Indri's stop words list. Regarding stemming, we see that it significantly increases recall. This happens because of the increased number

of matches between query words and passage words, seeing that both the terms in the query and the passage are stemmed to possibly the same term.

Finalizing the analysis of the indexing strategies, we see that the **best option for our data is using a combination of both stemming and stopword removal using the list of stop words from Lucene**.

6.2.2 Retrieval Models Results

After analyzing the different indexing strategies in the previous section, we now focus on choosing the best retrieval model. For this, we used the complete collection of data (MS MARCO, TREC CAR, and WaPo) and the best method for indexing the data from the previous experiment, that used stemming with KStem and stopword removal using the list of stop words from Lucene.

We tested 3 different algorithms: LMD, LMJM, and BM25. Each one of these algorithms has specific parameters that need to be optimized to achieve the best performance (Section 2.4.2). The list of parameters, search space used, and the parameters that achieved the highest recall in the training set using the *Raw* queries, are presented in table 6.5. **We chose to optimize recall, since, at retrieval time, our objective is to get as many relevant passages as possible**.

Table 6.5: Tunable parameters for BM25, LMD and LMJM, their search spaces, and the parameters that achieved the highest recall in the training set using the *Raw* queries.

Retrieval Model	Parameter	Search Space	Best Parameters
BM25	k1	0.5 - 2.0, step=0.2	1.1
	b	0.1 - 0.9, step=0.2	0.3
LMD	μ	250 - 2500, step=250	1000
LMJM	λ	0.1 - 0.8, step=0.1	0.8

From the experiments conducted, we saw that the parameter that optimizes recall for LMD is μ=1000, for LMJM is λ=0.8, and for BM25 is k1=1.1 and b=0.3. Table 6.6 shows the results of applying these retrieval models in the training and evaluation sets.

Table 6.6: Results for LMD, LMJM, and BM25 with 1000 passages retrieved with the parameters that achieved the highest recall using the *Raw* queries in the training set.

Index containing MS MARCO / TREC CAR / WaPo			
Queries	**LMD μ=1000**	**LMJM λ=0.8**	**BM25 k1=1.1 b=0.3**
Training:			
Raw	**0.565**	0.446	0.533
Evaluation:			
Raw	**0.454**	0.389	0.431
Manual	**0.820**	0.773	0.813

When comparing the three algorithms, **the best-performing one is LMD**, followed by BM25, and finally LMJM with a more significant difference to the other two. This difference may be explained because LMJM generally has better results with longer queries [67], which are less common in this dataset. LMD being the best performing method can also be explained because it is known to work best with shorter queries [67], which in a conversational scenario like this are very common.

The results are also in line with the ones obtained in the previous experiment (table 6.4) using only part of the dataset, showing the ability to generalize results for a bigger collection of passages. The decrease in recall in this experiment in relation to the experiment using only MS MARCO happens because here we are using the entire collection of passages and relevance judgments.

Another important aspect to point out is the impact of the optimization of parameters when compared to the results obtained. We analyzed the results using the complete search space, and found that the best performing methods in terms of recall are all very close to one another, so the selection of these specific parameters does not seem to be of the utmost importance.

With these preliminary retrieval results in the remainder of this chapter we will use LMD with a μ=1000 unless specified.

6.3 Conversational Context as Query Rewriting Evaluation

The task of conversational search is more challenging than simple retrieval, so to maintain conversational context, we developed various query rewriting techniques, basing these on the methods described in Chapter 4. In this section, **we continue to be interested in achieving the highest recall possible while also monitoring all of the other metrics** (MAP, MRR, nDCG, and precision) to find the query rewriting techniques that achieve the best results.

6.3.1 Methods

Table 6.7 shows a summary of the query rewriting techniques developed, as well as the queries provided by the TREC CAsT 2019 organizers.

6.3.1.1 Query Rewriting with Previous Queries

In a conversation that spans various turns, it is bound to have references to previous queries, so incorporating previous turns is one of the simplest ways of providing context as shown in Section 4.2.1.

From the analysis of the TREC CAsT 2019 dataset and the conversations within, we noted that the first query's entities are typically the driving force for the topic, being mentioned in various points. Considering this, we developed the *Pref* (Prefixing) query format. This format prefixes the current query with the first query of the conversation.

Table 6.7: Summary of the query rewriting techniques developed.

Query Rewriting Method	Description
Raw	Raw (original) conversational, queries provided by TREC CAsT.
Manual	Queries with manually resolved coreferences performed by TREC CAsT's organizers (only available for the evaluation set).
Section 6.3.1.1:	
Pref	Original queries from TREC CAsT prefixed with the first query of the conversation.
Title	Original queries from TREC CAsT prefixed with the title of the conversation.
Section 6.3.1.2:	
Coref	Queries resolved using AllenNLP [18] replacing all mentions with the first one.
CorefPronoun	Queries resolved using AllenNLP [18] replacing only pronouns with the first mention.
Section 6.3.1.2:	
Pref+Coref	Queries resolved using AllenNLP [18] replacing all mentions with the first one prefixed with the first query of the conversation.
Pref+CorefPronoun	Queries resolved using AllenNLP [18] replacing only pronouns with the first mention prefixed with the first query of the conversation.
Title+Coref	Queries resolved using AllenNLP [18] replacing all mentions with the first one prefixed with the title of the conversation.
Title+CorefPronoun	Queries resolved using AllenNLP [18] replacing only pronouns with the first mention prefixed with the title of the conversation.
Section 6.3.1.3:	
CorefPronoun+Union	Union of each of the previous queries with the current query (multiple queries per turn), using queries resolved with AllenNLP [18] replacing only pronouns with the first mention.
CorefPronoun+Full-Union	All the text of previous queries and current query concatenated (only one query) using queries resolved with AllenNLP [18] replacing only pronouns with the first mention.
Section 6.3.1.4:	
T5	Queries resolved using a trained T5 [47] model.
Pref+T5	Queries resolved using a trained T5 [47] model prefixed with the first query of the conversation.
CorefPronoun+T5	Queries resolved using AllenNLP [18] replacing only pronouns with the first mention and then apply T5 [47] to the resulting query.
T5+Union	Union of each of the previous queries with the current query (multiple queries per turn), using queries resolved with T5 [47].

This is simple but is able to give context to some queries while adding noise to queries unrelated to the first one.

Also provided in the dataset is a title for each topic. This title gives a general description of the information-seeking intentions during that conversation. We prefixed the title of the conversation to each query, after the first one, forming the query type *Title*. This, in most cases, introduces all of the information needed to decipher the context in the query, but like *Pref*, occasionally adds information unrelated to the current query. The *Title* approach also has the limitation that in a real conversation a title does not exist, so this query formulation is specific to this dataset and serves only as a baseline.

6.3.1.2 Resolving Coreferences

In this dataset, as well as in an actual conversation, the use of mentions to previous entities using pronouns is a recurrent matter. This is a problem for non-conversational retrieval systems since they have no idea of how to resolve the coreferences to search for

the relevant information. To solve this, we used the coreference resolution model from AllenNLP [18] and adapted it to our query rewriting task.

As explained in Section 4.2.2, we applied AllenNLP's model to the concatenation of all the queries and used two replacement methods. The first replaces all of the mentions by the first one *Coref* (Coreference). The second method, *CorefPronoun* (Coreference Pronouns), replaces the mentions by the first one if it is an English Pronoun.

These developed techniques are also compatible with *Pref* and *Title*, forming *Pref-+Coref*, *Pref+CorefPronoun*, *Title+Coref*, and *Title+CorefPronoun*.

6.3.1.3 Union of Previous Queries

When making a union with previous queries, we consider all of the historical queries available. Combining this with the previously mentioned methods we developed *Coref-Pronoun+Full-Union* and *CorefPronoun+Union*. In *CorefPronoun+Full-Union*, we concatenate the current query with all of the previous queries, this way the retrieval model has access to all of the terms, relevant or not, and generates a longer query the longer the conversation. *CorefPronoun-Union*, on the other hand, issues multiple queries by concatenating the current query with each of the previous turns, resulting in various lists of ranked passages. We then fuse these lists, choosing the passages that have the highest retrieval scores, and cut the list to the top-1000 passages without repetition, i.e., when the same passage appears in more than one list, we keep the one that achieved the highest score. To speed the search process, we perform the queries to the index in parallel since the queries are independent of each other.

6.3.1.4 Text-To-Text Transfer Transformer (T5) Model

In this section, we detail the results of fine-tuning the T5 model in the conversational query rewriting task and describe its usage in TREC CAsT 2019, following the explanation in Section 4.2.3.

Model Fine-tuning To have a model capable of performing conversational query rewriting, we followed research conducted by Lin et al. [30] and fine-tuned the model using the CANARD dataset [15]. In particular, we trained the model using the standard maximum likelihood, for 4000 steps, using a maximum input sequence length of 512 tokens, a maximum output sequence length of 64 tokens, a learning rate of 0.0001, batches of 256 sequences, and a beam size of 1 (equivalent to no beam search).

In table 6.8, we show the BLEU-4 scores obtained in CANARD and TREC CAsT 2019 *Manual* queries. The rows *Human* and *Raw* are from [15], the row *T5-BASE* is from [30]. The other rows correspond to our implementation, explained in Section 4.2.3, with the versions of input *Utterance-Separated*, which separates every historical utterance with a special token, and *Turn-Separated*, which separates turns by only inserting the special token every two historical utterances (query-answer pair). Our results are on par

with [30], being lower in the CANARD dataset and higher in TREC CAsT. We believe that this difference is related to the use of different input sequences, as the exact method of constructing the input is not specified in [30].

Table 6.8: BLEU-4 scores for CANARD dev and test sets and for TREC CAsT using the annotated resolved queries (*Manual*).

	CANARD		**TREC CAsT**
	Dev	**Test**	**Manual Queries**
Human [15]	59.92		-
Raw [15]	33.84	47.44	-
T5-BASE [30]	**59.13**	**58.08**	75.07
Our implementations:			
T5-Utterance-Separated	58.29	56.88	79.43
T5-Turn-Separated	58.48	56.84	**79.67**

From the analysis of the BLEU-4 scores and the outputs, we can conclude that **the model is performing both coreference and context resolution**, approximating the queries in a conversational format to usual non-conversational queries, as seen in the examples of table 6.9. Regarding the different input formats, the BLEU-4 scores are similar in both *Utterance-Separated* and *Turn-Separated* input formats, but from now on, when referencing queries rewritten using T5 we consider the *Turn-Separated* input format since it achieved marginally better BLEU-4 scores on the TREC CAsT dataset.

Table 6.9: Example of a conversation from TREC CAsT 2019 training set and the corresponding T5 outputs.

Turn	**Type of Query**	**Conversational Query**
1	Raw	What is a physician's assistant?
	T5 Output	What is a physician's assistant?
2	Raw	What are the educational requirements required to become one?
	T5 Output	What are the educational requirements required to become a physician's assistant?
3	Raw	What does it cost?
	T5 Output	What does becoming a physician's assistant cost?
4	Raw	What's the average starting salary in the UK?
	T5 Output	What's the average starting salary in the UK for a physician's assistant?

Usage in Conversational Search After fine-tuning the T5 model, we can use it in our use case of conversational search to perform query rewriting in context. So after the first turn, we apply the fine-tuned T5 model to the concatenation of all conversational queries, and the model outputs the predicted non-conversational query for the current turn. With this, we created the query rewriting techniques *T5* that only uses the T5 generated queries, *Pref+T5* which concatenates the first query to the predicted queries, and *T5+Union*, which

is the same as *CorefPronoun+Union* but uses T5 predicted queries instead of *CorefPronoun* predicted queries.

Lastly, after analyzing the outputs of T5, we observed that it sometimes mistakes similar entities replacing the mention with the wrong coreference as seen in the example in Section 4.2.3, so we developed *CorefPronoun+T5*. In this method, we first apply the AllenNLP coreference resolution model (replacing only the pronouns), which is an algorithm specifically created to resolve coreferences, and then we apply the T5 model to give context to the current query (implicit context) and to resolve missing coreferences (explicit coreference).

6.3.2 Query Rewriting Results

In table 6.10, we provide an overview of the results obtained with the different query rewriting methods on the evaluation set.

Table 6.10: Summary of the query rewriting techniques results using LMD with μ=1000 on the evaluation set.

Queries	Recall	MAP	MRR@10	nDCG@3	P@3
Raw	0.454	0.141	0.336	0.167	0.262
Manual	0.820	0.327	0.694	0.406	0.590
Pref	0.667	0.227	0.547	0.284	0.432
Title	0.646	0.224	0.599	0.317	0.472
Coref	0.573	0.179	0.445	0.238	0.360
CorefPronoun	0.619	0.203	0.486	0.258	0.380
Pref+Coref	0.670	0.218	0.540	0.281	0.420
Pref+CorefPronoun	0.715	0.246	0.571	0.304	0.462
Title+Coref	0.663	0.220	0.576	0.305	0.459
Title+CorefPronoun	0.703	0.247	**0.626**	**0.338**	**0.505**
CorefPronoun+Full-Union	0.623	0.178	0.528	0.255	0.389
CorefPronoun+Union	**0.737**	0.216	0.557	0.278	0.430
T5	0.697	0.251	0.597	0.322	0.474
Pref+T5	0.679	0.231	0.576	0.304	0.470
CorefPronoun+T5	0.733	0.251	0.596	0.331	0.484
T5+Union	0.689	0.222	0.541	0.281	0.437

The first thing we can observe in table 6.10 is that **all query rewriting methods had the desired effect**, improving on the results of the original conversational queries (*Raw*). With this, the results moved closer to the upper bound corresponding to the manually-rewritten queries (*Manual*), although as we can see, there is still a large difference between them.

