Radboud University Nijmegen

Faculty of Science

The Effect of Conversational History and Query Rewriting on Conversational Search Performance

Master Thesis Computing Science - Data Science

Supervisor:
dr. F. Hasibi

Author:
Josip Kopršina

Co-Supervisor:
Hideaki Joko

Second reader:
prof. Arjen de Vries

October 2023

Abstract

Conversational search is the process of retrieving a document to a user, based on the user's question, in the form of conversational sentences. One of the common approaches is to use a pipeline of reformulating the user's input to make it easier to answer, retrieving a list of documents based on the user's query, and reranking them so that the best answer is returned to the user.

However, previous work reported that the scores seemed to get worse the longer the conversation goes on, and hypothesised that this is due to the models working worse on later turns. Our goal was to test whether this assumption is true and, if so, could we make the models work better on longer conversations.

Using the data from TREC CAsT 2020 version, we tested the assumption by setting up our own pipeline, inspired by the methods that reached the best scores in TREC CAsT 2020, focusing on approaches that use data from previous turns. The approach consisted of a BM25 retriever, a T5 reranker and a T5 query rewriter. We tried variations of the method to address this problem.

We show that the drop in performance in conversational question answering models is not always correlated with the conversation length, but it can be an artifact of the data; e.g., topic shift in later conversation turns.

Contents

1 Introduction

Conversational systems are increasingly being integrated into various everyday applications, due to their improved accuracy and user-friendly nature. Amazon Alexa and Apple Siri are two examples of conversational systems that provide a variety of useful features. One such feature is conversational search, which is the process of interacting with a system in a conversational manner via natural language to retrieve information.

Conversational search systems aim to give the user an experience of conversation instead of inputting questions into a search system (for example; google). As part of the conversational search system, there is an information retrieval system that looks at the user's question and retrieves a document that most suits it. When humans converse, we commonly start by giving context in the first sentence, so that the other person understands what the conversation is going to be about. The same is done when talking to a conversational system, which means the first question is usually self-sufficient and easily answerable. A question we type into a search bar is very similar to this. The questions in the conversation from there onward are generally structured differently, as we commonly omit context when continuing a conversation, assuming that the other person is keeping track of what has been said so far. This is a trivial problem for humans, but not for computers. Computers can only do what they are told and given detailed instructions on how to do. For search systems to understand the context and give proper information back to the user, the retrieval pipeline needs to be altered or the users need to still consider the conversation system a search engine. The focus is always on the first option because the goal is to make the user's experience as simple as possible.

In recent years, research on the topic has been done as part of the TREC Conversational Assistance Track (CAsT), which aims to improve conversational search system retrieval. The track establishes a yearly objective for researchers, breaking down the challenging task of conversational system retrieval into smaller, more easily manageable problems. They challenge all the participants to work on the same project using the same data and rules. In the 2020 version, the goal was to return the best possible block of information, called a passage, for each turn of conversations, where each conversation has a different length and focus. The given data was specifically made so that the query (question) the user has at any given turn can, but does not have to, depend on previous queries and retrieved passages. The previous tracks gave data that depended only on previous queries and ignored the returned passages. This meant that more attention needed to be paid to the passages in 2020 than in previous years.

The main focus of this paper is exploring how conversation history and conversation length influence the quality of conversation retrieval. This means that a dataset where each turn depends on all the previous conversation history is optimal for our research, which is exactly what the TREC CAsT 2020 dataset offers.

1.1 Approach

Our approach to study the aforementioned research question is to employ a pipeline consisting of the following steps:

- Query Rewriter
- Retriever
- Reranker

The query rewriter is responsible for the process of changing the user's query to make it easier for a computer to understand, so that data with more dense information is sent

to the retriever [8, 14, 25]. This typically means disambiguating the query, making the question answerable by itself without any knowledge of the context. It is needed as the user commonly not repeating some of the words used in previous queries due to the illusion of talking to a human, making the later turns in the conversation more difficult for the conversation system to answer correctly. The user's question is considered a search *query* while the returned document is simplified into a shorter text we will refer to as a *passage*. The retriever is given the user's query, goes through all the possible passages in the given corpora and returns a large list of possibly relevant documents along with their relevance scores. The focus in this part of the pipeline is on a high recall. A reranker shifts the list of documents, so that they are ordered by how likely they are to be useful. It decides which returned document is the most likely to have the data the user is looking for, focusing on high precision. The passage placed on the first spot is the one that will be shown to the user. The retriever and reranker steps mentioned are normal steps in retriever systems, while query rewriting is added to help with a lack of context in the queries.

1.2 Objective

It is reported that the length of the conversation influences the drop in quality in later turns [11]. We will be testing whether this is true or not. In Figure [1] we see that the manual runs have a straighter line than the canonical and automatic runs, which dip down over time. The manual runs use a manually rewritten query which represents the optimal query that would allow the retriever and reranker to find the optimal passage from the corpora. This means that the red line in the graph is not influenced by the query rewriter. The blue and yellow lines, on the other hand, are influenced by the query rewriter, and this shows us how big the average mistake of all the query rewriters in the TREC CAsT 2020 competition was. Due to the increase of the mistake over turn depth, the assumption is that the conversation length is to blame for the drop in canonical and automatic NDCG@3 scores. We will take a detailed look at the data and the quality of the models over longer conversations and try to see whether this is the case or if something else is responsible for this drop. If we were to give an optimal query to the retriever/reranker system, we should get the best passage returned to us, irrespective of which turn it is. The conversation length only influences how the user formulates the query, meaning that the user could be giving more difficult queries as time goes on. If, on the other hand, our query rewriter does its job well and turns the queries into ones that the retriever can work better with, the results of the overall process will improve. This means the focus will be on query rewriting, as the retriever and reranker are not affected by the difference in conversation length if the query is written properly. We look at the performance of the query rewriters using conversation history (previous queries and returned passages) in different ways, focusing on those shown to work best so far [11].

1.3 Approach

The method we use for testing is a mix of BM25 for retrieval, MonoT5 for reranking, and a T5 query rewriter. The focus is on different variations of the rewriter using the data from previous turns, meaning the reranker and retriever will stay the same through all the experiments. We run two different query rewriter models; the baseline model from TREC CAsT 2021 and a pre-trained T5 model we fitted by using the approach given by h2oloo [25]. The TREC baseline model is used to give us a simple and thoroughly tested baseline to which we can compare the one we fit ourselves. Query rewriting data usage approaches are modeled after the ones used by h2oloo [25], in which they work with all

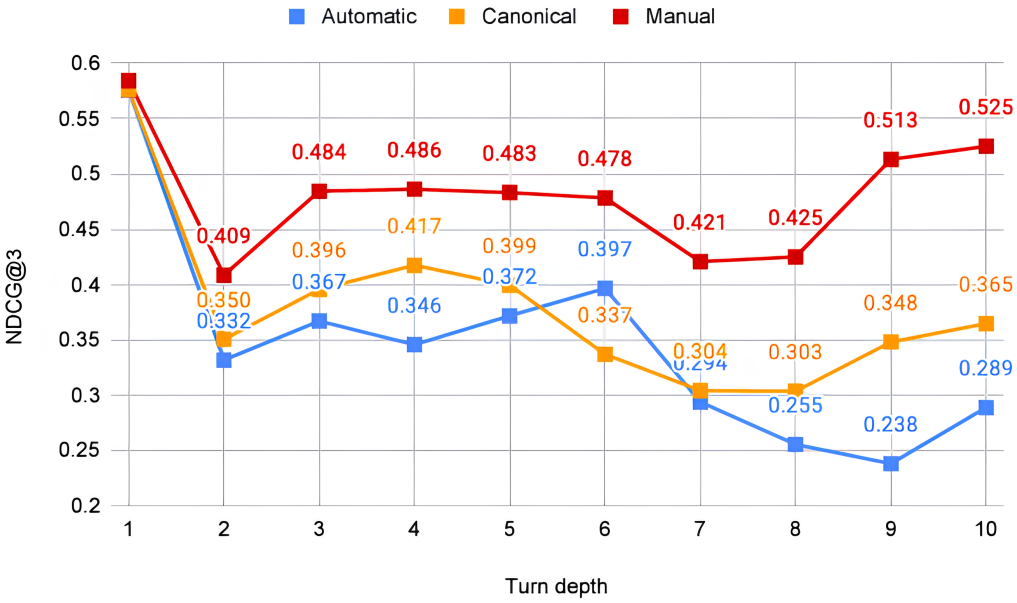

Figure 1: NDCG@3 at varying conversation turn depths from TREC CAsT 2020 overview paper [10]. The y-axis shows the NDCG@3 score while the x-axis shows the turn depth, focusing on the first 10 turns. Manual results outperform canonical and automatic results showing us that the query rewriting is underperforming.

previously rewritten or raw queries in addition to the sentence in the previous returned passage most similar to the current turn query. Although this method has achieved good results, we try to change some aspects that we assume could be improved. We also try to use text summarization to replace sentence similarity. Text summarization is used to create smaller chunks of data that consist of more concise information taken from the conversation history. This way the rewriter has the greatest amount of useful data and the least amount of irrelevant data which could hinder results.