Pref and *Title* are very simple approaches but proved to be useful, even getting better results than *Coref* and *CorefPronoun* because the model in these last two is not able to

detect all of the coreferences. As we expected *CorefPronoun* was also better than *Coref* due to only replacing the mention when it is a pronoun and so minimizes mistakes.

The combination *Pref+Coref*, *Pref+CorefPronoun*, *Title+Coref*, and *Title+CorefPronoun* further improved on the results by combining both coreference resolution and concatenation of previous queries. *Title+CorefPronoun* in specific was one of the best methods in most metrics but is not generalizable to a real system.

The techniques based on union provided mixed results. *CorefPronoun+Full-Union* was one of the worst performing methods because of the long queries and excessive noise provided by unrelated terms. On the other hand, ***CorefPronoun+Union* was the best performing technique in terms of recall with 0.737**, because it combines coreference resolution with previous queries, while also eliminating passages where the retrieval score is low thanks to the fusion algorithm. Albeit this improvement in recall, *CorefPronoun+Union* does not surpass *Pref+Coref*, *Pref+CorefPronoun*, *Title+Coref*, and *Title+CorefPronoun* in most of the other metrics, due to the fact that *CorefPronoun+Union* is a recall maximization technique and so metrics that focus on the earlier positions of the rank are not prioritized.

The **T5-based approaches provided a good balance between recall and all of the other metrics**. The combination *Pref+T5* did not improve on *T5* because of the added noise of the concatenated first query. *CorefPronoun+T5* achieved one of the best results in terms of recall (0.733) while also having good results in all of the other metrics, achieving, for example, one of the best results in nDCG@3 with 0.331. This at the expense of applying two models to the query (AllenNLP and T5). Although we obtained good results with *CorefPronoun+Union*, the same was not achieved with *T5+Union* when compared to the base *T5* approach, this may be explained, again by the addition of noise from previous queries on account that *T5* does a better job resolving context than *CorefPronoun*.

After all these considerations, we decided to use in the following sections a selection of the best performing algorithms derived from these results. With this, we choose the *Pref+CorefPronoun*, *CorefPronoun+Union*, *T5*, and *CorefPronoun+T5* query rewriting techniques. We left out *Title+CorefPronoun* despite the good results because we are developing a conversational system that can adapt to a real conversation where a title is not provided.

To complement the analysis of the various query rewriting methods done in this section, in Appendix A, we show the results obtained using the query-expansion method RM3 [32] and the document expansion methods: doc2query [41] and docTTTTTquery [39].

6.4 Conversational Context-Aware Neural Ranking Evaluation

In the previous sections of this chapter, we focused on achieving a good recall without disregarding the other metrics. The reason for this was to have the greatest amount of relevant passages to give the re-ranker better chances of achieving a good result in metrics that evaluate the top positions of the rank. In this section, we present the evaluation of the various architectures described in Chapter 5.

We utilized the well known Transformers Library from Huggingface [60] that aims to combine various transformer-based models in a single library. This library is mainly directed for Pytorch[3], although during the development of this thesis it has been expanded to work with Tensorflow[4] as well. Another important note is to run these models on GPU to speed up prediction. We experimented with both CPU and GPU and obtained speedups of over 25 times when using GPUs.

6.4.1 BERT Model for Passage Re-ranking Results

As presented in Section 5.2, we use a BERT model with the same format as presented by Nogueira and Cho [37]. The Huggingface library provides various BERT models developed for different tasks. In our work, in particular, we used the model architecture "BertForSequenceClassification" which is comprised of a BERT model with a sequence classification layer (feed-forward neural network) on top, which has the same structure as the BERT model defined in [37]. This classification layer is the one responsible for using the embeddings generated by BERT to classify a passage as relevant or non-relevant.

BERT, as stated before, is pre-trained on large amounts of data in order to generate word embeddings conditioned by the context to the left and right of a word. These word embeddings alone without any fine-tuning can be used to solve tasks, but the **real power of BERT is available after performing fine-tuning** on the downstream task, which in our case is passage relevance classification given a query. Due to the large hardware requirements and time needed to fine-tune a model such as BERT, we opted to use the fine-tuned BERT model *nboost*[5]. This model was trained following [37] on exactly the same data, the MS MARCO dataset [36] on 12.8 million query-passage pairs.

The use of a re-ranker is accompanied by a new parameter that we call the **re-ranking threshold**. The re-ranking threshold is the number of passages that are re-ranked with respect to the total number of retrieved passages. This is an important parameter to analyze since a higher threshold will, in theory, provide a better ranking at the expense of a longer retrieval time, especially when using large models. Although, as stated, we expect a better ranking, it is important to verify this in practice so as not to spend time re-ranking more passages without achieving any benefits.

We also experimented with two BERT model sizes: *BASE* and *LARGE*. The differences between BERT *BASE* and *LARGE* are evidenced in table 6.11. It is clear to see that BERT *LARGE*, as the name implies, is larger, having more than three times the total number of parameters than the *BASE* version. Usually, a larger model, i.e., a model with more parameters, allows for better sentence and word representations, but it also comes with the tradeoff of an increase in retrieval time and hardware resources.

Table 6.12 shows the results of retrieval with LMD on the complete index (MS MARCO, TREC CAR, and WaPo), and re-ranking the top 10, 100, and 1000 passages, using the

[3] https://pytorch.org/
[4] https://www.tensorflow.org/
[5] https://huggingface.co/nboost

Table 6.11: BERT *BASE* and BERT *LARGE* architecture comparison.

Model	Layers	Hidden Size	Attention Heads	Total Parameters
BERT BASE	12	768	12	110 million
BERT LARGE	24	1024	16	340 million

BERT models *BASE* and *LARGE* fine-tuned on MS MARCO. The underlines indicate the best method in each group of queries, and the bold results indicate the best method overall, excluding manually resolved queries that are always the upper bound of the experiment.

When re-ranking, we use the same queries at retrieval and re-ranking times, except for *CorefPronoun+Union*, because this format issues multiple queries. So, at re-ranking time, we use the queries generated by our fine-tuned T5 model since these achieved good results in our previous experiments in metrics that evaluate the earlier positions (MRR@10, nDCG@3, and P@3). We call this query-rewriting/re-ranking method *CorefPronoun+Union / T5*. As a final note, "(LMD)" in front of a query rewriting method indicates the absence of a re-ranking step (only retrieval is performed).

Table 6.12: Results of retrieval with LMD using a μ=1000 and re-ranking the top 10, 100, and 1000 passages using BERT *BASE* and *LARGE* fine-tuned on MS MARCO.

Index Containing MS MARCO / TREC CAR / WaPo										
			BASE				LARGE			
Retrieval	Threshold	Recall	MAP	MRR10	nDCG3	P@3	MAP	MRR10	nDCG3	P@3
Raw (LMD)	-	0.454	0.141	0.336	0.167	0.262	0.141	0.336	0.167	0.262
Raw	10	0.454	0.147	0.412	0.226	0.330	0.146	0.403	0.223	0.328
Raw	100	0.454	0.167	0.463	0.276	0.382	0.168	0.466	0.282	0.397
Raw	1000	0.454	0.177	0.454	0.277	0.393	0.181	0.456	0.272	0.385
Manual (LMD)	-	0.820	0.327	0.694	0.406	0.590	0.327	0.694	0.406	0.590
Manual	10	0.820	0.343	0.812	0.512	0.686	0.342	0.793	0.512	0.692
Manual	100	0.820	0.372	0.874	0.569	0.726	0.378	0.868	0.582	0.757
Manual	1000	0.820	0.379	0.859	0.570	0.748	0.389	0.857	0.577	0.757
Pref+CorefPronoun (LMD)	-	0.715	0.246	0.571	0.304	0.462	0.246	0.571	0.304	0.462
Pref+CorefPronoun	10	0.715	0.256	0.676	0.392	0.543	0.255	0.656	0.377	0.541
Pref+CorefPronoun	100	0.715	0.282	0.692	0.418	0.565	0.283	0.706	0.429	0.574
Pref+CorefPronoun	1000	0.715	0.277	0.713	0.412	0.553	0.274	0.702	0.427	0.559
CorefPronoun+Union (LMD)	-	0.737	0.216	0.557	0.278	0.430	0.216	0.557	0.278	0.430
CorefPronoun+Union / T5	10	0.737	0.226	0.660	0.384	0.547	0.225	0.650	0.378	0.540
CorefPronoun+Union / T5	100	0.737	0.277	0.773	0.479	0.644	0.279	0.788	0.490	0.644
CorefPronoun+Union / T5	1000	0.737	0.325	0.791	0.502	0.661	**0.332**	**0.799**	**0.509**	**0.674**
T5 (LMD)	-	0.697	0.251	0.597	0.322	0.474	0.251	0.597	0.322	0.474
T5	10	0.697	0.263	0.688	0.409	0.567	0.263	0.674	0.406	0.568
T5	100	0.697	0.297	0.724	0.461	0.611	0.300	0.733	0.472	0.626
T5	1000	0.697	0.303	0.739	0.472	0.630	0.310	0.739	0.475	0.632
CorefPronoun+T5 (LMD)	-	0.733	0.251	0.596	0.331	0.484	0.251	0.596	0.331	0.484
CorefPronoun+T5	10	0.733	0.262	0.701	0.426	0.582	0.261	0.690	0.420	0.576
CorefPronoun+T5	100	0.733	0.291	0.757	0.480	0.622	0.289	0.740	0.470	0.630
CorefPronoun+T5	1000	0.733	0.302	0.754	0.487	0.636	0.305	0.749	0.484	0.649

When evaluating the results of table 6.12, the first thing that becomes evident is that **applying a re-ranker improves all metrics** except for recall because it uses exactly the same set of passages in retrieval and re-ranking. We expected these improvements because of BERT's ability to judge if a passage is actually relevant to a particular query,

not by term matching, but by having an understanding of the text in the query and passage.

Taking a closer look at the results of the various query rewriting methods, we see that they follow a similar trend to the ones obtained in first-stage retrieval (table 6.10). The best performing queries as expected are the *Manual*, being evident the importance of resolving the context and coreferences. The *Raw* queries, although improved, are still far from achieving good results keeping in line with previous results.

Pref+CorefPronoun is the one that has the lowest results when we use a threshold of 1000, showing the limitations of this method in giving context to the queries beyond coreference resolution.

CorefPronoun+Union / T5, which translates to *CorefPronoun+Union* in retrieval and *T5* in re-ranking, can be viewed as the opposite of *Pref+CorefPronoun* in terms of results, being not competitive with smaller thresholds (10, 100), but **achieving the best results in most metrics** (excluding *Manual* queries) when using a threshold of 1000 with an nDCG@3 of 0.509 using BERT *LARGE*. These results are explained because *CorefPronoun+Union* in the first-stage retrieval step returns passages in the top positions that are more varied and can be less important to the current query, so to harness the better recall of this technique it is important that the re-ranker sees a greater number of passages.

T5 is a method that also improves with the re-ranking threshold, achieving competitive results to all the other query rewriting methods in the various re-ranking thresholds. *CorefPronoun+T5* presents the same trend as *T5* but due to the better explicit coreference resolution provided by AllenNLP (*CorefPronoun*) is able to achieve better results than *T5*.

These results also show that **the previous sections of recall maximization are transferable to re-ranking. This also means that we can rely on the re-ranking model to distinguish relevant from non-relevant passages from a list of varied passages**, as shown in *CorefPronoun+Union / T5* queries.

Graphical View of the Re-ranking Threshold. Figures 6.3 and 6.4 show the results achieved using different re-ranking thresholds on the same query rewriting method for BERT *BASE* and *LARGE*, respectively.

We see that an increase in the re-ranking threshold from 10 to 100 brings improvements in all methods and metrics. When comparing the thresholds 100 and 1000, we also observe an increase in most methods and metrics, although the difference is smaller. It is also worth noting that after re-ranking, our rewritten queries get results close to the non-conversational queries (*Manual*) only with retrieval, **demonstrating that our combination of both query rewriting and re-ranking is an effective approach to the conversational search task**.

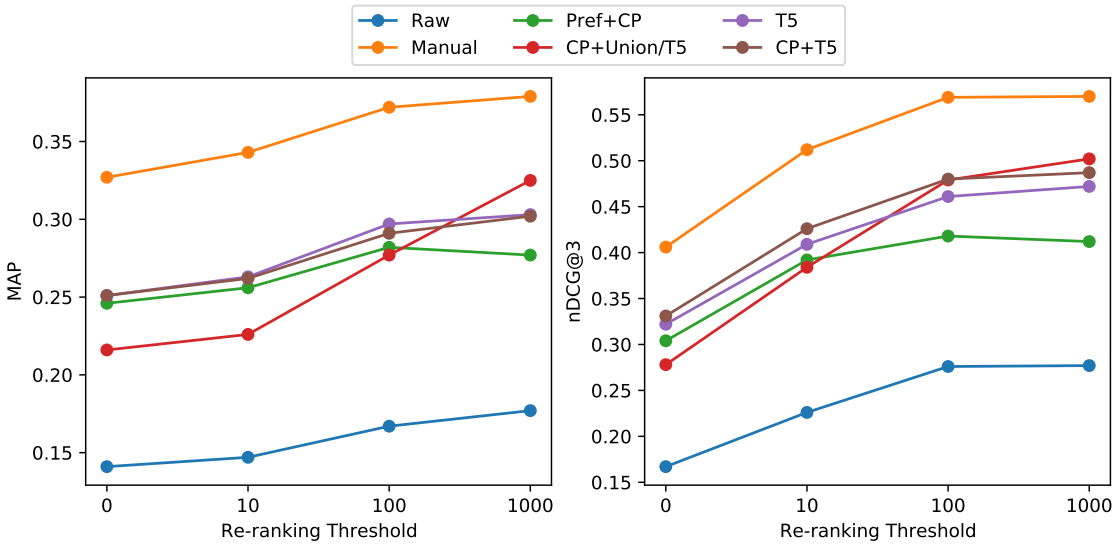


Figure 6.3: Results by re-ranking threshold using various queries, with LMD as retrieval model and BERT *BASE* model fine-tuned on MS MARCO for re-ranking.

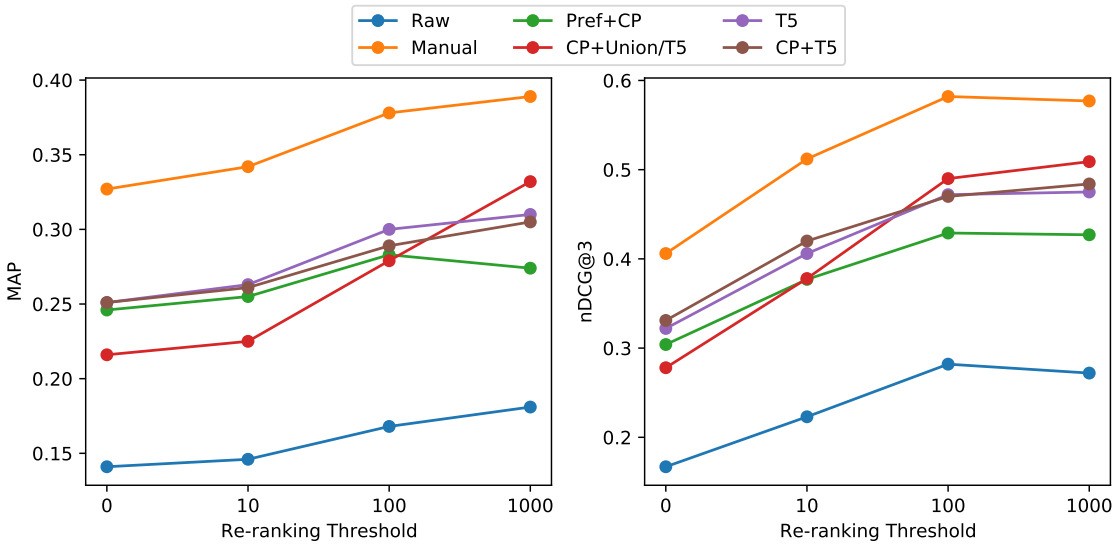


Figure 6.4: Results by re-ranking threshold using various queries, with LMD as retrieval model and BERT LARGE model fine-tuned on MS MARCO for re-ranking.

BERT *BASE* vs *LARGE* in nDCG@3. For an easier comparison between BERT *BASE* and *LARGE* results, in table 6.13, we can see the main evaluation metric of TREC CAsT, nDCG@3, in both models at different re-ranking thresholds. We observe that in terms of the queries, both models have a similar behavior. The results are better with a higher threshold, with the exception of *Pref+CorefPronoun* in both models when the re-ranking threshold is increased from 100 to 1000, and in the *Raw* and *Manual* queries in the BERT *LARGE* model, again when the threshold increases from 100 to 1000.

Table 6.13: nDCG@3 comparison between BERT *BASE* and *LARGE* with different queries and re-ranking thresholds.

Threshold	**10**		**100**		**1000**	
Queries / Model	**BASE**	**LARGE**	**BASE**	**LARGE**	**BASE**	**LARGE**
Raw	0.226	0.223	0.276	0.282	0.277	0.272
Manual	0.512	0.512	0.569	0.582	0.570	0.577
Prefix+CorefPronoun	0.392	0.377	0.418	0.429	0.412	0.427
CorefPronoun+Union / T5	0.384	0.378	0.479	0.490	0.502	**0.509**
T5	0.409	0.406	0.461	0.472	0.472	0.475
CorefPronoun+T5	0.426	0.420	0.480	0.470	0.487	0.484

Comparing the results of the models in each threshold (table 6.13), we see that when re-ranking the top 10 passages, the results are close to each other because the number of passages seen by the models is small. When we look at the results in the top-100, *LARGE* takes the advantage over *BASE* in all queries except for *CorefPronoun+T5*. In the top-1000, we see mixed results but the best performing method, excluding *Manual*, as seen before is *CorefPronoun+Union / T5* in both models. In particular, the BERT *LARGE* model achieved the best results so far with an nDCG@3 of 0.509.

Summing up, the use of a **fine-tuned re-ranker can improve results in both conversational and non-conversational systems**. The choice of re-ranking threshold is also very important, being necessary to understand the subtleties involved. An increase in the re-ranking threshold and model size will have different impacts depending on the query rewriting method chosen, while also increasing the retrieval time. So when designing a system, **if we want the best performance in general we use a larger model and set a larger threshold, if this is not needed and the hardware is not available at a large scale, we can compromise and use a smaller model/threshold**, that, as we saw in table 6.13, still provides good results.

To complement the results of this section, in appendix B, we show the results obtained with the BERT *LARGE* model combined with both query expansion using RM3 [32] and document/passage expansion via docTTTTTquery [39].

6.4.2 The Importance of Fine-tuning

In the previous experiments, we used a fine-tuned BERT models to perform the re-ranking. In this section, we analyze the importance of fine-tuning the model in our task.

Recalling BERT pre-training tasks [12], there is the masked language modeling (MLM) task, where the objective is to reconstruct the original input given corrupted input, and the next sentence prediction task (NSP), where the aim is to classify if the next sentence in the input is, in fact, the next sentence. We can think that the NSP task is relatively close to question-answering since, after a question, we expect to see an answer, or at the very

least, a concept related to that question. So, to analyze the importance of fine-tuning the model in the relevance estimation task, we tested with the original pre-trained weights of the BERT *BASE* model [12] without any particular fine-tuning to re-rank the top 10, 100, and 1000 passages. The results are presented in table 6.14 and a graphical representation can be seen in Figure 6.5.

Table 6.14: Results of retrieval on the evaluation set using LMD with μ=1000 and re-ranking the top 10, 100, and 1000 passages using BERT *BASE* Not fine-tuned using the next sentence prediction (NSP) scores as ranking criterion.

Index Containing MS MARCO / TREC CAR / WaPo						
Queries	**Threshold**	**Recall**	**MAP**	**MRR@10**	**nDCG@3**	**P@3**
Raw (LMD)	-	0.454	0.141	0.336	0.167	0.262
Raw	10	0.454	0.145	0.406	0.198	0.303
Raw	100	0.454	0.129	0.356	0.162	0.258
Raw	1000	0.454	0.092	0.294	0.114	0.189
Manual (Raw)	-	0.820	0.327	0.694	0.406	0.590
Manual	10	0.820	0.332	0.752	0.414	0.620
Manual	100	0.820	0.271	0.605	0.294	0.466
Manual	1000	0.820	0.180	0.485	0.205	0.328
Pref+CorefPronoun (LMD)	-	0.715	0.246	0.571	0.304	0.462
Pref+CorefPronoun	10	0.715	0.247	0.609	0.305	0.474
Pref+CorefPronoun	100	0.715	0.219	0.561	0.266	0.422
Pref+CorefPronoun	1000	0.715	0.149	0.406	0.169	0.266
CorefPronoun+Union (LMD)	-	0.737	0.216	0.557	0.278	0.430
CorefPronoun+Union / T5	10	0.737	0.219	0.591	0.299	0.459
CorefPronoun+Union / T5	100	0.737	0.214	0.640	0.306	0.464
CorefPronoun+Union / T5	1000	0.737	0.171	0.507	0.221	0.341
T5 (LMD)	-	0.697	0.251	0.597	0.322	0.474
T5	10	0.697	**0.256**	0.630	0.329	0.495
T5	100	0.697	0.218	0.539	0.259	0.403
T5	1000	0.697	0.147	0.438	0.177	0.276
CorefPronoun+T5 (LMD)	-	0.733	0.251	0.596	0.331	0.484
CorefPronoun+T5	10	0.733	**0.256**	**0.663**	**0.355**	**0.520**
CorefPronoun+T5	100	0.733	0.222	0.569	0.216	0.422
CorefPronoun+T5	1000	0.733	0.151	0.449	0.186	0.297

Comparing the results of re-ranking the top 10 passages with BERT, with the ones obtained in original retrieval, we observe a marginal increase in most metrics in most methods, although the results are lower than the fine-tuned model (table 6.12). When comparing the results using re-ranking thresholds of 100 and 1000 is when we see a substantial difference between first-stage retrieval (LMD), BERT not fine-tuned, and BERT fine-tuned. **Using BERT not fine-tuned, the results decrease significantly with a higher re-ranking threshold**, as it can be seen in Figure 6.5, being this is the opposite of what happens with a fine-tuned model. Adding to this, the results are much lower than the ones obtained via simple retrieval (LMD). For example, the best performing queries without re-ranking in terms of nDCG@3 was *CorefPronoun+T5* with 0.331, but this value

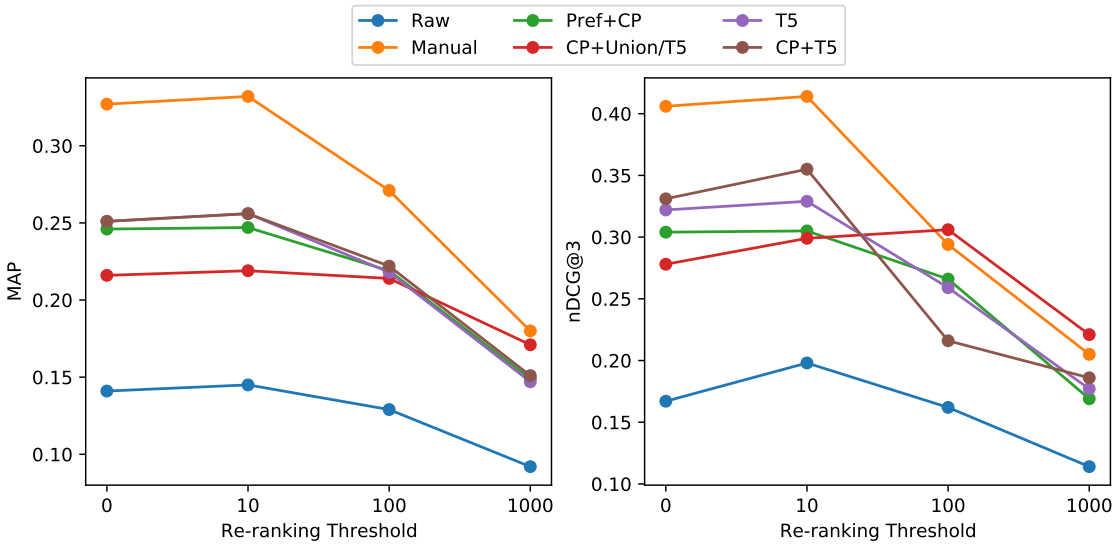


Figure 6.5: Results by re-ranking threshold using various queries, with LMD as retrieval model and BERT *BASE* model not fine-tuned using the next sentence prediction (NSP) scores for re-ranking.

decreases with re-ranking thresholds of 100 and 1000 to 0.216 and 0.186 respectively.

We attribute the decrease in performance with the increase in re-ranking threshold to the difference between the tasks of next sentence prediction (NSP) and relevance estimation. The next sentence prediction task is a much "looser" concept, with the sharing of words between query and passage a criterion that is almost enough to be a possible next sentence "in the eyes" of the model, and so the scores of most query-passage pairs were pushed closer to 1 (maximum value of relevance). This is in direct contrast to what happens with a fine-tuned model, where some passages are given scores close to 0 (minimum value of relevance).

Summarizing the results, although it can be useful in some situations to use the embeddings of the original BERT model, in re-ranking as the results show, **it is crucial to fine-tune the model** because it brings better results without any increase in retrieval time.

6.4.3 Conversational BERT for Passage Re-ranking

In a conversational scenario, we want to keep track of the context and push to the top the most relevant passages to that particular context. The results of the re-ranking model shown before only consider the current query and one passage at a time to decide the relevance of that passage. In this section, we show the results obtained with the re-ranking architectures developed that use RNNs and Memory Networks to evaluate the relevance of a passage given a query considering the context stored in their respective components (Section 5.3).

6.4.3.1 ConvBERT Training

We start by defining the training task as a **binary relevance classification task** following [37] using the cross-entropy loss. This task is used to train the model to classify passages as relevant or non-relevant given the current query and the conversational context (i.e., previous queries and passages).

Remembering the architecture, *ConvBERT* is composed of an *RNN* or *Memory Network* on top of a BERT model. In particular, we use the same fine-tuned BERT *BASE* model detailed in Section 6.4.1, and train only the top part (RNN or Memory Network), since the model already does a good job of detecting relevance in non-conversational queries, and the amounts of data needed to train the model from scratch in the conversational relevance classification task are not available.

Regarding the data, we used the TREC CAsT training set, that as we saw in Section 6.1.3, has about 20 different passages that are classified with 0, 1, or 2 (not relevant, relevant, very relevant) for each of the 120 classified turns. To use this data in a binary classification task we simplified the annotations to only 0 and 1, by considering the annotation with 1 and 2 as 1, creating a dataset where 80% of the passages are not relevant for the queries (have label 0). After this pre-processing, for each topic, we create approximately X conversations by randomly sampling without replacement an annotated passage for each turn (query), where X is the number of judged passages in the first turn. This approach creates 259 conversations for training with an average of 10 turns. With this set of conversations, we considered a batch to be a full conversation topic to guarantee that the model sees a full conversation in succession, instead of parts of different conversations at each gradient calculation step.