The first part of our paper focuses on creating a pipeline inspired by common TREC CAsT submissions, more precisely h2oloo. H2oloo was a contestant in 2020 TREC CAsT track that reached one of the best results. Since the main focus of our paper is the influence of conversation history on retrieval, we focus on the query rewriting aspect of their paper, ignoring their retrieval and reranking approach and using basic methods instead. We go through the aforementioned paper and try to build our own model that reaches the same BLEU score they had. In case some information in the written process is not given, we use knowledge from other papers to fill in the blanks. This also includes testing the best method they reported against slight alterations that use more or less conversation history data, which they claim work worse. Some of the results they showed point to less data being better for the quality of the rewriter, while others show the opposite. We will test this out and come to a final conclusion as to which is better.

This paper tries to answer two research questions, while the focus is always on the influence of conversation history on the results of conversation retrieval.

RQ1: How does turn depth influence query rewriting results?

The most important part of our paper is doing an in-depth analysis of the data,

showing and examining the different aspects of the models through graphs to see if conversation length influences the quality of the conversation retrieval. So far, the research has shown that a longer conversation leads to a lower accuracy both for the query rewriting and for the relevant paragraph returned. It has been assumed that the length itself is what influences this drop. We took a detailed look at whether this is the case. If not, we made assumptions as to what might be the cause through what our graphs and data showed.

RQ2: Could the method be altered to lower the quality drop in longer conversations using text summarization?

We examine different aspects of the model pipeline and how to improve them, or to be more precise, we look for methods that would lower the drop in quality as time goes on. We discuss the drop in quality shown in Figure 1. Our goal is therefore to make simple changes to the h2oloo method with the intent of lowering the drop in quality of automatic and canonical runs by using the data more optimally so that conversation length is less of an issue. We use a different method of extracting information from the passages than the one used by h2oloo. Instead of taking one sentence most similar to the current query (which is what h2oloo did), we summarized the passage into chunks of varying lengths to see if the summarized data is more useful during query rewriting. In theory, the shorter length of the summarized chunk compared to the passage's length should lead to less data hindering the query rewriter, while the density of the data used should lead to the relevant information still being available to the model.

1.4 Contribution

This work looks into the influence of conversation length on retrieval performance. By referring to the existing work, we successfully achieve competitive results for TREC CAsT 2020. The code is publicly available [1] and can be used for further research on the relationship between conversation length and retrieval performance.

We show through a detailed analysis of the data that the turn depth of the conversation does not directly influence the quality of the retrieval. The NDCG@3 scores actually get better until the middle of the conversation, and then get worse close to the end. We hypothesise that the improvement in the middle is due to the model learning the topic and getting better with it over time, while the lower scores at the end are due to conversations changing subtopics later on. We believe a different dataset is needed to test this out in more detail (a dataset with conversations focusing heavily on one topic, and conversations with obvious topic shifts in later turns), as well as more research into hindering topic shift's influence on information retrieval.

We also try using summarization to improve the quality and consistency of the data taken from conversation history. We saw no improvement in the average scores when using summarization compared to sentence similarity, but we did see an improvement in the consistency of the scores, meaning the scores fluctuate less on different depths. This could indicate that summarization is a stable approach for conversation history usage.

[1] https://github.com/jkoprcina/ConversationSearchQueryReformatterTesting

2 Related Work

Conversational Search is a conversation between a user and a bot (computer) and consists of turns where the user asks questions and is given relevant passages which should answer his question [6]. This becomes harder the longer the conversation goes on as the user tends to find it more natural to talk to the bot as if talking to a real person. Because of this, problems like word zero anaphora, topic change and topic return are common. Real people deal with this intuitively but it creates problems for a bot. An example conversation is given in Figure 2. All of this leads to worse scores as the conversation goes on. There is a downward trend from an average of approximately 0.3 at the first turn to an average of 0.23 by turn eight, which can be seen in Figure 3 The aim of conversational search is to allow users to communicate naturally but still get the information they want. Research has been done on this but not using common features. TREC CAsT [2] has been started as a way to focus scientists on this problem. TREC CAsT 2019. [9] and 2020 [10] show a trend where query reformulation, document ranking and document reranking are the main aspects that are being worked on to improve this. We will be looking at multiple types of approaches to this problem and they can all be seen summed up in tables 1 and 2

Title: GDPR
Description: learn about GDPR, the privacy issues in social networks, and the addiction of it.

Turn	Conversation Utterances
1	What is the purpose of GDPR?
2	What is different compared to previous legislation?
3	What are the privacy implications of those technologies?
4	Oh, IP addresses are considered PII? What is the full range of personal data?
5	How do big companies adapt to GDPR?
6	OK. Tell me about the privacy issues in social networks.
7	What do they get in return for their privacy?
8	What are the symptoms of that addiction?

Figure 2: An example of how queries (here called "utterences") change as the conversation developes. Figure taken from the 2019. TREC CAsT overview paper [9].

2.1 Query Rewriting

Query Rewriting is the process of changing the query so that it contains all the needed information. The process should remove all ambiguous terms so that the ranker has an

[2] https://www.treccast.ai/

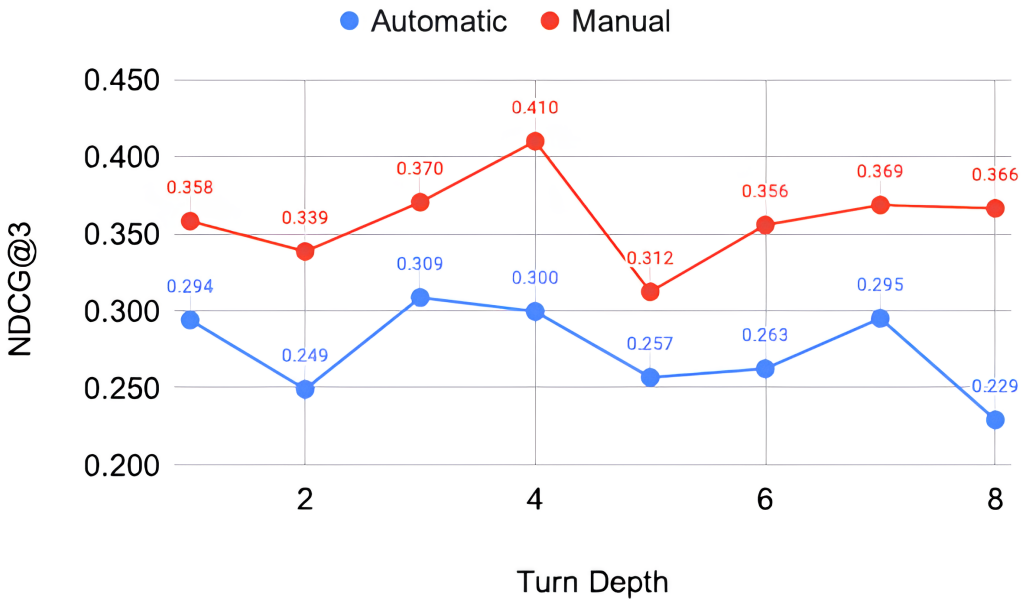

Figure 3: A graph showing how the NDCG@3 values get lower on average further in the conversation. TheNDCG@ score is shown on the y axis and the turn depth, indicating which turn in the conversation is taking place, is indicated on the x axis. The trend is not visible in manual approaches but only in automatic ones which indicates that proper query reformatting would solve this problem. Graph taken from the 2019. TREC CAsT overview paper [9].

easier job finding proper data to return to the user. So far this has been done mainly by either changing the query itself ([26]) or adding more text as additional information to the query ([44, 25, 14]). Most of the research so far has focused on looking at only past queries to find the data needed to rewrite the current turn query ([44, 14]). Some also took a look at part answers and those commonly get better results ([8, 25, 39, 3]). There is also a small subset of papers that uses a different approach all-together like [16]. Sadly the amount of papers that use responses is too low for us to come to an educated conclusion on what is better and how to best approach it. We will now talk a bit about some of the approaches that do and do not use the responses.

QuReTeC [44] uses bidirectional transformers to find which words in the previous queries that are relevant to the current one. This is done using a classification model which is fed the queries and the context (consisting of previous turn queries). The output is a binary value for each word in the context which states whether the word is relevant to the current turn query. The current query is then fed to the transformer alongside the words labeled as relevant with a SEP tag in between, so that BERT can differentiate which part is which. The output should be a new query which contains all the needed data for the ranker/reranker to find the needed information for the user. The upside of this approach is that BERT can be fed using a bag-of-words approach, which means that the order of the words appended to the query does not matter much. This on the other hand is bad because knowledge can be lost that way. Sometimes word order is important to create certain phrases. Another downside here is that the model only searches through the previous queries and ignores the previous retrieved passages, which can mean a loss of relevant information. A possible way to work around this would be

Table 1: A table showing the mentioned pipelines and the methods each one used in different steps. We can see that the most common retriever is BM25 (used in approximately 5 out of 6 situations), and all the mentioned rerankers are build upon different transformers. 5 out of 6 query rewriter are also build upon different transformers with the exception of HBKU.

Method Name	NDCG@3	Retriever	Reranker	Query Rewriter
QuReTeC	0.341	QL with Dirichlet smoothing	Reciprocal Rank Fusion	BERT
h2oloo	0.458	BM25	BERT-large	T5
GRILL	0.398	BM25/monoBERT	duoBERT	BERT
SBER	0.457	BM25	T5-based	T5
USI	0.339	BM25	ALBERT	ALBERT
HBKU	0.313	BM25	BERT	Tf-idf relevance feedback

Table 2: This table shows how different methods used conversation history. We see that half of them do not use passage data in any way. Two of them use similar sentences from passages which is the method we plan to test. For query usage we have more variety but most still use simple raw queries.

Method Name	Queries	Passages
QuReTeC	all queries	none
h2oloo	all previous rewriten queries	most similar sentence from paragraph
GRILL	all queries	none
SBER	previous query most similar to current	most similar sentence from paragraph
USI	previous relevant raw queries	none
HBKU	current query	current turn passages

to also use parts of the most relevant returned passages to enhance the context. The next method does something similar to what we advise.

H2oloo [25] reached the best automatic results in the 2020 version of TREC CAsT. It gives a query plus a self-made context to a T5 [36] transformer model, which then returns the reformatted query. The model is trained on CANARD [12]. The context is created by adding all the previous reformatted queries and also the sentence from the last returned document most similar to the current query. The later part is done by going through each sentence in the returned paragraph and calculating the similarity using a simple keyword count measure. Keywords were all words in the sentence that are a noun, verb or adjective by POS tagging. This outperformed using all the reformatted queries plus a sentence from the top returned document from every past turn. Sadly, this means that only the last turns answer is used in the context which means that a lot of possibly useful data might have been lost. It could also mean that focusing on only the last turn is optimal and maybe the conversation search is more a hindrance than help. We would also like to point out how the similarity measure used here is very simple. Sentence length is not taken into account which gives an advantage to longer sentences. Word order is also not observed so important phrases would be given too little relevance. Only three POS tags are looked at which means many words could have been neglected. We will look at this in more detail later as we will be using and changing this method in our work.

GRILL [14] uses a Transformer [43] that is initialised with pre-trained weights from BART[23] trained on Wikipedia and fine tuned on a summarization task. They try two approaches, one they call BART-FC, where the model tries to reproduce the original sequence of queries and the re-written final query, and one called BART-LT where it just produces the final query. BART-LT showed to be more effective. This method

completely ignores conversation answer history and only focuses on the queries. The reasoning given for this by the paper is that the data set has a low frequency of response dependencies and that they will therefore ignore it. This is true although we do not consider it enough to ignore the data completely. We will see will this method work better when we add data from the answers as well.

Semantic-based ellipsis reduction is used by [8] to tackle query reformatting. We will call this paper SBER from now on. The paper calls their model T5-CQR (T5 based coreference query reformation) because it removes coreference and is based on a T5 sequence to sequence model trained on CANARD. Previous turn dialog followed by the current query were given to the T5-CQR model which would give as output a coreference free query. After removing coreferences it expanded the current query with the most similar query written so far. Then it turns all relevant passages into queries using a Doc2Query model [31] and again adds the one most similar one to the current query. This twice expanded query is now the reformulated query used for retrieval. The similarity is calculated using a Transformer model fine-tuned on a natural-language inference data set. This paper is interesting as it is outperformed by h2oloo but it does something very similar. Because of this, we can assume which parts helped h2oloo work better. Mainly the first difference we see is that it used all of the previous reformatted queries as context while SBER used only the most similar one. This indicates that the previous queries have needed knowledge that helps the current turn give better answers. There is also the difference in the similarity measures being more complex in SBER, which might suggest a simpler one is better. They both concatenate only one sentence from the last turn retrieved passages so we can not compare is more better or worse.

USI [39] submitted 4 different runs. Out of the 4, three use query reformatting. The first using only current turn query, the second using the current turn query and last turns query plus most relevant passage, and the third one using the current query and three most relevant query-passage pairs from the history so far. This query-passage pair relevance is calculated using an ALBERT-based [22] model trained on CAsTUR [4], a data set containing labels which determine which of the previous queries in a conversation can be used to improve the current one. The downside of this is that it uses only raw queries (non-reformatted ones). The fourth approach used queries rewritten by GPT-2 [35] which were given by the organisers. Sadly the first three runs do not beat the baseline BERT model which indicates that more work needs to be done. The approach using three most relevant turns and the one using only the current turn query beat out the one using only previous turns query by a significant amount, meaning that the query from the previous turn is not always the one depended upon, which goes against what the TREC CAsT overview suggests (we will get to that in the next part of the related section).

HBKU [3] is an improvement on the TREC CAsT 2019. winner, historical query expansion model (HQE) [45]. HQE did query reformulation using topic and subtopic keywords extracted from previous queries. It would then feed them into a BM25 ranker and BERT reranker trained on MS MARCO. The HBKU paper went on to add terms from the most relevant returned passages. This step is a type of pseudo-relevance feedback (PRF) and they called it passage query expansion (PQE). The algorithms worked such that the HQE step would take place first, giving then the reformated queries to the ranked, BM25, which would return relevant passages. After this the PQE would take relevant terms from the top-k retrieved passages. Relevance was judged using a TF-IDF score (more on this in a later section of the related work). Top terms from the top-k passages were then added to the already reformatted query and then given to the BERT reranker. For their runs they took top 3 relevant words from top 3 relevant returned passages for PQE. 2 of their runs differentiate between queries that are ambiguous and those that are not, reformatting only those that are considered ambiguous. Unlike other

methods, this one uses passages returned in the same turn as the reformatting is happening and not in previous ones. Sadly, even though all 4 approaches beat the baseline, the one that did the best was the last one, that expanded only implicit queries and did not use PQE in during BERT reranking. This would show that adding terms from this turn's passages did not improve the results. It also shows that it is better to not change queries that are already not ambiguous, probably as we introduce possible errors by doing so. The authors of the paper believe that their approach of TF-IDF was too simple and that using something more complicated could have improved their results. We agree and think that it could have also been useful to try and do things such as use last turns relevant passages or less/more than 3 terms/passages during the PQE stage. Changing BERT for a T5 could have also helped improve results slightly. Sadly, as this approach by HBKU was tested on the 2020 TREC CAsT data set and the origian HQE on the 2019. one, we can not compare them to each other. We believe the approach used by h2oloo and SBER are a step in the right direction so we plan to test the h2oloo one out (because it reached better results) while however changing the similarity measure they used.

The 2020 TREC CaST track overview [10] gives a great view of what the state-of-the-art is. For starters, they indicate that 86% submitted papers used previous history as context and those that did not did about 31% worse on average. Sadly the data used is not perfect. Out of the 208 turns only 61 depended on previously returned responses while 140 depended on previous queries. This already gives an unfair advantage to methods that ignore the responses and focuse on the queries. There is also the problem of a small amount being "hard" dependencies. This means the query depends on more than only the last turn query or response. Only about 15% query and 8% response dependencies are "hard". This favours methods that only look at the last turn. It is more common that looking further back will cause data drift (adding unneeded data) than it is to help. This becomes even worse when we see that only 2% of dependencies are "hard" response dependencies. This explains why the method by h2oloo that uses only a sentence from the last turns most relevant passage outperforms their approach that takes a sentence from every previous turns most relevant passage. The overview points out that the papers that use the responses only slightly outperform those that do not. This would mean that the methods using responses have a long way to go before they can be considered usable in practice. This all raises the question of how good was the data set used in the first place. The data set was said to mimic real world interaction but had a very low degree of response dependency. This could mean that either the data did not do a good job in mimicking real-world scenarios or that in reality those dependencies are rare. The first raises a serious problem as it is difficult to create a large data set as funds are needed. The data being good on the other-hand could also mean that in real-life people really tend to rely mostly on the start of the conversation and the last part they talked about. This would mean methods should continue focusing their attention to these as they have so far while trying to maybe give minimal attention to the parts in between. [32] is an example of a paper that observed how their method works when looking at different depths (turns in the conversation). Their F1 score went down from 79.6 to 77.7 on the CoQA [37] data set and from 65.4 to 59.3 on the QuAC [13] data set when they did not use previous responses meaning the responses really were useful in getting a better score. The score actually fell less when previous queries were not used (from 79.6 to78.0 on CoQA and 65.4 to 64.7 on QuAC). They also showed that different data sets have a different preferred amount of previous queries and answers used. CoQA had little improvement when more turns were added but still showed some improvement. QuAC on the other hand improved by a lot (about 10) by using two turns but then the score started going down. The writers assume this is because of a more common topic change in the QuAC data set compared to CoQA.