To validate our models and choose the parameters, we used a train-validation split of 75%, 25%, using 5-fold cross-validation over the full training set. Since we are working with conversational topics and turns, when splitting the dataset, we enforce that the same topics only appears in one of the sets. We then chose the parameters that achieved the highest average validation F1 score considering all folds.

Regarding the input to the models, we experimented with: the pooled output of the [CLS] token, and the [CLS] token of the last hidden state of BERT, as explained in Section 5.3.1, batch sizes of [1, 2, 4] conversations (each conversation has approximately 10 query-passage pairs), and learning rates [0.01, 0.001, 0.0001] using the Adam optimizer [25].

For *ConvBERT RNN*, we experimented with the RNN types: *LSTM*, *Bidirectional-LSTM* [20], *GRU*, and *Bidirectional-GRU* [2]. In the case of the bidirectional RNNs, we concatenate the embeddings given by each direction together [42, 52]. To analyze the importance of context given by the RNNs and Memory networks, we also created a linear model that applies a one-layer feed-forward neural network on top of the BERT embeddings, which we call *Linear* that is trained on the same data (CAsT 2019).

The best parameters for the different architectures are presented in table 6.15. It is particularly interesting to see that the strategy that uses the **Last Hidden State of the [CLS] token always achieved better results**. We can attribute this to the Pooled Output of the [CLS] token being too close to the final task due to being further processed by another linear layer before being passed to the *ConvBERT* architectures.

Table 6.15: Parameters that optimized F1 score in the validation set for the *ConvBERT* architectures in the binary conversational relevance classification task.

Architecture	Pooling Strategy	Batch Size (# conversations)	Learning Rate
Linear	Last hidden [CLS]	1	0.001
LSTM	Last hidden [CLS]	1	0.01
Bi-LSTM	Last hidden [CLS]	1	0.01
GRU	Last hidden [CLS]	1	0.01
Bi-GRU	Last hidden [CLS]	1	0.01
Memory Network	Last hidden [CLS]	1	0.01

6.4.3.2 ConvBERT Results

Model evaluation follows the same procedure as before using TREC CAsT's 2019 evaluation set and metrics. Table 6.16 shows the results of the different *ConvBERT* architectures, as well as the *BASE* and *Linear* models since they use the same BERT model, but without a notion of context. All methods re-rank the top-1000 passages retrieved by LMD.

In table 6.17, we show only the values of nDCG@3, the main metric of TREC CAsT, achieved by the various models and queries for easier comparison.

Comparing the architectures trained on the CAsT dataset that make use of the conversational context (*ConvBERT* architectures) with the simplest approach that uses a single linear layer on top of BERT's output trained on the same data (*Linear*), **we see that the use of context is indeed helpful in all query formulations**. For example, in the *Coref-Pronoun+Union / T5* queries, we see that the nDCG@3 achieved by the *ConvBERT GRU* model is 3.95% higher than the one obtained with the *Linear* model. **This verifies our hypothesis that the use of state, in the form of BERT embeddings stored in the hidden state of the RNN and in the memory of the Memory Network, is important and can be used in a conversational re-ranking scenario.**

When comparing the *ConvBERT RNN* and *ConvBERT MemNet* with each other, we see that in most metrics more than one RNN-based approach surpassed the *MemNet* architecture. Although we cannot be certain, we consider that this may happen because the conversations are not very long, on average 10 turns, and so the RNN's hidden state is able to keep the important information to judge the relevance of a passage.

Analyzing the different RNNs architectures, we see that there is not a significant difference in the best models. Despite this, we generally see that the *GRU* is better than the *LSTM*, and that adding bi-directionality improves only the *LSTM* results.

Table 6.16: Results on the evaluation set using LMD (μ=1000) and re-ranking the top-1000 passages with *ConvBERT RNN*, *MemNet*, and BERT *BASE* fine-tuned (MS MARCO).

Index Containing MS MARCO / TREC CAR / WaPo						
Queries	**Re-ranking Model**	**Recall**	**MAP**	**MRR**	**nDCG@3**	**P@3**
Raw (LMD)	-	0.454	0.141	0.336	0.167	0.262
Raw	Linear	0.454	0.150	0.438	0.255	0.366
Raw	LSTM	0.454	0.170	0.457	0.268	0.372
Raw	Bi-LSTM	0.454	0.169	0.461	0.282	0.385
Raw	GRU	0.454	0.175	0.458	0.281	0.387
Raw	Bi-GRU	0.454	0.176	0.444	0.269	0.389
Raw	MemNet	0.454	0.159	0.435	0.266	0.372
Raw	BASE	0.454	0.177	0.454	0.277	0.393
Manual (LMD)	-	0.820	0.327	0.694	0.406	0.590
Manual	Linear	0.820	0.348	0.814	0.539	0.716
Manual	LSTM	0.820	0.362	0.805	0.526	0.701
Manual	Bi-LSTM	0.820	0.359	0.830	0.553	0.719
Manual	GRU	0.820	0.375	0.852	0.564	0.732
Manual	Bi-GRU	0.820	0.375	0.850	0.537	0.719
Manual	MemNet	0.820	0.347	0.839	0.564	0.725
Manual	BASE	0.820	0.379	0.859	0.570	0.748
Prefix+CorefPronoun (LMD)	-	0.715	0.246	0.571	0.304	0.462
Prefix+CorefPronoun	Linear	0.715	0.246	0.643	0.381	0.514
Prefix+CorefPronoun	LSTM	0.715	0.261	0.676	0.404	0.549
Prefix+CorefPronoun	Bi-LSTM	0.715	0.260	0.697	0.410	0.547
Prefix+CorefPronoun	GRU	0.715	0.275	0.692	0.423	0.570
Prefix+CorefPronoun	Bi-GRU	0.715	0.272	0.682	0.399	0.538
Prefix+CorefPronoun	MemNet	0.715	0.249	0.647	0.400	0.524
Prefix+CorefPronoun	BASE	0.715	0.277	0.713	0.412	0.553
CorefPronoun+Union (LMD)	-	0.737	0.216	0.557	0.278	0.430
CorefPronoun+Union / T5	Linear	0.737	0.285	0.766	0.481	0.649
CorefPronoun+Union / T5	LSTM	0.737	0.302	0.769	0.482	0.647
CorefPronoun+Union / T5	Bi-LSTM	0.737	0.300	0.778	0.497	**0.661**
CorefPronoun+Union / T5	GRU	0.737	0.318	0.775	0.500	**0.661**
CorefPronoun+Union / T5	Bi-GRU	0.737	0.320	0.779	0.487	0.653
CorefPronoun+Union / T5	MemNet	0.737	0.282	0.769	0.494	0.647
CorefPronoun+Union / T5	BASE	0.737	**0.325**	**0.791**	**0.502**	**0.661**
T5 (LMD)	-	0.697	0.251	0.597	0.322	0.474
T5	Linear	0.697	0.275	0.714	0.454	0.612
T5	LSTM	0.697	0.290	0.708	0.440	0.595
T5	Bi-LSTM	0.697	0.287	0.722	0.462	0.609
T5	GRU	0.697	0.298	0.715	0.457	0.607
T5	Bi-GRU	0.697	0.298	0.720	0.445	0.597
T5	MemNet	0.697	0.276	0.720	0.465	0.615
T5	BASE	0.697	0.303	0.739	0.472	0.630
CorefPronoun+T5 (LMD)	-	0.733	0.251	0.596	0.331	0.484
CorefPronoun+T5	Linear	0.733	0.275	0.713	0.457	0.616
CorefPronoun+T5	LSTM	0.733	0.287	0.724	0.449	0.603
CorefPronoun+T5	Bi-LSTM	0.733	0.282	0.733	0.467	0.601
CorefPronoun+T5	GRU	0.733	0.298	0.748	0.479	0.620
CorefPronoun+T5	Bi-GRU	0.733	0.298	0.731	0.456	0.613
CorefPronoun+T5	MemNet	0.733	0.267	0.726	0.468	0.599
CorefPronoun+T5	BASE	0.733	0.302	0.754	0.487	0.636

Table 6.17: nDCG@3 comparison between the *ConvBERT* architectures and BERT *BASE* with different queries.

Queries / Model	LMD	Linear	LSTM	Bi-LSTM	GRU	Bi-GRU	MemNet	BASE
Raw	0.167	0.255	0.268	**0.282**	0.281	0.269	0.266	0.277
Manual	0.406	0.539	0.526	0.553	0.564	0.537	0.564	**0.570**
Prefix+CorefPronoun	0.304	0.381	0.404	0.410	**0.423**	0.399	0.400	0.412
CorefPronoun+Union / T5	0.278	0.481	0.482	0.497	0.500	0.487	0.494	**0.502**
T5	0.322	0.454	0.440	0.462	0.457	0.445	0.465	**0.472**
CorefPronoun+T5	0.331	0.457	0.449	0.467	0.479	0.456	0.468	**0.487**

Comparing all the RNN architectures, **we believe that the best model is obtained when using a *GRU* architecture** because it has a smaller embedding size than the *Bi-LSTM* and has proven to be more effective in situations where less training data is available [2].

Concerning the different query types effect on the *ConvBERT* architectures, the results are in line with the ones achieved by BERT *BASE* and *LARGE* (table 6.12), with *Manual* being the best by a large margin, followed by *CorefPronoun+Union / T5*, *CorefPronoun+T5*, *T5*, and *Prefix+CorefPronoun*.

Comparing the results of the *ConvBERT* architectures to the BERT *BASE* model trained on MS MARCO, we see that the results are close to each other despite the large difference in the amount of training data. In more detail, the *ConvBERT* architectures trained on CAsT used 2590 conversational query-passage pairs, while the linear layer in the BERT *BASE* model trained on MS MARCO saw 12.8 million query-passage pairs [37]. This also shows that the embeddings generated by BERT are of high quality, being able to create a comparable model even with much less data. So a future promising step is to try to use other conversational datasets to obtain more data to train the *ConvBERT* models. However, we need to perform adjustments to these datasets since most of them lack the retrieval aspect of this task, being the objective the extraction of a span of text containing the answer from a single passage.

To summarize, we showed that the **conversational context is helpful**, with context-aware models (*ConvBERT*) surpassing models without context (*Linear*) when trained on the same data, as exemplified by the results achieved with the *CorefPronoun+Union / T5* queries using the *ConvBERT GRU* architecture when compared to the *Linear* architecture. **This verifies our hypothesis that it is possible to create context-aware re-ranking models** even with limited training data.

6.5 Analysis of Conversational Patterns

While in the previous sections, we examined the methods in a general way, in this section, we provide an analysis of the results in the light of the specific characteristics of conversational search.

6.5.1 Per-Turn results analysis

Due to the conversational aspects of the task, it is important to evaluate the performance of the different models with respect to the turn depth.

Figure 6.6 shows the metrics by turn depth obtained with different query rewriting methods using the BERT *LARGE* model trained on MS MARCO as the re-ranker.

As expected, the *Raw* queries achieved good results in the first turn since they are not conversational, but after this results suffer a large decline due to the introduction of coreference, with most metrics proceeding to decrease the longer the conversation. The *Manual* queries, also as expected, are the best-performing ones in most metrics thanks to the manual resolution of coreferences, turning this into a typical information retrieval task, and so are not directly affected by turn depth. *Pref+CorefPronoun* presents a similar behavior to Raw but with much better results, thanks to the coreference resolution model. *T5* and *CorefPronoun+T5* have a similar pattern since the only difference resides in the coreference resolution model used. The best performing algorithm *CorefPronoun+Union* in retrieval with *T5* in re-ranking shows good performance until turn depth 6, which **demonstrates the importance of having a good query rewriting method**.

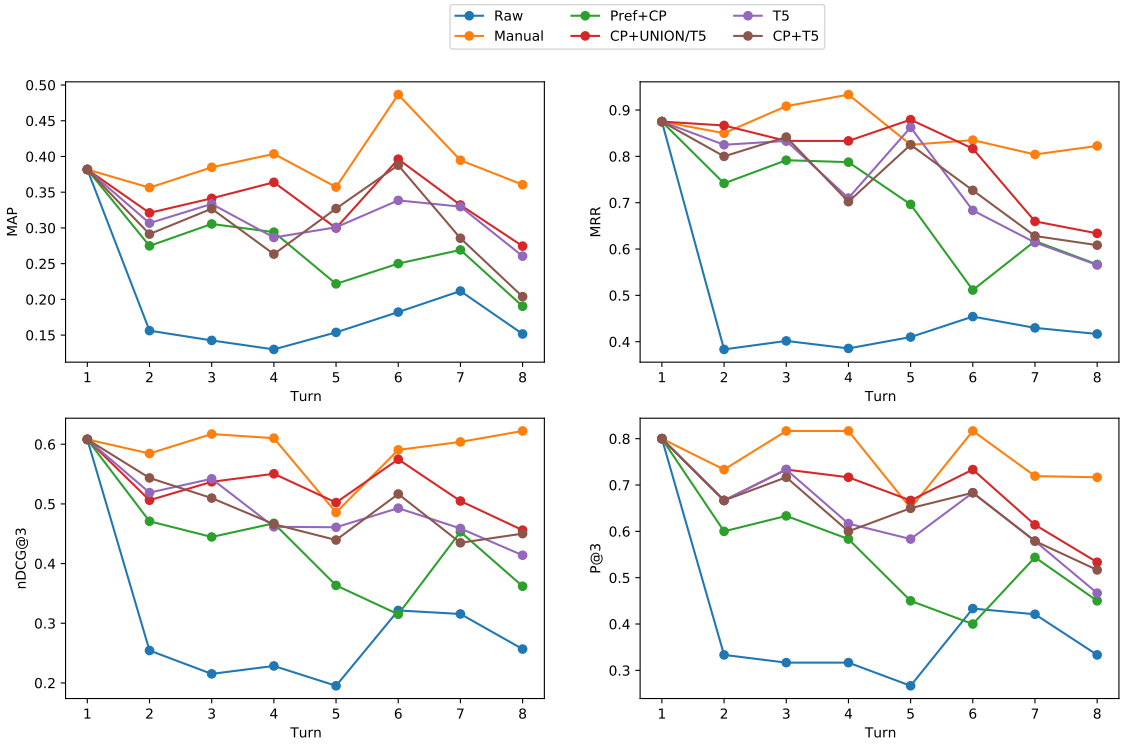


Figure 6.6: Results by turn depth using various query types, using LMD as the retrieval model and BERT *LARGE* re-ranking in the top-1000.