2.2 Ranking/Reranking

After the query is reformatted so that it is self sufficient to show what data needs to be returned, we use ranking models to return the top n most relevant documents. After we have the top n, we extract the single most relevant document, this is called re-ranking. This is what papers tend to do but not all do, some have only a ranker and some have multiple rerankers [3]. The focus of the starting ranker is more on recall than precision as the aim of this step is to greatly narrow down the number of documents under consideration. The reranker on the other hand fouses on narroving down all the documents that are left and finding that one that is best to return. GRILL reaches best manual results in the 2020 TREC CAsT [10]. As the manual results use manually rewritten queries, therefore ignoring the query rewriting aspect of the task, we will assume that the best manual result actually has the best ranking and reranking. Because of this we will look at GRILL as the optimal approach and use it in our paper. The ranking is done using BM25 or a SDM [27] (Sequential Dependency Model) which return the top 1000 documents for the rewritten query, SDM giving better results. MonoBERT [29] performs bi-encoding of the query and document to give the relevance of the document to the query. The 1000 documents are then reranked according to the relevance scores. DuoBERT [30] is used as a second level of reranking. It is a pair-wise system where two documents are given to BERT alongside the query and separators. Due to the time complexity being $O(l_2)$ the paper decided to take only the top 10 most relevant documents as decided by monoBERT. DuoBERTs results are then compared to each other and normalized before taking the best one and finally returning that document to the user. A possible way to try and improve this is using T5 instead of monoBERT as it is shown to get better results, but the size of it could mean slower computation. Some other approaches include h2oloo's usage of tightly coupled teacher distillation [24] which incorporates dual encoders and spare representation from BM25. They use ColBERT [21] for their teacher model and duel encoders with BERT-base for their student model, both trained on the MS MARCO [7] passage ranking data set. Document reranking is done using a T5 model with pre-defined weights and fine tuned for paired (query, document) text relevance ranking, again on the MS MARCO data set. [42] used a simple BM25 ranker and a BERT-large reranker fine-tuned on the MARCO data set. The reranker is improved additionally by adding a reading comprehension model using RoBERTa-Large trained to predict an answer given a text. The score given by RoBERTA would be summed to the one given by BERT to give the final ranking. For some reason RoBERTA was used instead of T5 even though it was shown that T5 is the state-of-the-art for question answering. These are some of the approaches used so far concerning query reformulation and ranking/reranking. TREC CAsT 2020 shows how these and many other compare to each other. Among other methods there are a few that use answers to boost their performance, most commonly taking the top n relevant passages and extracting data from them. Sadly they all used different reanking/reranking methods so we can not judge their query reformulation methods to each other.

2.3 Alternative Approaches

Most papers use the query rewriting, retrieval and reranking approach, as that is now becoming the standard. [19] tried using entity linking instead. Their goal was to understand the effect of Entity information on a BERT-based retrieval model, a mix between monoBERT and E-BERT. They use entity information in an entity-enhanced BERT

model and apply it to rank documents, then perform pointwise re-ranking based on monoBERT. They inject entity embeddings along with BERT wordpiece embeddings to their BERT model using an entity linker. The approach did not give an improvement compared to other pipelines used in the TREC CAsT 2021 track. Nonetheless, they noticed that their entity-enriched model works better on entity-related documents rather than general news articles, which could lead to further research in that area.

2.4 Conversation History Summarization

2.4.1 Summarizing conversation history in the aspect of the current query

The main focus of this paper will be on using the answer passages given in previous turns and trying to extract knowledge that could improve this turn. We will be keeping the same approach of ranking/reranking throughout all the query reformulation methods we try so we can properly compare them. As stated above in the query reformulation part, there are some papers that focus mainly on previous questions like GRILL [14] and QuReTeC [44] and some that introduce using the answers alongside the questions such as h2oloo [25],SBER [8] and HBKU [3]. H2oloo and SBER use answers but not heavily. They extract sentences most similar to the queries. We plan to use multiple other methods to see whether better results can be reached. Some methods we plan to use include text summarization and similarity measures.

2.4.2 Text summarization

Automatic text summarization is the process of taking a large text and creating a smaller summary of it that keeps as much of the relevant data as possible [28, 5]. It is a non trivial task that started out in order to summarize large numbers of science papers. In the start simple approaches such as counting non-trivial words were used. State-of-the-art approach has been reached using the T5 model and we plan to use that one in the paper. It was trained on the non-anonymized version [38] of the CNN/Daily Mail data set [15]. As it is only one of the tasks the model is used for, text summarization is indicated with "TL;DR" (meaning "Too long, didn not read", which is a common abbreviation). Because of it is long character sequence, beam search [41] was used to improve the score. This is a sequence-learning approach that uses a multilayered Long Short-Term Memory(LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Improvements were seen in all metrics compared to methods so far especially when looking at the ROUGE-2-F score.

2.5 Similarity measures

We find similarity measures important to our work as they point to similar knowledge being talked about in different parts of the search system. In h2oloo's example this was the current turns query and the previous turns retrieved passages. We are looking at them separately to see could something have been done better and can we improve it in out own method. Similarity measures are functions that show how similar two sentences are [2]. The simplest similarity measure is word count, the more words two sentences have in common the more similar we say they are. This of course gives an advantage to long sentences so something like dividing by the summed number of words in both

sentences divided by 2 would be a simple way to remove that. The paper [2] tests 14 similarity measures split into three classes; word overlap, those based on term-frequency (TF) and linguistic measures. We will be testing out one method from each class that had the overall best score on all the tested data sets. Phrasal overlap [34] is a word overlap measure that takes into account phrase lengths and their occurrence rate. TF-IDF Vector similarity is a simple TF model that will turn our paragraph into a vector whose features consist of indexing words. Values are calculated using the term-frequency inverse-document-frequency method (TF-IDF) and similarity is then calculated using a cosine similarity between the values of two sentences. Identity is a subtype of this [17] and the score is instead derived from the sum of inverse document frequency of the words that appear in both sentences normalized by the overall lengths of the sentences and the relative frequency of a word between the two sentences. For linguistic measures we have a combination of word order (sim_{wo}) and a simplified variation of semantic similarity measure (sim_{sem}). sim_{wo} is defined as the normalized difference of word order between the two sentences while sim_{sem} is defined as determining sentence similarity based on the sum of maximum word similarity scores of words in the same part-of-speech class normalized by the sum of sentence's lengths. This paper focused mostly on similarity between sentences and not on larger texts which benefits us as we will be comparing the paragraph parts to queries which are commonly single sentence questions.

Another of the approaches we plan to try is using various types of entity information. Papers like [20, 16] try to use entity linking (EL) [40] to extract knowledge from the passages. The main focus of [20] was testing how well modern EL methods work in conversational search. They looked at multiple data sets and state-of-the-art methods. The results all-together were mediocre and showed how EL techniques need more research to be usable. This is also important for our paper as it shows how methods using the whole history beat out the ones using only the last turn on the CAsT data set. Sadly this research did not take into account returned passages and focused only on queries so we can not reach much conclusion there. [16] went a step further and tried to make a new model that would work better on conversational systems. They used a BERT-based model which they injected with entity information. To sum it up, they created a word-space using Wikipedia2Vec [18] and then, using a linear transformation, combined the embeddings into BERTs wordpiece vector space. This is used for the retrieval model. Query reformatting was avoided by using the manual approach, which meant gold standard or manually reformatted queries were used instead. The results were sadly worse than when using just BERT without injecting the embeddings because of the difficulty to perform entity linking for conversations [20] and the lack of their model's adaptation for the news article in the TREC CAsT document collection [16].

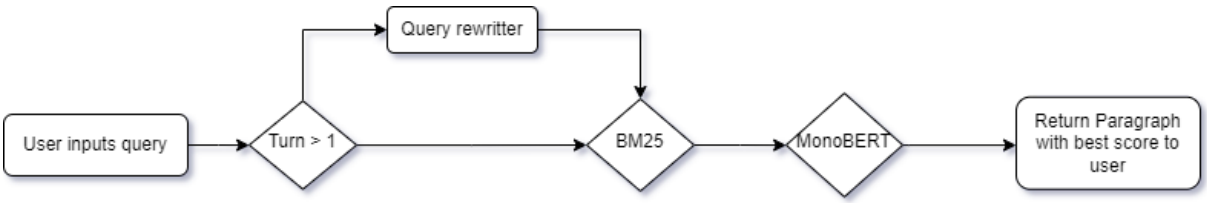

Figure 4: A diagram showing the high-view of the program cycle for one turn.

3 Approach

3.1 Task

The TREC Conversation search Task consists of user queries $[q_1, ..., q_{i-1}, q_i]$, which are answered by *passages* $[p_1, ..., p_{j-1}, p_j]$. Each cycle of query - passage pairs is called a *turn* (t). Conversation *depth* indicates the conversation length, or in other words, the number of turns in a conversation. Let q_i be the current user query. Let $[q_1, ..., q_{i-1}]$ and $[p_1, ..., p_{j-1}]$ together be the conversation history. Given the conversation history and q_i, the task of the retriever is to return a list of relevant passages D. The reranker takes D and reorders it so that the optimal passage is ranked first.
Query rewriting is the process of taking q_i and changing it so that it does not depend on the context from conversation history, therefore making it self-answerable.

3.2 Method Overview

We follow [25, 14, 8, 39] to rank relevant passages for each conversation turn using a pipeline consisting of a T5 query rewriter, BM25 retriever and a MonoT5 reranker.

The approach takes one conversation at a time and processes it turn by turn. The first turn of every conversation starts by taking the first raw query q_1 and giving it to the BM25 retrieval model. The model returns 1000 passages, which are then given to the MonoT5 [29] reranker. The reranker gives relevance scores to all the passages, and the one with the best score is returned to the user. From the second turn onward, the process gets an extra step; before the query is given to the retrieval model, it is given to a query rewriter. This is either the T5 model we trained ourselves or the one provided by the TREC CAsT organizers that is used as the 2021 TREC CAsT baseline. The query is given to the model alongside data collected from conversation history in different ways. The rewriter returns a query that does not depend on any context. The rest of the cycle is the same as the one for the first turn, the difference being the rewritten query is given to the retrieval model instead of the raw query. This process is the same through all our runs. The changes are made to the way in which we use conversation history, which varies from using raw or rewritten data, summarized passages and simple samplings.

3.3 Retriever

The retriever used in the experiments is BM25, which is the most used retriever in the TREC CAsT 2020 track, as well as the one used in the baseline model of the track. The exact implementation is search by Pyserini [3]. The parameters used are k1 = 0.9 and b = 0.4. We also tried k1=1.2, b=0.75, as these hyperparameters were used in the 2020 BM25 TREC CAsT baseline model, but it did not improve the scores.

[3]https://github.com/castorini/pyserini

3.4 Reranker

The reranker used is a MonoT5 implementation by Pygaggle [4]. Using T5 for reranking was inspired by the TREC CAsT 2021 track [11], which used it in its baseline runs. As our focus is not on the retriever and reranker, we chose two models that are easy to implement and have been shown to achieve good results in previous research.

3.5 Query Rewriter

Starting from the second turn of a conversation, the query is sent to the query rewriter before it is sent to the retriever. The query rewriter changes the query so that it does not depend on context (it can be self-answerable).

We use two different options for our model; the one given by the TREC CAsT 2020 as the baseline, which we refer to as the *baseline* model, and one we train ourselves following the guidelines given in h2oloo's paper [25], which is the main one we will be focusing on. We refer to it as *h2oloo*.

The baseline model is tested by participants of the TREC CAsT 2020 and the parameters are taken from castorini [5]. We are mainly using it to check the quality of our own model.

The *h2oloo* model is inspired by the h2oloo's paper [25] as it achieved one of the best results in the 2020 CAsT. The main difference between their model and the common approach that year is the usage of both the previous query and passage data when constructing the new query. Most other methods did not involve using previous passage data and just focused on the queries.

We say that the model is inspired by h2oloo instead of a reproduction of it due to a few limitations. Firstly, h2oloo's paper does not go into detail as to how they train their query rewriter, they only give a short explanation which is not enough for a good reproduction. The explanations for using previous passage data are also very general, mentioning no stemming or text processing, which we then assumed was used. We could not completely reproduce the process. Secondly, we do not use the same retriever as them, therefore the results could easily be influenced by that, as well as the lack of computational power on our end. We will nonetheless try to reach similar results to theirs.

We start by taking a pre-trained version of T5-base called simplet5 [6]. We then follow the accompanying google colab code to train the model further for query rewriting. When training the T5 model following h2oloo's method, we use training data of 10000 samples and a validation of a 1000 samples. Parameters were set to:

$$source\_max\_token\_len = 400$$

$$target\_max\_token\_len = 50$$

$$batch\_size = 8$$

$$maximum\_epoch = 10$$

Even though the maximum epoch is set to 10, the evaluation loss does not go down past the second epoch, therefore the second epoch is used during inference. The parameters are set by us because they were not given in the h2oloo paper.

Due to the limited amount of resources, we focus on reproducing the results reached by h2oloo using the T5-base model, ignoring the ones reached by the T5-large model, assuming our model would likewise simply reach slightly better results if trained on the

[4] https://github.com/castorini/pygaggle

[5] https://huggingface.co/castorini/t5-base-canard/tree/main

[6] https://github.com/Shivanandroy/simpleT5

large version. The data used for training the model is the CANARD dataset [7]. During training, the data is given to the model in the form of an input, called the source text, and an output called the target text. The source text contains a list of all previous queries in addition to the current query concatenated with a " ||| " in between. This is not found in detail in the h2oloo paper [25] but is instead assumed after looking at the usage of the TREC CAsT baseline castorini model [8]. The target text is a sentence denoting how the rewritten query should look. One source text and one target text are called one sample.

The T5 model is then added to our pipeline. We use different ways of adding information from the conversation history to the query rewriting process. To start off, we test the methods that h2oloo used to see if the model reaches the same results. We take the most relevant passage from the previous turn and go through all of its sentences. Each sentence has all its words stemmed and then removed if they are not a noun, verb or adjective. The raw query is processed the same way. The processed sentence from the passage shown to have the most word overlap with the processed raw query are considered the most similar. If none of the sentences have any word overlap then no sentence is taken and an empty string is passed along instead. Spacy [9] and NLTK [10] libraries are used for the processing of sentences. The five methods mentioned in the h2oloo paper are:

- Query only - all previous raw queries
- Naive - all previous raw queries and all passages
- Type a - all previous raw queries, all similar sentences so far and a new similar sentence form the previous passages
- Type b - all previous raw queries in addition to a similar sentence form the previous passages
- Recursive - all previous rewritten queries as well as a similar sentence from the previous passages

Out of the mentioned methods, the paper does not give results for the Naive method and we assume they did not run it, due to the very large amount of tokens it would have had, easily going over the maximum of 512 tokens allowed. A better visualisation can be seen in figure 5 taken from h2oloo's paper. The query only method is not shown in the mentioned image as is not talked about extensively in the paper but the scores were mentioned.

3.6 Summarization methods

The methods explained are taken from h2oloo's paper with some improvising on our end due to the process not being explained in great details. We saw earlier in Figure 1 that the curve of the manual runs shows only a small drop in performance as the conversation depth increases, while the automatic and canonical runs performance significantly decreases as the conversation goes on. The goal of our second research question is to see if the score on later turns could be improved by changing the way we use data in the query rewriting process.

[7] https://paperswithcode.com/dataset/canard
[8] https://huggingface.co/castorini/t5-base-canard/tree/main
[9] https://spacy.io/
[10] https://www.nltk.org/

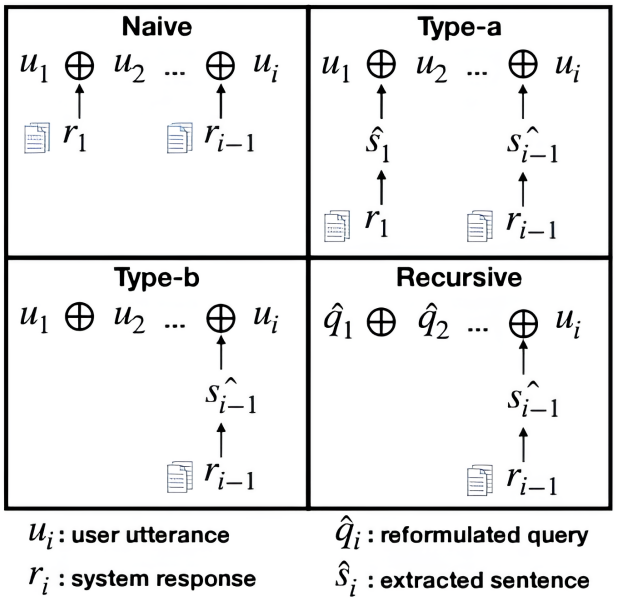

Figure 5: The image, taken from [25], show the four different data that h2oloo used for query rewriting. Some different terminology is used compared to our paper; utterance is another name for "raw query", while system response is the passage returned to the user.

The methods h2oloo reported are different from the other competitors because they use a most similar sentence from the previous paragraph for query rewriting. Looking at their process we see that the similarity measure used is very simple.

H2oloo used a simple word count that gives the amount of same words in each sentence in the paragraph compared to the current query. We think hypothesise this simple approach can be improved by applying some changes to it.

The approach we focus on uses a text summarizer to make the paragraph we give to the query rewriter smaller and more concise. The previous goal of taking the most similar sentence is to take the most useful data from the passage and give it to the rewriter. A better approach could be summarizing the paragraph into one sentence, hoping the summarized text will have a higher density of relevant data. We try a few different variations, mainly focusing on different lengths of summaries.

We use the same T5-base from simpleT5 [11] and fit it for text summarization. For training data we use a simple newspaper headline dataset that goes with the simpleT5 colab notebook. The data is organised into sample and target text (x and y). Sample text is the news article with a pretext, *summarize*, as shown in the original T5 paper [36]. Target text is the headline paired with the article. The goal of the training is for the model to learn to reach the headline when given the news article. 10000 samples are used for training and 1000 for validation.

The parameters are similar to training the query rewriter:

$source\_max\_token\_len = 512$

$target\_max\_token\_len = 50$

[11] https://github.com/Shivanandroy/simpleT5

Cond.	Query reformulation		Retrieval (dense+sparse)			Re-ranking (T5-3B)		BLEU	Run
	Model(T5)	Inference	R@1000	MAP	NDCG@3	MAP	NDCG@3		
Manual	-	-	0.840	0.324	0.463	0.459	0.613	100.00	-
1	base	Query-only	0.668	0.225	0.343	0.330	0.452	63.75	Run4
2	base	Type-b	0.661	0.216	0.337	-	-	63.12	-
3	base	Recursive	0.684	0.220	0.328	-	-	62.18	-
4	large	Query-only	0.696	0.238	0.360	-	-	64.33	-
5	large	Type-a	0.708	0.239	0.364	-	-	64.43	-
6	large	Type-b	0.697	0.238	0.358	0.345	0.480	64.64	-
7	large	Recursive	**0.724**	**0.250**	**0.367**	**0.363**	**0.494**	**65.23**	Run2

Figure 6: The table is taken from [25] and displays their results. We can see that their best performing method when using T5-base is Query-only with a BLEU score 63.75, while the best T5-large run is Recursive with a BLEU score of 65.23. Certain numbers were highlighted by us in blue and yellow and were not in the original paper.