Figure 6.7 shows the metrics by turn depth using the queries *CorefPronoun+Union* in retrieval, and *T5* in re-ranking, using different re-ranking models.

As seen before, **re-ranking has a big influence on performance** with LMD (no re-ranking), achieving lower scores in all metrics by a large margin. The other models present similar behaviors between themselves since all of them share the same BERT model. BERT *LARGE* has better results in most conversational turns with a sharp decline in turn depth 8, for an unknown reason. BERT *BASE* and *ConvBERT GRU* have similar performance across the conversational turns, and *ConvBERT MemNet* generally has the worst results among the re-rankers in most conversational turns, although an increase in performance is observed in turn depth 8 for MRR, nDCG@3, and P@3, **which may be an indication of the advantages of using Memory Networks in longer conversations**.

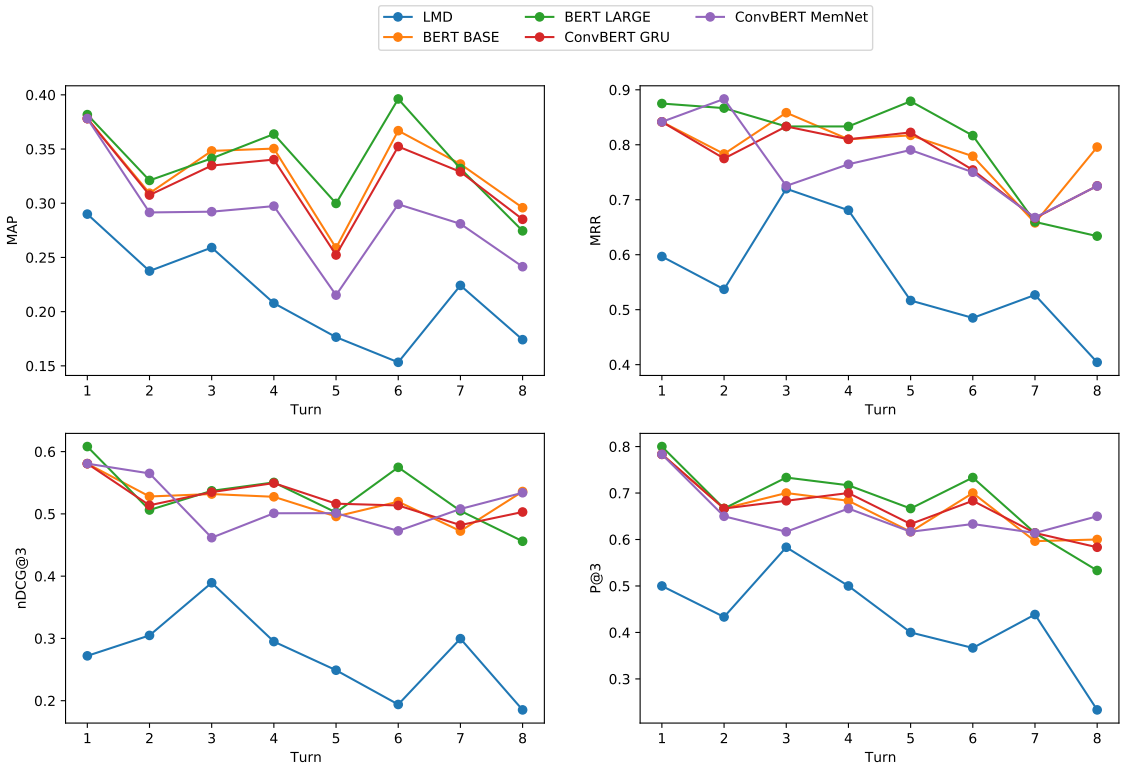


Figure 6.7: Results by turn depth using various re-ranking models, using as query rewriting method *CorefPronoun+Union* in retrieval and *T5* in re-ranking.

6.5.2 Per-Question type analysis

Following the dataset analysis and query classification in Section 6.1.3, we investigate the performance of the different methods given a query type.

Figure 6.8 shows the results of using the BERT *LARGE* re-ranking model with different query rewriting methods by type of query. The query types considered were: *Describe*, *Yes/No*, *List*, *Comparison and Connection*, and *Compositional*, as explained in Figure 6.2. The value inside parentheses indicates the number of turns with that query type.

Unexpectedly, one of the best performing query types is the *Compositional*, which in theory, are the most complex. We attribute this to the shortage of these queries in the dataset and that most of them (80%) are not conversational. *Yes/No* was the query type that achieved the lowest scores, this is interesting since these are theoretically the most basic kind of answers, showing that retrieving just a yes/no type of answer from a dataset containing passages can be a difficult task.

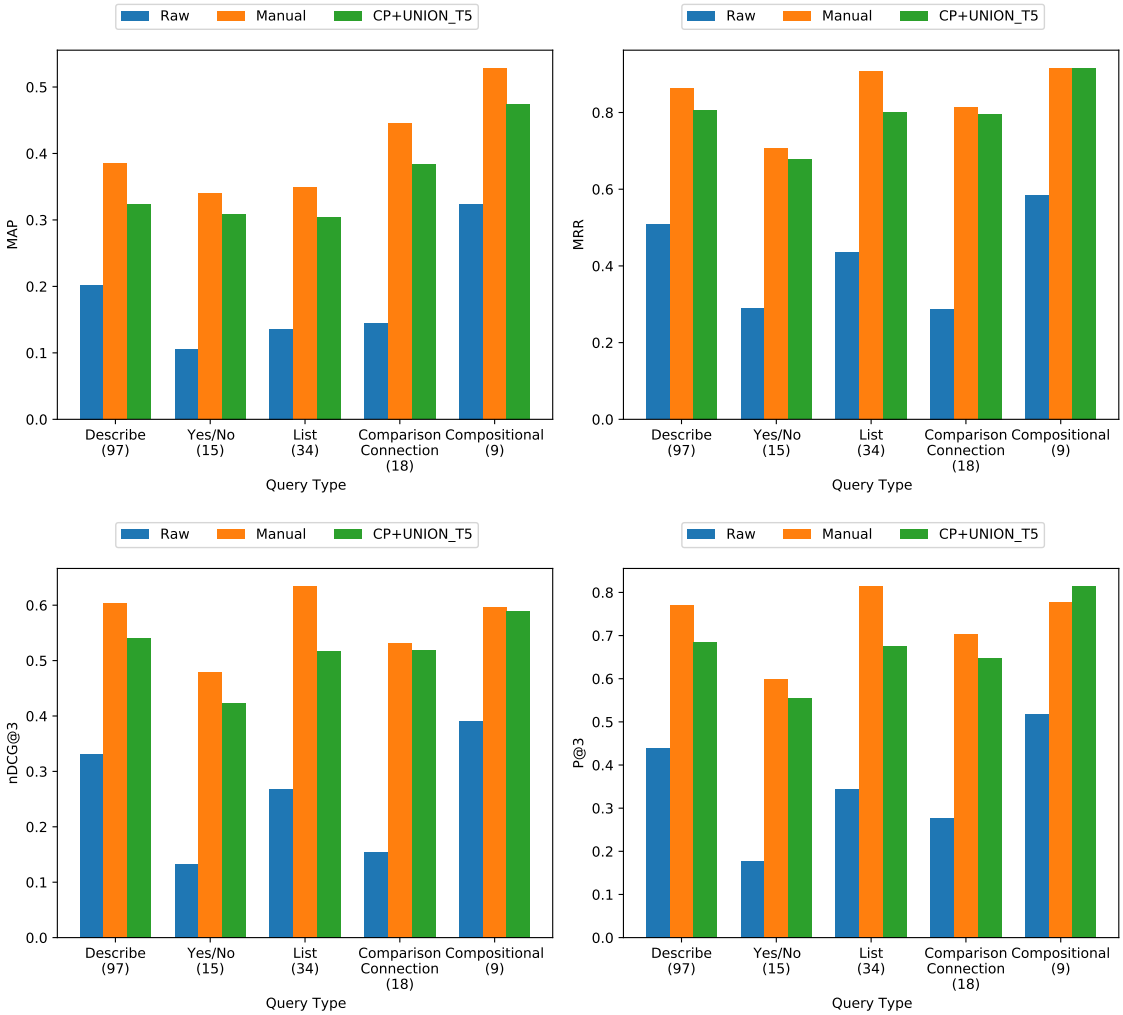


Figure 6.8: Results by query type using various query rewriting methods, using LMD as the retrieval model and BERT *LARGE* re-ranking in the top-1000.

In Figure 6.9, we present the results by query type, using different models with the same query *CorefPronoun+Union* in retrieval and *T5* in re-ranking. The query type with the best performance was *Compositional*, and the worst performance was *Yes/No* for the same reasons enumerated before. In terms of the model's performance per query type, LMD is far from the results of the re-ranking models in all query types. *ConvBERT GRU* achieved results close to the BERT *BASE* model in most query types, except the *List* queries. The context-aware model even achieved better results in the *Compositional* queries in MRR and nDCG@3, in the *Describe* queries in nDCG@3, and in the *Yes/No*

queries in nDCG@3 and P@3, despite using less training data.

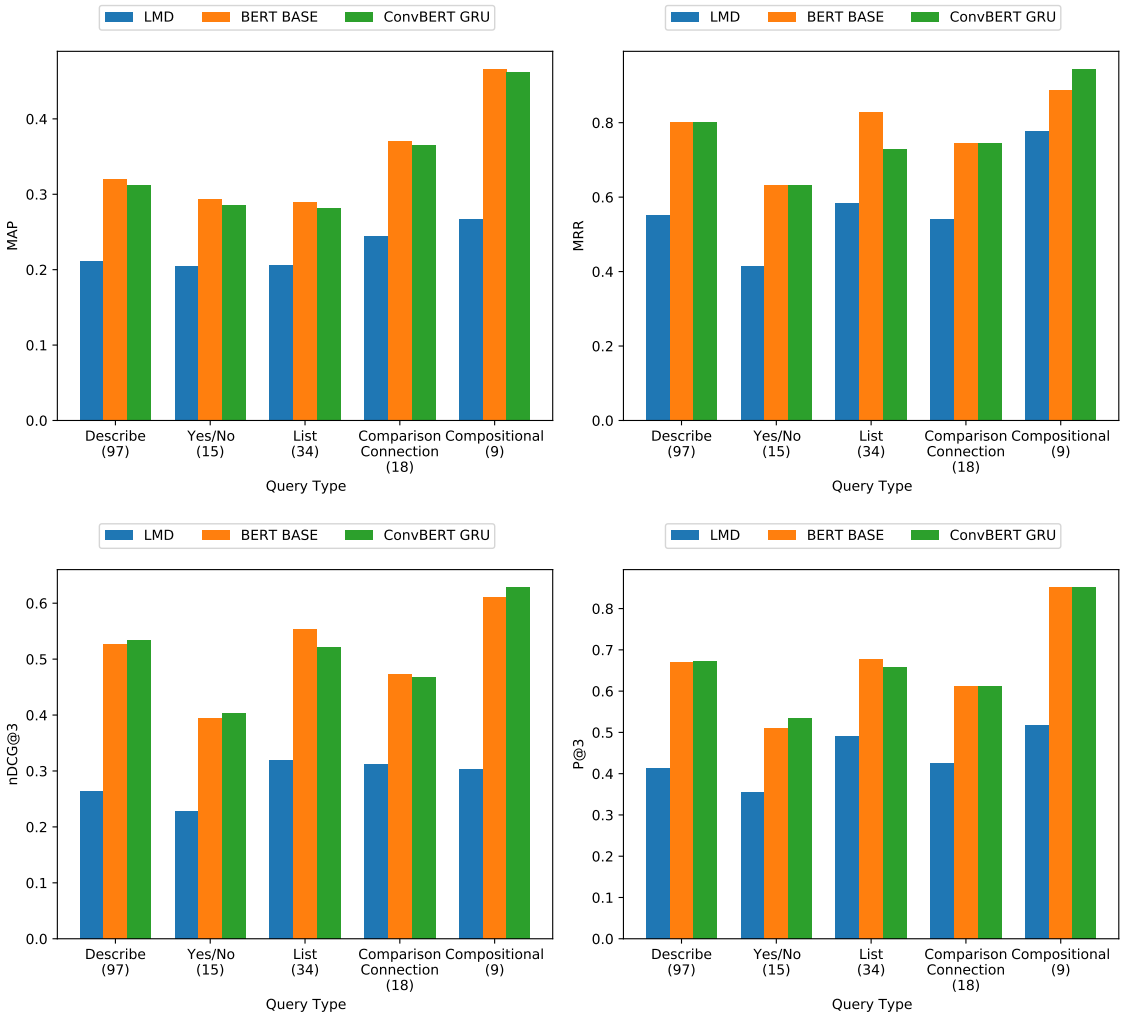


Figure 6.9: Results by query type using various re-ranking models, using as query rewriting method *CorefPronoun+Union* in retrieval and *T5* in re-ranking.