$batch\_size = 8$

$maximum\_epoch = 10$

The input to the summarization model varies for each method, but it is always a variation of previous best passage data. The output is always a chunk of text whose size varied depending on the run. This chunk of text is then given to the query rewriter alongside the raw query. Next we will explain some of the variations used. We use the term short when the summary is between 2 and 10 words, medium when it is between 5 and 30 words, and long for summaries between 10 and 50 words. We test the following setup:

- One short/medium summarized passage

  Takes the previously returned passage and summarizes it to short or medium length.

- All passages summarized to short/medium texts and then combined

  Takes each of the previously returned passages, summarizes it to either short or medium size, and then concatenates the results.

- All passages combined and then summarized to short/medium/large texts

  Takes all of the previously returned passages, concatenates them and then summarizes them either to a short, medium, or large size.

4 Evaluation

4.1 Data

We use the data from the TREC Conversational Assistance Track CAsT 2020. The focus of the TREC CAsT 2020 was using a dataset where q_i depends on previous queries and previous passages. We chose this one specifically because the goal of our paper is to show how conversation history and conversation length influences conversation retrieval.

The collection from which the passages are returned consist of the MS MARCO passage collection[12] [7] and the Wikipedia collection[13] from the TREC CAR paragraph collection V2.0. The TREC CAR collection is transferred into a .json format for easier usage, while the MS MARCO collection is already in that format, therefore needing no preprocessing. Both collections were then combined to create an index file. This index is used for the BM25 model for passage retrieval.

For evaluation, we use the evaluation data provided by the TREC CAsT 2020 called *2020_manual_evaluation_topics_v1.0.json*. It consists of 25 conversations ranging from 6 to 13 turns of conversation depth; shown in figure 7. When calculating the overall score of a method, we use all turns, but when showing the scores in graphs, we show only the ones for the first 10 turns following the experiments from the CAsT overview paper [10]. Each turn of the evaluation data has a raw query, a manually rewritten query, and an Id. The manually rewritten query shows how the raw query should look after it has been changed to be self-answerable, which is useful when calculating the BLEU and Rouge scores used for evaluating the query rewriter. The id is the identifier of an example ideal passage found in the corpus to be returned for the given query.

We also use the *qrel* file. This is the main evaluation file for retrieval and reranking in the TREC CAsT 2020. All the groups taking part in the research sent in their top 1000 documents for every turn in every conversation. The TREC organizers took the top n documents from every run and annotated them manually, adding a relevance scale of 0 to 4, showing how relevant the passage is to the given query. These scores would later be used to calculate the NDCG@3 score, which is the main score used for evaluating how well the models perform. The official retrieval scores in the track are calculated using the qrel file, as well as our own.

We use the CANARD dataset[14] [12] for training the query rewriting T5 model. The dataset is made for question-in-context rewriting and consists of conversations with different depths, where each turn has a raw query, the passage returned in the turn before, and the gold standard rewritten query.

4.2 Evaluation Metrics

The evaluation methods used are NDCG@3, BLEU, and ROUGE. NDCG@3 which looks at the top 3 retrieved documents is considered when evaluating the retriever and reranker, while BLEU and ROUGE is used for evaluating the query rewriting. NDCG@3 is chosen because it is the official metric used in TREC CAsT track as well as the h2oloo paper. BLEU is chosen as it is being used in the h2oloo paper and allows us to compare our results to theirs. We decided to use the ROUGE score following [1], who used it to measure the performance of query rewriting. Same as in their paper, we used the *ROUGE-l* version, which focuses on the *Longest Common Subsequence* (LCS for short) instead of the *ROUGE-n* as ROUGE-n focuses on n-grams same as the BLEU score.

[12] https://msmarco.blob.core.windows.net/msmarcoranking/collection.tar.gz
[13] http://trec-car.cs.unh.edu/datareleases/v2.0/paragraphCorpus.v2.0.tar.xz
[14] https://paperswithcode.com/dataset/canard

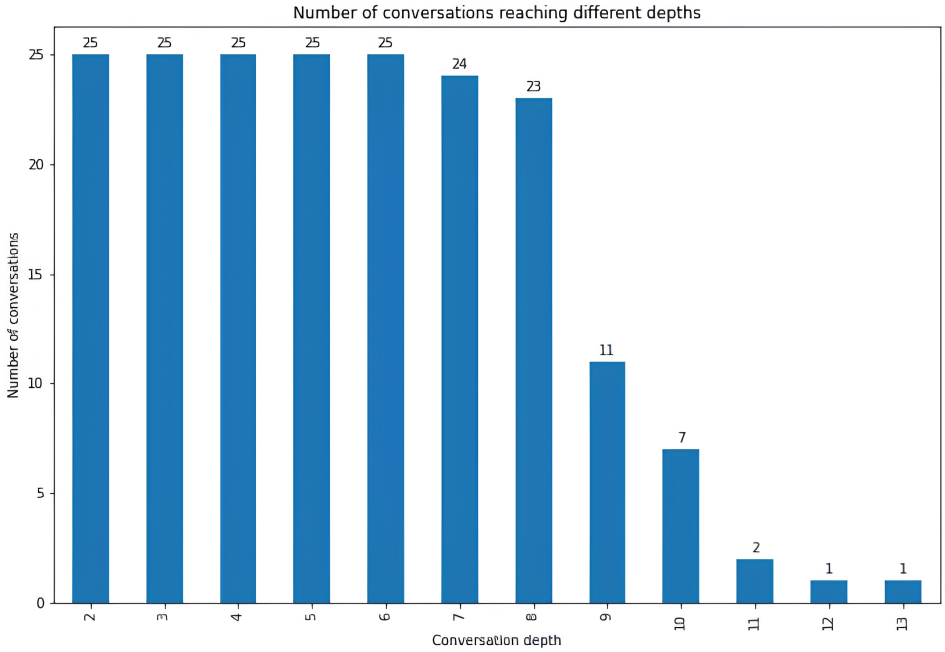

Figure 7: A graph showing how many conversations reach different turn depths. It is noticeable that there is a sharp decline at turn 9, which means less than half of the conversations reach that far. This leads to the data being less dependable when looking at the model's quality at turn 9 and 10.

We use the official trec_eval library [15] for NDCG@3, the 0.3 package *bleu* [16] for BLEU, and the 1.1 py-rouge [17] implementation for ROUGE.

To calculate NDCG@3, we run the code all the way through and save the scores the retriever gives us, which would be a value of how good the top 1000 documents BM25 returned earlier are. We then give this txt document to the trec eval script and get the NDCG@3 scores.

The mentioned BLEU implementation is a python version of the original BLEU metric [33]. Each turn is calculated by itself and an average of all of them is taken when giving the final results. When calculating scores for turn depths all turns of those depths are taken and averaged over. Meaning that when calculating the average BLEU score for turn 3, we will take all the BLEU scores calculated for every conversation at turn 3 and then calculate their average. This means all conversation depths are treated equally, although larger depths (e.g. 10) are reached less frequently than others. We notice that for 5 conversations on turn 1, the raw query and the manually rewritten query (the golden standard for how the rewritten query is supposed to look) do not match. This is not expected as query rewriting is not applied on turn 1 therefore these two should always be identical. We leave it as is as the difference in results is minor.

[15] https://github.com/usnistgov/trec$_eval$

[16] https://github.com/zhijing-jin/bleu

[17] https://pypi.org/project/py-rouge/

4.3 Experimental setup

The runs are split into three main categories; manual, canonical and automatic. The manual run skips the query rewriting step, using ground truth rewritten queries instead. This gives the best possible NDCG@3 score and has three purposes. Firstly, it showes whether or not our retriever/reranker reaches the results that TREC CAsT 2020 submissions did on average, and serves as a sanity-check for our experiments. Secondly, it will show whether performance really does minimally drop as the conversation goes on when the optimal query is used, as is shown in figure 1. Third, it is used as a goal to reach when using *h2oloo* query rewriter instead of the manual gold standard query.

Canonical testing uses ground truth retrieved passages, meaning that the pipeline is given the optimal passages using the ids from the evaluation data. On turns after the first, the previous passage is sent to the query rewriter. This way the rewriting does not depend on the quality of the data returned by the retriever or the reranker. These runs are therefore a great way to measure our query rewriter performance. We can also compare them to the standard runs to get an even better idea of how large the impact of adding previous conversation history to our model is.

Standard automatic runs use no ground truth data. They run the full pipeline; retriever, reranker and query rewriter, using raw data. These runs are our main focus when comparing our methods to each other as they show how good our methods would be in a real world scenario. They will be compared to the h2oloo and TREC CAsT overview paper results as those are the main focus there too.

Table 3: Table showing NDCG@3 scores of some of the main methods we discuss in our paper; our best run using both the query rewriter we trained ourselves and the one we took from TREC CAsT 2021 baseline, h2oloo's best run as well as the baseline model score in the 2020 TREC CAsT.

Method	NDCG@3
Query-only (h2oloo paper [25]	0.452
Query-only (our using TREC CAsT 2021 query rewriter)	0.3939
Query-only (our h2oloo inspired)	0.3494
TREC CAsT - 2020 baseline	0.30

5 Results

Our first goal is creating a pipeline that works on par with the rest of the TREC CAsT 2020 submissions, so that we can further use it to reach conclusions.

We start this by mimicking the query rewriting process used in the h2oloo paper [25]. We compare our results to the models that h2oloo reached using the T5 model. Due to a lack of computational power, we mainly look at the results they reached using T5-*base*. We assume that the model would have given slightly better results if trained using T5-*large* because we saw the same trend in h2oloo's paper [25] as well as the original T5 paper [36]. In Table 3 we see the results the h2oloo [25] paper produced as well as two of our results; using a query rewriter we trained ourselves, and the one using the TREC CAsT 2021 baseline query rewriter.

The table shows that when looking at the methods run on the T5 query rewriter we trained ourselves, the *query-only* method reaches good results. Had it been submitted in the TREC CAsT 2020 with that score it would have placed us in 7th place which we decided is good enough. We also see that it outperforms the TREC CAsT 2020 baseline method by about 0.0494 while losing to the h2oloo method by about 0.1, which was in fact the method with the highest NDCG@3 score that year. On the other hand, the best method when we ran the pipeline using the TREC CAsT 2021 baseline query rewriter is *Query only*, which reached a score of 0.3939. This loses to h2oloo's best method by 0.053, and is 0.093 better than the TREC CAsT 2020 baseline.

In Table 4 we see the BLEU score comparison of our runs with the runs in the h2oloo paper. Our runs do not manage to reach the same BLEU scores as the h2oloo paper. We can also see that the canonical scores are only a slight improvement compared to our automatic scores, which shows that the problem is the query rewriter and not the retriever and reranker (as the optimal passage is used).

We decided that these scores are good enough to go forward with the analysis of the data using the mentioned methods as they outperform most of the submissions in the TREC CAsT 2020 overview paper [10].

5.1 H2oloo-inspired method run score analysis

5.1.1 Analysis

In this section, we mainly look at three important graphs: NDCG@3 (Figure 8), BLEU (Figure 9) and ROUGE (Figure 10) score graphs. Each shows the change in scores for a different metric across different conversation depths. We took the metric scores of the four methods we copied from h2oloo, and then average them at each turn to get a good visualisation of how their methods scores change as conversation length increases.

Table 4: Results of the four methods used in the h2oloo paper. We show the BLEU score they reported as well as the results from our own automatic and canonical runs of the same methods. Our runs reach a noticeably lower score than the ones h2oloo reported.

Method	h2oloo	Our Automatic	Our Canonical
Recursive	62.18	51.93	52.63
Type-b	63.12	52.67	53.31
Query only	63.75	51.77	51.77
Type-a	—	51.71	52.32

The focus is on using the query rewriter we trained ourselves but we also mention other scores.

When analysing the trend of the curves we see some simliarity and differences between Figure 8 and Figure 1. The automatic curve looks almost exactly the same up to turn 6. At turns 7 and 9 our methods have a dip upwards, while theirs have a dip downwards. The canonical line is less similar but we can still see the same trend. Our canonical line falls down on turns 4, 5 and 10 which does not happen in the TREC CAsT overview paper. The problem with this analysis is that a part of the difference happens in the last two turns, 9 and 10, where we have a lot less data (11 and 7 conversations respectively, instead of the usual 25). **Looking at the data, we do not clearly find any trend that the performance drops by increasing Conversation turns**

When analysing the query rewriting scores, it is more proper to look at the BLEU and ROUGE score curves. Neither TREC CAsT overview nor the h2oloo paper have curves for showing their BLEU and ROUGE scores, which means we could not compare this part to their papers. In Figure 9 we see the BLEU score of the mentioned methods. The turns at which the scores reach the best value are actually turns 6 and 8, both for the automatic and canonical runs. We see that the score drops at turns 2 and 3, then slowly gets better. At turns 9 and 10 it drops down again. In Figure 10 we see the ROUGE score. Here, the same as with the BLEU score, we see a drop at turns 2 and 3, then an increase. From turns 4 to 8 the curve is almost flat, meaning the score does not change much. At turn 9 we again see a drop but at turn 10 we see it go up. All together when looking at the BLEU and ROUGE scores we see a very similar trend. A drop happens early on, then the scores get better before going down again late in the conversation. For both metrics, the scores do not vary much at different turn depths, meaning that even when we have the curve going up and down, it is not by much. **We can not see any trends that support the hypothesis when looking at the BLEU and ROUGE metrics**

The three graphs do not show strong evidence to support the hypothesis of conversation length influencing the quality of the model. NDCG seems to fall down at turns 2 and 4 while BLEU and ROUGE dip down at turns 2 and 3, which is early on. The fact that they get better around the middle of the conversation goes against the hypothesis. We, however, emphasize that fluctuations of scores for turns 9 and 10 should be considered carefully, because of a lack of data.

5.1.2 Discussion

We believe we can give an explanation to the curve going up and down by taking a more detailed look at the methods we use. We use conversation history to improve the

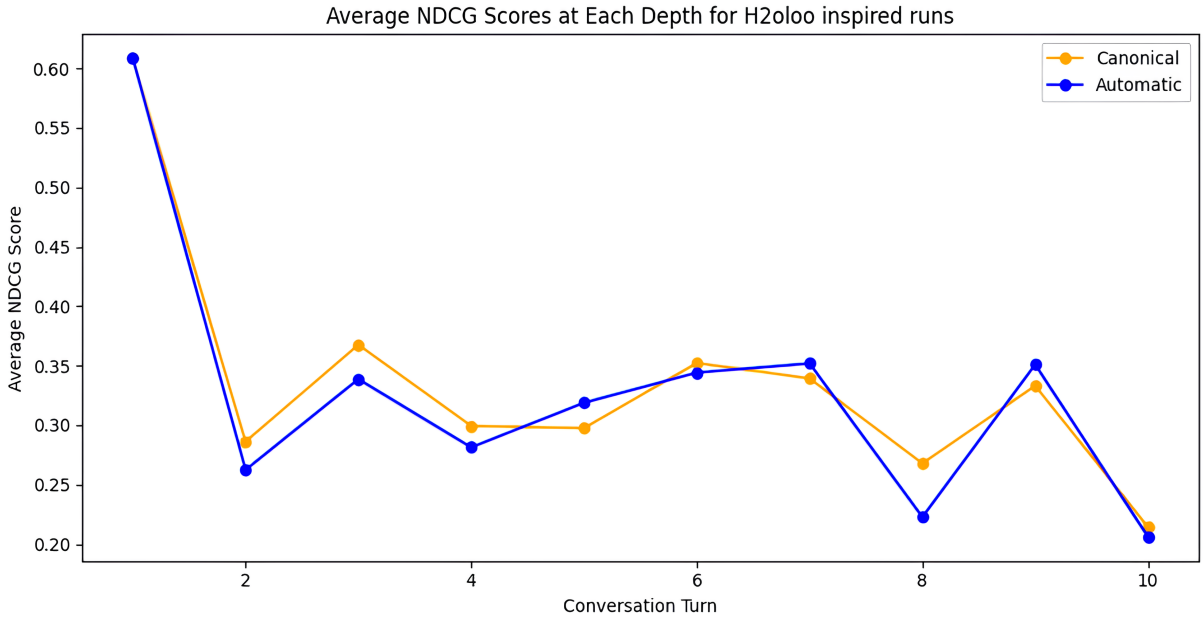

Figure 8: A graph representing the average NDCG scores for the four methods we implemented (following h2oloo [25]), calculated separately at different turns. The blue line represents the automatic runs, while the yellow line represents the canonical ones.

query rewriting part of our pipeline, meaning that the longer a conversation goes on following the exact same topic, the better the results should be, due to our model using data from previous turns and therefore getting better at that specific topic. This would explain the improvement of scores in the middle of the conversation, as the longer we follow the same topic, the better the model is at returning passages connected to it. On the other hand, this makes the model more sensitive to minor changes to the subtopic. This means that the model focuses more and more on the exact details we are talking about, and therefore will get confused over slight changes in the topic. We believe that these changes in subtopic in later turns are to the reason for the lower scores at turn 9 and 10. This is called *topic shift* and is actually a known problem.

5.2 Improving results using summarization

5.2.1 Analysis

In order to test and improve this, we tried using summarisation instead of similarity.

The hypothesis that runs using less data will outperform the ones using more data in later turns of a conversation, due to the model being less confused by changes in the topic. This would prove that the problem is not the length of the conversation but the methods used for calculating the scores.

Another hypothesis is that those runs using only the previous turn data will also work better as the conversation length increases. This would also prove that the problem is a shift in topics, as methods using less data from previous turns should be hindered less when faced with a change in topics.