6.6 Comparison to TREC CAsT 2019 Baselines

In this section, we compare the performance of our system to the performance of the baselines submitted to TREC CAsT 2019 [10].

Table 6.18 shows the best models developed in this thesis and compares them to the best models submitted to TREC CAsT 2019 [10]. The first lines represent the performance of the *Raw* conversational queries, with and without BERT *LARGE* re-ranking, and serve as a lower-bound.

With respect to the baselines, we present the results of:

- **clacBase** [4] - uses AllenNLP coreference resolution [27] and a fine-tuned BM25 model with pseudo-relevance feedback.

- **ilps-bert-feat1 [57]** - uses language modeling and expansion with RM3. A BERT model is used to encode the passages and obtain a score. The final score is achieved by linearly combining the BERT score and the unsupervised ranker's score.

- **pgbert [10]** - uses a trained GPT-2 Transformer model to rewrite queries and a BERT-based re-ranking model.

- **HistoricalQE [63]** - uses a query expansion algorithm based on session and query words together with a BERT *LARGE* model for re-ranking. This was the best performing method in TREC CAsT 2019.

The developed methods we present are the ones that achieved the highest nDCG@3 in first-stage retrieval, with and without *T5* query rewriting, which corresponds to LMD with *Pref+CorefPronoun* and *CorefPronoun+T5* queries, and the best models with re-ranking, with and without conversational context, represented by BERT *BASE*, *LARGE*, and the *ConvBERT GRU* architecture. The last lines show the results of the *Manual* rewritten queries (upper-bound) with and without BERT *LARGE* re-ranking.

Table 6.18: Comparison between the developed methods and the TREC CAsT 2019 [10] baselines on the evaluation set.

Queries	Re-ranking Model	Recall	P@3	MAP	MRR	nDCG@3
Conversational Queries						
Raw (LMD)	-	0.454	0.262	0.141	0.342	0.167
Raw	BERT LARGE	0.454	0.385	0.181	0.459	0.272
TREC CAsT 2019 Baselines						
clacBase [4]	-	-	-	0.246	0.640	0.360
ilps-bert-feat1 [57]	BERT LARGE	-	-	0.260	0.614	0.377
pgbert [10]	BERT LARGE	-	-	0.269	0.665	0.413
HistoricalQE [63]	BERT LARGE	-	-	0.267	0.715	0.436
Developed Methods						
Pref+CorefPronoun (LMD)	-	0.715	0.462	0.246	0.578	0.302
CorefPronoun+T5 (LMD)	-	0.733	0.484	0.251	0.602	0.331
CorefPronoun+Union / T5	ConvBERT GRU	0.737	0.661	0.318	0.777	0.500
CorefPronoun+Union / T5	BERT BASE	0.737	0.661	0.325	0.793	0.502
CorefPronoun+Union / T5	BERT LARGE	0.737	0.674	**0.332**	**0.801**	**0.509**
Manual Baselines						
Manual (LMD)	-	0.820	0.590	0.327	0.698	0.406
Manual	BERT LARGE	0.820	0.757	0.389	0.858	0.577

The results **show the need for a query rewriting method** evidenced by the low scores in all metrics achieved by the *Raw* conversational queries. **All the query rewriting methods developed brought an improvement in results**, especially with the introduction of the *T5* query rewriting model, achieving an nDCG@3 of 0.331 with *CorefPronoun+T5*, even without a re-ranking step.

With the previous experiments, **we created strong first-stage retrieval baselines, that with the introduction of a BERT-based re-ranker, were able to surpass the state-of-the-art systems from CAsT 2019**. In particular, our *ConvBERT GRU* model using the queries *CorefPronoun+Union / T5* surpassed state-of-the-art by a large margin, achieving an nDCG@3 of 0.500. Showing that **it is possible to create context-aware re-rankers that achieve good performance even with limited training data**. **Our best results were obtained using the BERT *LARGE* model** and *CorefPronoun+Union / T5* queries, achieving an nDCG@3 of 0.509, 0.073 nDCG@3 points higher than the best model in TREC CAsT 2019, which was HistoricalQE with 0.436.

6.7 Summary

In this chapter, we performed an extensive evaluation of all the components of the conversational system implemented. In Figure 6.10, we can see an overview of the components of the system.

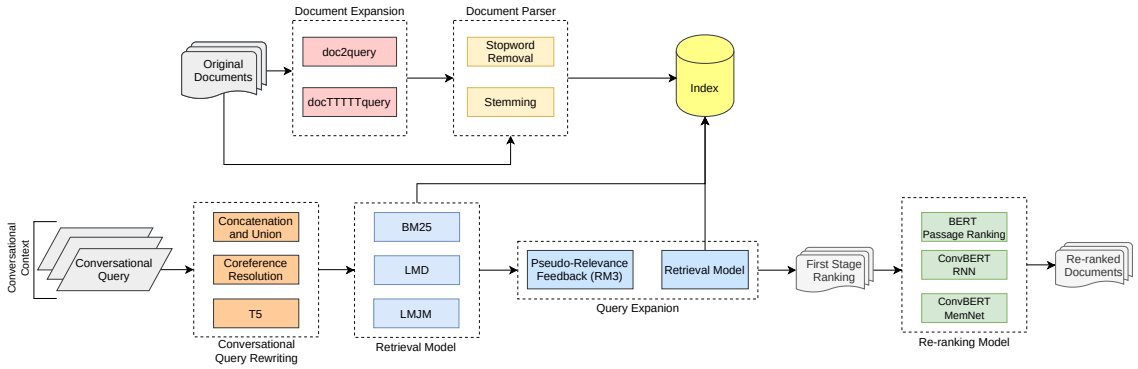


Figure 6.10: Complete architecture and pipeline of the system developed. In each box it is possible to apply none or various algorithms. Indexing and document/passage expansion approaches are done offline (top half). Query rewriting and expansion, retrieval and re-ranking are performed online (bottom half).

The top half describes the offline methods responsible for the creation of the index. In particular, passage expansion via pre-trained models, and the document-passage parser:

- **Document-passage parser** (Section 6.2.1) - We experimented with different stop word lists and with stemming. We found that the best results are achieved with a combination of a shorter list of stop words and stemming. Stemming, in particular, was very important to improve recall, making it easier to discover relevant passages by reducing the vocabulary mismatch between the query and passage terms.

- **Document/passage expansion** (appendices A and B) - We tested the models doc2-query [41] and docTTTTTquery [39]. Both are expensive to compute due to the large size of the passage collection, but this calculation is only performed once before

indexing, so retrieval time is not significantly increased. In the results, docTTTTTquery surpassed doc2query by its ability to generate more diverse questions. Despite this, both models did not bring a substantial improvement in performance.

The bottom half of the Figure shows the online portion of the system:

- **Conversational query rewriting** (Section 6.3.2) - We explored diverse methods, from simple concatenation with previous queries to neural models such as AllenNLP [18] and T5 [47]. The simple concatenation and union strategies proved to be effective in improving recall but add irrelevant terms to the query that may negatively influence performance. The AllenNLP model also showed good performance by resolving the coreferences in the query but lacks the ability to resolve implicit context, which in the conversational search setting is very important. From all these methods, the best performing model was our fine-tuned T5 model, being able to resolve explicit and implicit context.

- **Retrieval models** (Section 6.2.2) - We tested established information retrieval models: BM25 [51], LMD, and LMJM [67], and showed that LMD was the best performing one in this dataset. This is consistent with previous knowledge that stated that LMD works best for shorter queries [67], which in a conversation are very common.

- **Query Expansion with pseudo-relevance feedback** (appendices A and B) - After the first search with the retrieval model, we tested with RM3 [32] to expand the query terms. This approach showed small improvements in most cases, however, the retrieval time was increased due to creating a longer query and performing a second search over the index.

- **Re-ranking models** (Section 6.4) - We used state-of-the-art re-rankers using the BERT model [12] trained in a relevance classification task [37]. In Section 6.4.3, we also extended BERT to use the conversational context via two new architectures based on RNNs [2, 20] and Memory Networks [54, 59] that showed that the context of the conversation is present in the conversational utterances and embeddings, surpassing a linear model trained on the same data. When combining the re-ranking models implemented with the query rewriting methods developed, we were able to surpass state-of-the-art when compared to TREC CAsT 2019 baselines [10]. These results proved the effectiveness of the recall maximization step done in the first-stage retrieval and the power of the new pre-trained models both in conversation query rewriting [18, 30] and in re-ranking [37, 40].

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we studied and implemented the various components of a conversational search system. The goal of this system is to provide a more natural interaction with the user by allowing the user to formulate sequences of natural language queries to retrieve information about open topics.

We specifically tackled the conversational search task as a context tracking task, where the context can be seen in both previous queries and system answers. To model this, we explored various ways of tracking the context via query rewriting and contextual re-ranking approaches. In particular, we propose a three-stage architecture composed by query rewriting, retrieval, and re-ranking.

For query rewriting, we used previous queries as context to create non-conversational queries to allow search using traditional information retrieval components. We explored simple concatenation-based approaches that yielded some improvements over the original conversational queries. We also explored deep-neural models, such as AllenNLP's [18] coreference resolution model, and a specific conversational query rewriting method based on the T5 model. We analyzed the performance of AllenNLP and saw that it could only disambiguate explicit coreferences, hence we fine-tuned a T5 model [30], resulting in a model that was able to resolve explicit and implicit coreferences.

Regarding the re-ranking component, we explored current state-of-the-art methods based on the large pre-trained model BERT [12, 37]. We also expanded BERT's architecture to make use of the conversational context by using its embeddings as input to an RNN [2, 20] or a Memory Network [54, 59]. We demonstrated the importance of the context with the ConvBERT architectures surpassing a linear model without context trained on the same data. We also showed that it is possible to use a context-based architecture in

a re-ranking scenario, producing results similar to a basic BERT model while only being trained on conversational data, which at this time is very limited.

Hence, from a research point of view, the key takeaways are as follows:

- **Conversational Query Rewriting** - Rewriting the user utterance using the conversation context is critical. We tested different query rewriting approaches, and T5 delivered the most significant boost in performance when combined with a coreference resolution model and with the previous queries in the conversation.
- **BERT embeddings can be used to capture conversational context** - The proposed context-aware re-ranking models showed that modeling the sequence of BERT embeddings throughout the conversation results in an improvement of 3.95% over a linear model.

Besides the scientific contributions, the other key achievements are the following:

- **Thorough evaluation of state-of-the-art methods** - Using the recently developed conversational search dataset TREC CAsT [10], we were able to assess the performance of the various components, allowing for an extensive evaluation of the characteristics of conversational search.
- **State-of-the-art results on the TREC CAsT 2019 dataset** - The complete architecture achieved state-of-the-art results in TREC CAsT 2019 [10], mainly due to the better query rewriting performance of the fine-tuned T5 model, which allowed the search of more relevant passages that can then be re-ranked by the more complex model.
- **Participation in conferences** - We implemented a system to participate in TREC CAsT 2020 and had one paper accepted to ECIR 2021 and another currently under review.

7.2 Publications

7.2.1 TREC CAsT 2020 Submission

With all the work done in this thesis using TREC CAsT's 2019 dataset, we submitted our system's runs and a paper titled: *"NOVA at TREC 2020 Conversational Assistance Track"* to TREC CAsT 2020 [9].

The dataset in 2020 is only composed of the MS MARCO [36] and TREC CAR [13] datasets, removing WaPo [22] due to a deduplication error. Concerning the queries, there are 24 new topics available with the respective raw (conversational) and manually resolved queries, along with the queries resolved by an algorithm developed by the track organizers. Also available in this year's edition is a possible answer (passage) retrieved for each query that can be used as the conversational context for upcoming turns.

Another important change is the introduction of queries about previous answers (passages) retrieved, which makes the task more challenging and closer to a real conversation.

7.2.2 Papers Submitted

The system and results obtained in this thesis were also used in two papers:

- "*Open-Domain Conversational Search Assistant with Transformers*" [16] accepted in ECIR 2021[1] details the development of a conversational search assistant composed by 4 main components: (1) query rewriting, (2) retrieval, (3) re-ranking, and (4) answer summarization.

- "*Knowledge-driven Answer Generation for Conversational Search*", details a conversational search assistant in which the answer summarization takes into account the most salient entities of a particular conversation, and is currently under review.

7.3 Impact of Conversational Search in IR

In our modern and connected everyday lives, the search for information is one of the most basic actions. Introducing conversational features to IR is a step forward in allowing a more natural interaction with intelligent agents. It is also applicable in diverse markets, such as e-commerce, voice assistants, and others. As such, in the upcoming years, it is expected that conversational search will be just as integrated into our lives as searching for information using a typical search engine like Google.

7.4 Future work

Despite the good results in TREC CAsT 2019, we believe that there is still room for improvement. In particular, there are several topics we would like to explore:

- **Conversation context of passages** - query rewriting methods can also make use of the context in the passages retrieved. This would be done by using a model capable of perceiving the more relevant words in the passages returned, similar to an attention mechanism [12, 55, 56], and use them when in the appropriate moment to rewrite the user's query.

- **Long conversations** - we observed that the performance decreases by turn depth, and although it is optimistic to assume that this is an easily solvable problem, we believe that there are still ways to address this, such as asking clarification questions to ensure that the information being retrieved is relevant to the user [1, 62].

[1] https://www.ecir2021.eu/accepted-papers/

- **Re-ranking models** - with the emergence of new architectures based on pre-trained models every year, there is a possibility to experiment with different models to achieve better results [31, 47, 65]. As an example of this, in [38] is presented an adaption of the sequence-to-sequence model T5 to the passage re-ranking task [38]. Another problem that we encountered was that the size of the conversation may not fit entirely in the model, limitation that is being addressed with new architectures based on large pre-trained autoregressive models [65].

- **Conversational data augmentation** - the conversational re-ranking architectures developed would benefit from more training data, so an adaptation of conversational question-answering datasets [3, 50] to use in the conversational search task is also an avenue for future work.

- **Summarized answers** - there is also the possibility of using text summarization models to generate a single informative answer from the passages retrieved [28, 47]. This is yet another next step in allowing for a more natural interaction with the system and is very appropriate for conversational systems, where we want quick interactions with minimum effort on the user's side to search for the correct information.

Bibliography

[1] M. Aliannejadi, H. Zamani, F. Crestani, and W. B. Croft. "Asking Clarifying Questions in Open-Domain Information-Seeking Conversations." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*. ACM, 2019, pp. 475–484. DOI: 10.1145/3331184.3331265.

[2] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2014, pp. 1724–1734. DOI: 10.3115/v1/d14-1179.

[3] E. Choi, H. He, M. Iyyer, M. Yatskar, W. Yih, Y. Choi, P. Liang, and L. Zettlemoyer. "QuAC: Question Answering in Context." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Association for Computational Linguistics, 2018, pp. 2174–2184. DOI: 10.18653/v1/d18-1241.

[4] C. L. A. Clarke. "WaterlooClarke at the TREC 2019 Conversational Assistant Track." In: *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019*. Vol. 1250. NIST Special Publication. National Institute of Standards and Technology (NIST), 2019. URL: https://trec.nist.gov/pubs/trec28/papers/WaterlooClarke.C.pdf.

[5] J. S. Culpepper, F. Diaz, and M. D. Smucker. "Research Frontiers in Information Retrieval: Report from the Third Strategic Workshop on Information Retrieval in Lorne (SWIRL 2018)." In: *SIGIR Forum* 52.1 (2018), pp. 34–90. DOI: 10.1145/3274784.3274788.

[6] Z. Dai and J. Callan. "Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval." In: *CoRR* abs/1910.10687 (2019). arXiv: 1910.10687.

[7] Z. Dai and J. Callan. "Deeper Text Understanding for IR with Contextual Neural Language Modeling." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*. ACM, 2019, pp. 985–988. DOI: 10.1145/3331184.3331303.

[8] Z. Dai, C. Xiong, J. Callan, and Z. Liu. "Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search." In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*. ACM, 2018, pp. 126–134. DOI: 10.1145/3159652.3159659.

[9] J. Dalton, C. Xiong, and J. Callan. *The TREC Conversational Assistance Track (CAsT)*. Jan. 2020. URL: http://www.treccast.ai/.

[10] J. Dalton, C. Xiong, and J. Callan. "TREC CAsT 2019: The Conversational Assistance Track Overview." In: *CoRR* abs/2003.13624 (2020). arXiv: 2003.13624.

[11] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. F. Moura, D. Parikh, and D. Batra. "Visual Dialog." In: *CoRR* abs/1611.08669 (2016). arXiv: 1611.08669.

[12] J. Devlin, M. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423.

[13] L. Dietz, B. Gamari, and J. Dalton. *TREC CAR 2.1: A Data Set for Complex Answer Retrieval*. July 2018. URL: http://trec-car.cs.unh.edu.

[14] E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston. "Wizard of Wikipedia: Knowledge-Powered Conversational agents." In: *CoRR* abs/1811.01241 (2018). arXiv: 1811.01241.

[15] A. Elgohary, D. Peskov, and J. L. Boyd-Graber. "Can You Unpack That? Learning to Rewrite Questions-in-Context." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Association for Computational Linguistics, 2019, pp. 5917–5923. DOI: 10.18653/v1/D19-1605.

[16] R. Ferreira, M. Leite, D. Semedo, and J. Magalhaes. "Open-Domain Conversational Search Assistant with Transformers." In: *Advances in Information Retrieval - 43nd European Conference on IR Research, ECIR 2021, Lucca, Italy, March 28-April 1, 2021, Proceedings*. Lecture Notes in Computer Science. Springer, 2021. arXiv: 2101.08197 [cs.IR].

[17] J. Gao, M. Galley, and L. Li. "Neural Approaches to Conversational AI." In: *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, Tutorial Abstracts*. Association for Computational Linguistics, 2018, pp. 2–7. DOI: 10.18653/v1/P18-5002.

[18] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. E. Peters, M. Schmitz, and L. Zettlemoyer. "AllenNLP: A Deep Semantic Natural Language Processing Platform." In: *CoRR* abs/1803.07640 (2018). arXiv: 1803.07640.

[19] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. "A Deep Relevance Matching Model for Ad-hoc Retrieval." In: *CoRR* abs/1711.08611 (2017). arXiv: 1711.08611.

[20] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory." In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[21] H. Huang, E. Choi, and W. Yih. "FlowQA: Grasping Flow in History for Conversational Machine Comprehension." In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=ByftGnR9KX.

[22] S. Huang, D. Harman, and I. Soboroff. *TREC Washington Post Corpus*. Dec. 2019. URL: https://trec.nist.gov/data/wapost/.

[23] Y. Ju, F. Zhao, S. Chen, B. Zheng, X. Yang, and Y. Liu. "Technical report on Conversational Question Answering." In: *CoRR* abs/1909.10772 (2019). arXiv: 1909.10772.

[24] D. Jurafsky and J. H. Martin. *Speech and Language Processings 3rd ed. draft*. Oct. 2019. URL: https://web.stanford.edu/~jurafsky/slp3/edbook_oct162019.pdf.

[25] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1412.6980.

[26] V. Lavrenko and W. B. Croft. "Relevance-Based Language Models." In: *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*. ACM, 2001, pp. 120–127. DOI: 10.1145/383952.383972.

[27] K. Lee, L. He, M. Lewis, and L. Zettlemoyer. "End-to-end Neural Coreference Resolution." In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Association for Computational Linguistics, 2017, pp. 188–197. DOI: 10.18653/v1/d17-1018.

[28] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Association for Computational Linguistics, 2020, pp. 7871–7880. URL: https://www.aclweb.org/anthology/2020.acl-main.703/.

[29] C.-Y. Lin. "ROUGE: A Package for Automatic Evaluation of Summaries." In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013.

[30] S. Lin, J. Yang, R. Nogueira, M. Tsai, C. Wang, and J. Lin. "Conversational Question Reformulation via Sequence-to-Sequence Architectures and Pretrained Language Models." In: *CoRR* abs/2004.01909 (2020). arXiv: 2004.01909.

[31] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692.

[32] Y. Lv and C. Zhai. "A comparative study of methods for estimating query language models with pseudo feedback." In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*. ACM, 2009, pp. 1895–1898. DOI: 10.1145/1645953.1646259.

[33] D. Metzler and W. B. Croft. "Linear feature-based models for information retrieval." In: *Inf. Retr.* 10.3 (2007), pp. 257–274. DOI: 10.1007/s10791-006-9019-z.

[34] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. "Distributed Representations of Words and Phrases and their Compositionality." In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. 2013, pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.

[35] G. Neff and P. Nagy. "Talking to Bots: Symbiotic Agency and the Case of Tay." In: *International Journal of Communication* 10 (Oct. 2016), pp. 4915–4931. URL: https://ijoc.org/index.php/ijoc/article/view/6277.

[36] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. "MS MARCO: A Human Generated MAchine Reading COmprehension Dataset." In: *CoRR* abs/1611.09268 (2016). arXiv: 1611.09268.

[37] R. Nogueira and K. Cho. "Passage Re-ranking with BERT." In: *CoRR* abs/1901.04085 (2019). arXiv: 1901.04085.

[38] R. Nogueira, Z. Jiang, R. Pradeep, and J. Lin. "Document Ranking with a Pretrained Sequence-to-Sequence Model." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*. Association for Computational Linguistics, 2020, pp. 708–718. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.63/.

[39] R. Nogueira and J. Lin. "From doc2query to docTTTTTquery." In: (2019). URL: https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-v2.pdf.

[40] R. Nogueira, W. Yang, K. Cho, and J. Lin. "Multi-Stage Document Ranking with BERT." In: *CoRR* abs/1910.14424 (2019). arXiv: 1910.14424.

[41] R. Nogueira, W. Yang, J. Lin, and K. Cho. "Document Expansion by Query Prediction." In: *CoRR* abs/1904.08375 (2019). arXiv: 1904.08375.

[42] Y. Ohsugi, I. Saito, K. Nishida, H. Asano, and J. Tomita. "A Simple but Effective Method to Incorporate Multi-turn Context with BERT for Conversational Machine Comprehension." In: *CoRR* abs/1905.12848 (2019). arXiv: 1905.12848.

[43] K. Papineni, S. Roukos, T. Ward, and W. Zhu. "Bleu: a Method for Automatic Evaluation of Machine Translation." In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*. ACL, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135.

[44] Y. Qiao, C. Xiong, Z. Liu, and Z. Liu. "Understanding the Behaviors of BERT in Ranking." In: *CoRR* abs/1904.07531 (2019). arXiv: 1904.07531.

[45] C. Qu, L. Yang, M. Qiu, W. B. Croft, Y. Zhang, and M. Iyyer. "BERT with History Answer Embedding for Conversational Question Answering." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*. ACM, 2019, pp. 1133–1136. DOI: 10.1145/3331184.3331341.

[46] C. Qu, L. Yang, M. Qiu, Y. Zhang, C. Chen, W. B. Croft, and M. Iyyer. "Attentive History Selection for Conversational Question Answering." In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*. ACM, 2019, pp. 1391–1400. DOI: 10.1145/3357384.3357905.

[47] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." In: *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. URL: http://jmlr.org/papers/v21/20-074.html.

[48] P. Rajpurkar, R. Jia, and P. Liang. "Know What You Don't Know: Unanswerable Questions for SQuAD." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*. Association for Computational Linguistics, 2018, pp. 784–789. DOI: 10.18653/v1/P18-2124.

[49] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text." In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. The Association for Computational Linguistics, 2016, pp. 2383–2392. DOI: 10.18653/v1/d16-1264.

[50] S. Reddy, D. Chen, and C. D. Manning. "CoQA: A Conversational Question Answering Challenge." In: *Trans. Assoc. Comput. Linguistics* 7 (2019), pp. 249–266. URL: https://transacl.org/ojs/index.php/tacl/article/view/1572.

[51] S. Robertson and H. Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond." In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389. ISSN: 1554-0669. DOI: 10.1561/1500000019.

[52] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. *Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models*. 2016. arXiv: 1507.04808 [cs.CL].

[53] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J. Nie. "A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion." In: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. ACM, 2015, pp. 553–562. DOI: 10.1145/2806416.2806493.

[54] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. "End-To-End Memory Networks." In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 2015, pp. 2440–2448. URL: http://papers.nips.cc/paper/5846-end-to-end-memory-networks.

[55] A. Summerville, J. Hashemi, J. Ryan, and W. Ferguson. "How to Tame Your Data: Data Augmentation for Dialog State Tracking." In: *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*. Online: Association for Computational Linguistics, July 2020, pp. 32–37. DOI: 10.18653/v1/2020.nlp4convai-1.4.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is All you Need." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.

[57] N. Voskarides, D. Li, A. Panteli, and P. Ren. "ILPS at TREC 2019 Conversational Assistant Track." In: *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019*. Vol. 1250. NIST Special Publication. National Institute of Standards and Technology (NIST), 2019. URL: https://trec.nist.gov/pubs/trec28/papers/UvA.ILPS.C.pdf.

[58] N. Voskarides, D. Li, P. Ren, E. Kanoulas, and M. de Rijke. "Query Resolution for Conversational Search with Limited Supervision." In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020). DOI: 10.1145/3397271.3401130.

[59] J. Weston, S. Chopra, and A. Bordes. "Memory Networks." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1410.3916.

[60] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. "HuggingFace's Transformers: State-of-the-art Natural Language Processing." In: *CoRR* abs/1910.03771 (2019). arXiv: 1910.03771.

[61] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. "End-to-End Neural Ad-hoc Ranking with Kernel Pooling." In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. ACM, 2017, pp. 55–64. DOI: 10.1145/3077136.3080809.

[62] J. Xu, Y. Wang, D. Tang, N. Duan, P. Yang, Q. Zeng, M. Zhou, and X. Sun. "Asking Clarification Questions in Knowledge-Based Question Answering." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Association for Computational Linguistics, 2019, pp. 1618–1629. DOI: 10.18653/v1/D19-1172.

[63] J. Yang, S. Lin, C. Wang, J. Lin, and M. Tsai. "Query and Answer Expansion from Conversation History." In: *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019*. Vol. 1250. NIST Special Publication. National Institute of Standards and Technology (NIST), 2019. URL: https://trec.nist.gov/pubs/trec28/papers/CFDA_CLIP.C.pdf.

[64] P. Yang, H. Fang, and J. Lin. "Anserini: Enabling the Use of Lucene for Information Retrieval Research." In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. ACM, 2017, pp. 1253–1256. DOI: 10.1145/3077136.3080721.

[65] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le. "XLNet: Generalized Autoregressive Pretraining for Language Understanding." In: *CoRR* abs/1906.08237 (2019). arXiv: 1906.08237.

[66] Y. Yeh and Y. Chen. "FlowDelta: Modeling Flow Information Gain in Reasoning for Conversational Machine Comprehension." In: *Proceedings of the 2nd Workshop on Machine Reading for Question Answering, MRQA@EMNLP 2019, Hong Kong, China, November 4, 2019*. Association for Computational Linguistics, 2019, pp. 86–90. DOI: 10.18653/v1/D19-5812.

[67] C. Zhai and J. Lafferty. "A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval." In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '01. New Orleans, Louisiana, USA: Association for Computing Machinery, 2001, pp. 334–342. ISBN: 1581133316. DOI: 10.1145/383952.384019.

Appendix A

Query and Document Expansion Retrieval Results

A.1 Query Expansion Using Pseudo-Relevance Feedback

In this section, we discuss the usage and results obtained using the pseudo-relevance feedback technique RM3 [26] in combination with the developed query rewriting techniques. As explained in Section 4.3, RM3 is composed of 3 parameters: α (alpha), the weight attributed to original query terms, the number of feedback documents, and the number of feedback terms used to expand the query.

We used Anserini's implementation of RM3 [64] and experimented with different parameters in the index which contains the full collection composed of MS MARCO, TREC CAR, and WaPo. The search space of each parameter and the parameters that obtained the highest recall in the training set using the *Pref+CorefPronoun* queries are provided in table A.1.

Table A.1: Tunable parameters for RM3, the search spaces considered, and the parameters that achieved the highest recall in the training set using the *Pref+CorefPronoun* queries.

RM3 - Parameters	Search Space	Best Parameters
Number of Feedback Documents	5 - 45 step=10	15
Number of Feedback Terms	5 - 20 step=5	20
Original Query Weight (α)	0.6 - 0.9 step=0.1	0.9

Table A.2 presents the results in the evaluation set of applying RM3 with the fine-tuned parameters to the various query rewriting methods. With the use of RM3 our main goal was to improve recall by performing a search with other relevant terms, that in turn would provide us with different passages. The results show that RM3 was useful for

improving recall, but it was also advantageous in improving other metrics across multiple query rewriting techniques, with the added benefit of not needing any input from the user. With RM3, we achieved the best results in terms of recall with *CorefPronoun+Union* achieving 0.759, closing the gap to the manually coreference resolved queries (*Manual*) with 0.820 and 0.834, with and without RM3 respectively.

Table A.2: Results in the evaluation set of the query rewriting techniques using RM3 with parameters α=0.8, number feedback documents=5, and number feedback terms=15 using LMD with μ=1000.

Index Containing MS MARCO / TREC CAR / WaPo						
Queries	**RM3**	**Recall**	**MAP**	**MRR@10**	**nDCG@3**	**P@3**
Raw	✗	0.454	0.141	0.336	0.167	0.262
Raw	✓	0.463	0.153	0.344	0.176	0.272
Manual	✗	0.820	0.327	0.694	0.406	0.590
Manual	✓	0.834	0.343	0.706	0.408	0.593
Pref+CorefPronoun	✗	0.715	0.246	0.571	0.304	0.462
Pref+CorefPronoun	✓	0.712	0.262	0.581	0.315	0.482
CorefPronoun+Union	✗	0.737	0.216	0.557	0.278	0.430
CorefPronoun+Union	✓	**0.759**	0.254	0.588	0.302	0.476
T5	✗	0.697	0.251	0.597	0.322	0.474
T5	✓	0.717	0.271	0.602	0.332	0.474
CorefPronoun+T5	✗	0.733	0.251	0.596	0.331	0.484
CorefPronoun+T5	✓	0.744	**0.274**	**0.608**	**0.345**	**0.495**

It is also important to note that although pseudo-relevance feedback improves the results in this situation, this may not be the case in others because the insertion of other terms can deviate the topic and introduce noise or irrelevant terms to the current query. Another important aspect is the retrieval time. RM3 increases retrieval time because it needs to perform two searches over the same collection, which in our case is quite large (in total over 47 million passages), so it is necessary to weigh if the improvements justify the increase in retrieval time.

A.2 Query Expansion and Document/Passage Expansion

A.2.1 Document/Passage Expansion

To expand the passages, we used the two neural-based techniques previously discussed in Section 3.2.2: doc2query [41], and docTTTTTquery [39]. As stated before, expanding passages using neural approaches is time-consuming and expensive to compute, especially in our case where the collection is of several million passages. Because of this, we used the publicly available predicted queries for the MS MARCO dataset for both doc2query and docTTTTTquery, expanding each passage with 5 predicted queries. In these experiments, we did not expand TREC CAR because of its large size (29 million passages), and because

of the GPU and TPU requirements of the models. On WaPo, we didn't expand any of its passages since this collection was removed from the final assessment in TREC CAsT 2019 [10].

In the previous section, we saw that it is possible to use RM3 to improve all metrics in most cases at the expense of an increase in retrieval time. Thanks to the flexibility of the various methods, we can evaluate the combination of both RM3 and passage expansion. Table A.3 shows the full overview of the multiple query rewriting techniques and RM3 with the parameters optimized in the previous experiment in the original and expanded indices. The bold values indicate the best results achieved, excluding the *Manual* queries, and the underlines indicate the best performing combination per query rewriting method.

Considering only the different expanded indices with no query expansion (no RM3), we see that doc2query achieves mixed results in all query rewriting methods and metrics. Using docTTTTTquery, on the other hand, generally achieves better results in most methods and metrics.

From the results of combining RM3 and document expansion, we can conclude that this can be effective since we obtained the highest results in the metrics: Recall, MRR@10, nDCG@3, and P@3 with the combination of RM3 and docTTTTTquery. In particular, the combination of RM3 and docTTTTTquery achieved the best value for recall with 0.766 using *CorefPronoun+Union*, coming a long way from the original conversational queries (*Raw*) with a recall of 0.454 using the most basic setup.

Summing up, in this section, we showed that not only is it possible to use the passage expansion technique docTTTTTquery to improve the results of the first-stage retrieval but that we can also combine it with query expansion, in particular RM3 (pseudo-relevance feedback), to further improve these results. As always, when developing a system, we need to have a critical view of the different aspects. Specifically, it is important to see if expanding passages, which is done before indexing but is very time consuming (despite it only being performed once), and the use of a pseudo-relevance model, which increases retrieval time, are required to achieve the best results.

Table A.3: Results in the evaluation set of the query rewriting methods with RM3 and passage expansion in the evaluation set using LMD with μ=1000. The passage expansion models use 5 predicted queries.

Index Containing MS MARCO / TREC CAR / WaPo							
Queries	**MS MARCO Expansion**	**RM3**	**Recall**	**MAP**	**MRR@10**	**nDCG@3**	**P@3**
Raw	✗	✗	0.454	0.141	0.336	0.167	0.262
Raw	✗	✓	0.463	0.153	0.344	0.176	0.272
Raw	doc2query	✗	0.461	0.135	0.346	0.173	0.262
Raw	doc2query	✓	0.467	0.146	0.355	0.185	0.281
Raw	docTTTTTquery	✗	0.471	0.142	0.360	0.190	0.279
Raw	docTTTTTquery	✓	0.480	0.153	0.371	0.199	0.285
Manual	✗	✗	0.820	0.327	0.694	0.406	0.590
Manual	✗	✓	0.834	0.343	0.706	0.408	0.593
Manual	doc2query	✗	0.817	0.311	0.710	0.396	0.570
Manual	doc2query	✓	0.828	0.319	0.718	0.411	0.576
Manual	docTTTTTquery	✗	0.822	0.314	0.716	0.419	0.592
Manual	docTTTTTquery	✓	0.825	0.322	0.721	0.419	0.588
Pref+CorefPronoun	✗	✗	0.715	0.246	0.571	0.304	0.462
Pref+CorefPronoun	✗	✓	0.712	0.262	0.581	0.315	0.482
Pref+CorefPronoun	doc2query	✗	0.720	0.234	0.590	0.305	0.472
Pref+CorefPronoun	doc2query	✓	0.721	0.244	0.606	0.316	0.476
Pref+CorefPronoun	docTTTTTquery	✗	0.725	0.237	0.582	0.304	0.468
Pref+CorefPronoun	docTTTTTquery	✓	0.722	0.246	0.575	0.308	0.480
CorefPronoun+Union	✗	✗	0.737	0.216	0.557	0.278	0.430
CorefPronoun+Union	✗	✓	0.759	0.254	0.588	0.302	0.476
CorefPronoun+Union	doc2query	✗	0.735	0.210	0.574	0.287	0.437
CorefPronoun+Union	doc2query	✓	0.758	0.244	0.620	0.324	0.505
CorefPronoun+Union	docTTTTTquery	✗	0.742	0.210	0.590	0.288	0.436
CorefPronoun+Union	docTTTTTquery	✓	**0.766**	0.248	0.603	0.310	0.470
T5	✗	✗	0.697	0.251	0.597	0.322	0.474
T5	✗	✓	0.717	0.271	0.602	0.332	0.474
T5	doc2query	✗	0.697	0.243	0.619	0.320	0.470
T5	doc2query	✓	0.715	0.258	0.635	0.339	0.484
T5	docTTTTTquery	✗	0.707	0.247	0.611	0.335	0.493
T5	docTTTTTquery	✓	0.720	0.264	0.636	**0.357**	**0.505**
CorefPronoun+T5	✗	✗	0.733	0.251	0.596	0.331	0.484
CorefPronoun+T5	✗	✓	0.744	**0.274**	0.608	0.345	0.495
CorefPronoun+T5	doc2query	✗	0.733	0.241	0.612	0.327	0.480
CorefPronoun+T5	doc2query	✓	0.742	0.257	0.628	0.339	0.499
CorefPronoun+T5	docTTTTTquery	✗	0.742	0.245	0.619	0.341	0.493
CorefPronoun+T5	docTTTTTquery	✓	0.746	0.261	**0.637**	0.354	0.499

Appendix B

Query and Document Expansion Re-ranking Results

B.1 Query Expansion and Document/Passage Expansion

Thanks to the modularity and flexibility of the system, in table B.1, we show the results of applying the pseudo-relevance feedback model RM3 [32], and the passage expansion method docTTTTTquery [39] in conjunction with the BERT *LARGE* re-ranking model explained in Section 6.4.1.

As we saw in appendix A and again in table B.1, RM3 and docTTTTTquery can be used to improve the recall. When we add re-ranking, we see that RM3 can also be used to improve other metrics depending on the query used. In particular, the most improved method with the addition of RM3 was T5, going from an nDCG@3 of 0.475 to 0.486.

With the addition of docTTTTTquery, the results show mostly lower values for most metrics, especially in MRR@10 and nDCG@3 in the *Prefix+CorefPronoun* query. This may be due to the queries that expanded the passages negatively influencing the re-ranker scores since the BERT model was trained on passages that did not have any type of expansion.

When combining both RM3 and docTTTTTquery, we see that the results on the metrics that evaluate the earlier positions are again worse than the ones obtained with BERT *LARGE* without any additions.

In summary, although we saw an increase in recall in most query rewriting methods with the introduction of RM3 and docTTTTTquery, these did not translate in re-ranking to an increase in metrics that evaluate the earlier positions, which are the main focus of this task. This may be due to the fine-tuning of the models not accounting for this expansion. Adding to this, RM3, as explained before, increases retrieval time because it performs two searches over the full index, so we consider the gain obtained in some

query types not enough to justify the cost.

Table B.1: Results in the evaluation set of query rewriting with RM3 and docTTTTT-query passage expansion on the MS MARCO dataset, using LMD with μ=1000 and a BERT *LARGE* re-ranker trained on MS MARCO in the top-1000 passages. The passage expansion model uses 5 predicted queries.

Index Containing MS MARCO / TREC CAR / WaPo							
Queries	**MS MARCO docTTTTTquery**	**RM3**	**Recall**	**MAP**	**MRR@10**	**nDCG@3**	**P@3**
Raw	✗	✗	0.454	0.181	0.456	0.272	0.385
Raw	✗	✓	0.463	0.183	0.461	0.273	0.385
Raw	✓	✗	0.471	0.183	0.446	0.263	0.383
Raw	✓	✓	0.480	0.186	0.451	0.267	0.385
Manual	✗	✗	0.820	0.389	0.857	0.577	0.757
Manual	✗	✓	0.834	0.395	0.868	0.585	0.765
Manual	✓	✗	0.822	0.388	0.860	0.582	0.748
Manual	✓	✓	0.825	0.386	0.855	0.578	0.740
Prefix+CorefPronoun	✗	✗	0.715	0.274	0.702	0.427	0.559
Prefix+CorefPronoun	✗	✓	0.712	0.271	0.703	0.428	0.559
Prefix+CorefPronoun	✓	✗	0.725	0.271	0.648	0.409	0.545
Prefix+CorefPronoun	✓	✓	0.722	0.267	0.689	0.410	0.551
CorefPronoun+Union / T5	✗	✗	0.737	**0.332**	**0.799**	0.509	0.674
CorefPronoun+Union / T5	✗	✓	0.759	0.329	0.798	**0.513**	**0.686**
CorefPronoun+Union / T5	✓	✗	0.742	0.331	0.776	0.507	0.674
CorefPronoun+Union / T5	✓	✓	**0.766**	0.330	0.775	0.505	0.665
T5	✗	✗	0.697	0.310	0.739	0.475	0.632
T5	✗	✓	0.717	0.321	0.753	0.486	0.642
T5	✓	✗	0.707	0.313	0.725	0.467	0.626
T5	✓	✓	0.720	0.316	0.723	0.467	0.626
CorefPronoun+T5	✗	✗	0.733	0.305	0.749	0.484	0.649
CorefPronoun+T5	✗	✓	0.744	0.306	0.765	0.489	0.653
CorefPronoun+T5	✓	✗	0.742	0.306	0.744	0.480	0.636
CorefPronoun+T5	✓	✓	0.746	0.305	0.740	0.479	0.634