We will again be looking at three important graphs; NDCG@3 (Figure 11), BLEU (Figure 12) and ROUGE (Figure 13) score graphs, constructed in the same manner explained in the previous section. Instead of using the h2oloo inspired runs, which

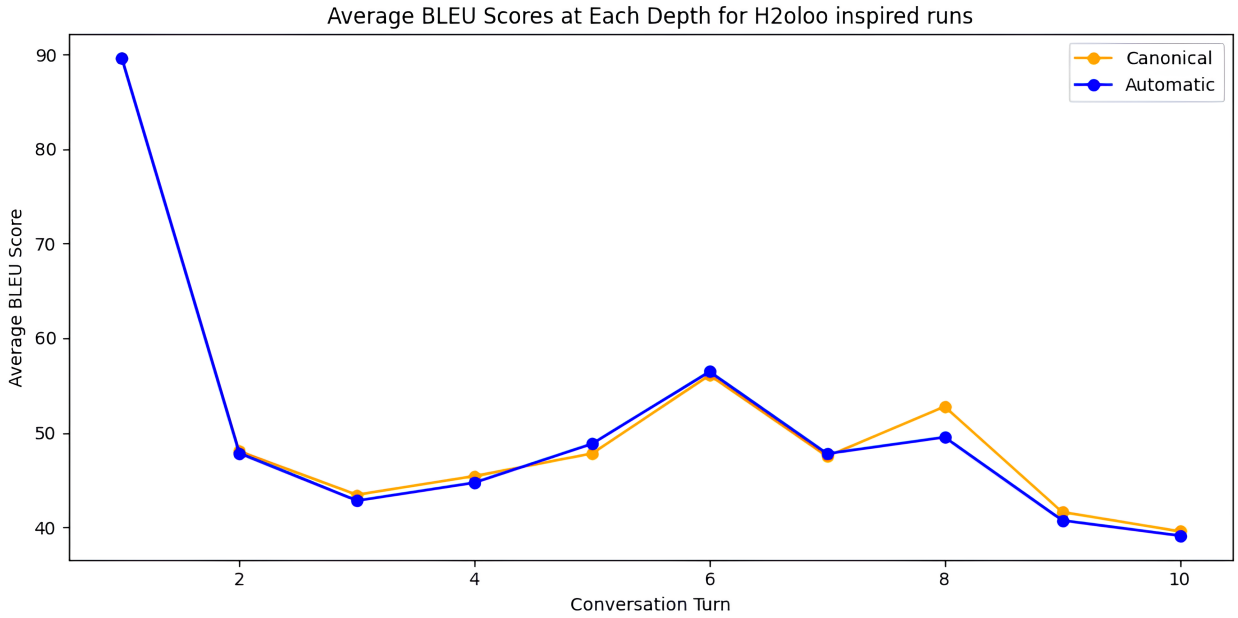

Figure 9: A graph showing the BLEU scores of the 4 methods we ran (copied from h2oloo [25]), averaged over at different turns. The blue line represents the automatic runs, while the yellow line represents the canonical ones.

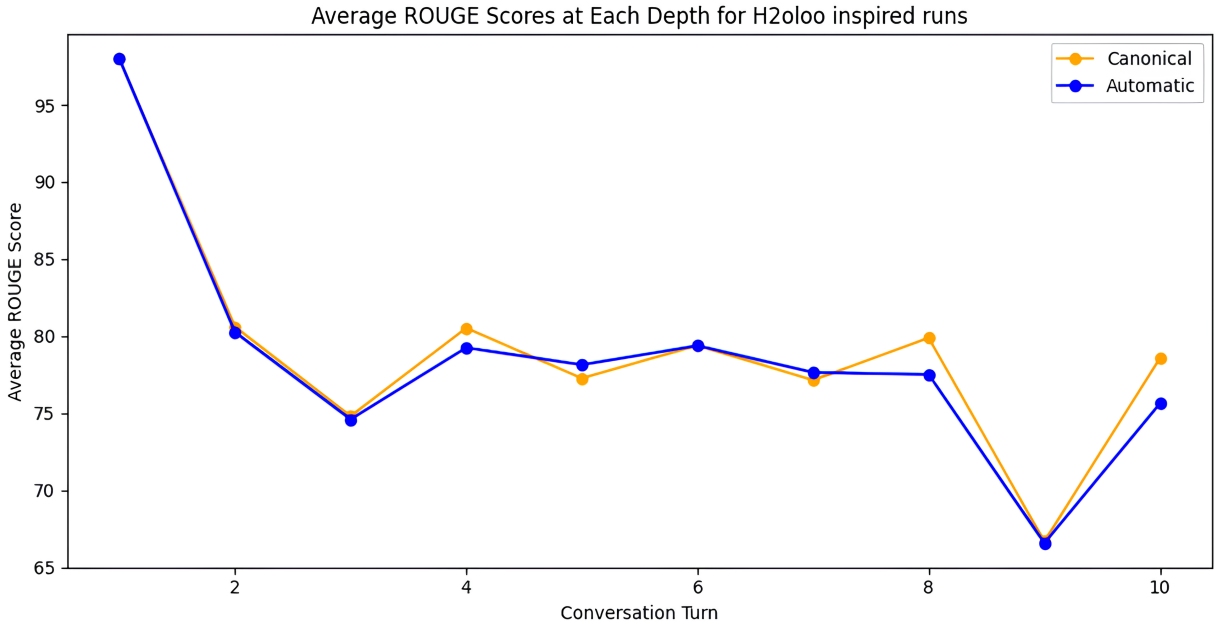

Figure 10: A graph showing the ROUGE scores of the 4 methods we ran (copied from h2oloo [25]), averaged over at different turns. The blue line represents the automatic runs, while the yellow line represents the canonical ones.

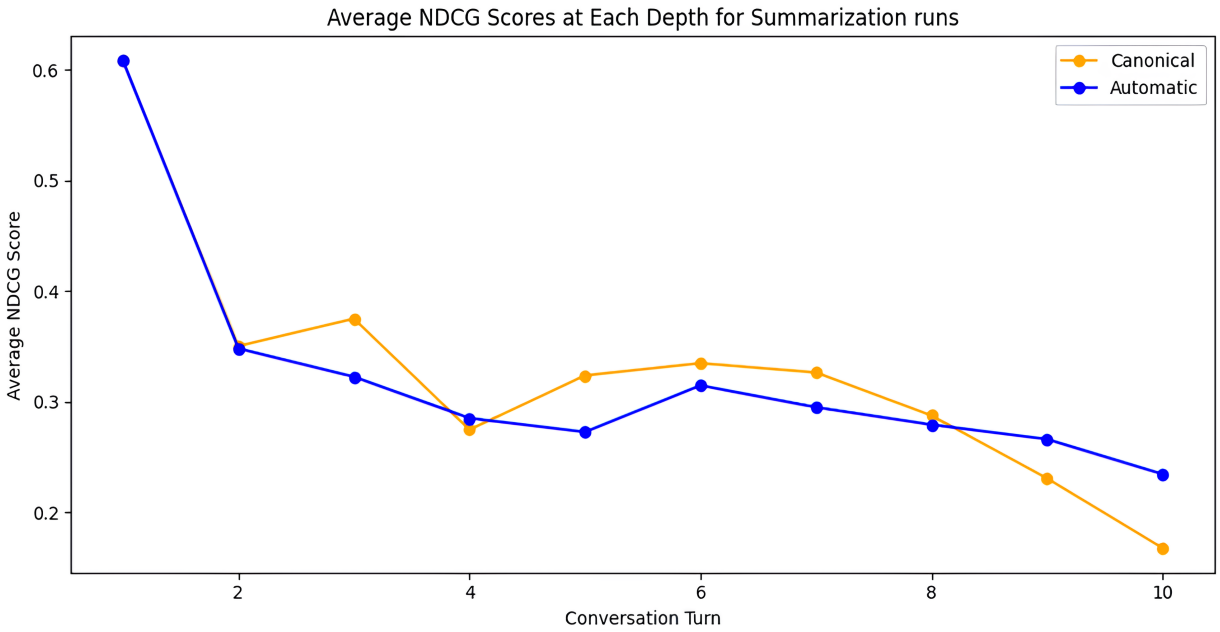

Figure 11: A graph showing the NDCG scores of the methods we ran using summarization, averaged over at different turns. The blue line represents the automatic runs, while the yellow line represents the canonical ones.

use sentence similarity, we will now be averaging over methods using summarization to collect previous turn paragraph data. We will again be focusing on runs using our own query rewriter.

We see in Figure 11 a similar logic to the h2oloo methods Figure 8. The NDCG@3 scores seem to fall down after turn 1, here falling till turn 5, after which they rise up and start falling again. The scores fluctuate less, and the whole curve is similar to a straight line. The main differences between the two figures are the fluctuations the h2oloo method figure has, upward at turn 3, and downward at turns 8 and 10. The BLEU score (shown in Figure 12) and ROUGE scores (Figure 13) are almost identical to the earlier h2oloo BLEU and ROUGE score Figures 9 10. Only difference is the increase in BLEU score at turn 10 in the summarization BLEU score Figure. The BLEU and ROUGE graphs did not show any new information or signs of improvement. The NDCG@3 score also does not improve, or does so very slightly. On the other hand, we could look at the straightness of the curve of the NDCG score as a type of improvement, as it shows the methods are more stable when working with different turn lengths. We see that the scores lower less at later turns. **We do not see a significant different in scores when adding summarization to our methods.**

5.2.2 Discussion

In Tables 5 and 6 we will now compare the results of the summarization methods to those of the h2oloo methods for runs using both our query rewriter and the 2021 TREC CAsT baseline model.

Table 5 shows that the one short summarized method reaches the best scores out of all our runs when using the query rewriter we trained ourselves, meaning that our usage of summarization helped improve the scores slightly. The second best method is Query only, which was also the best method for h2oloo when they used T5-base. This

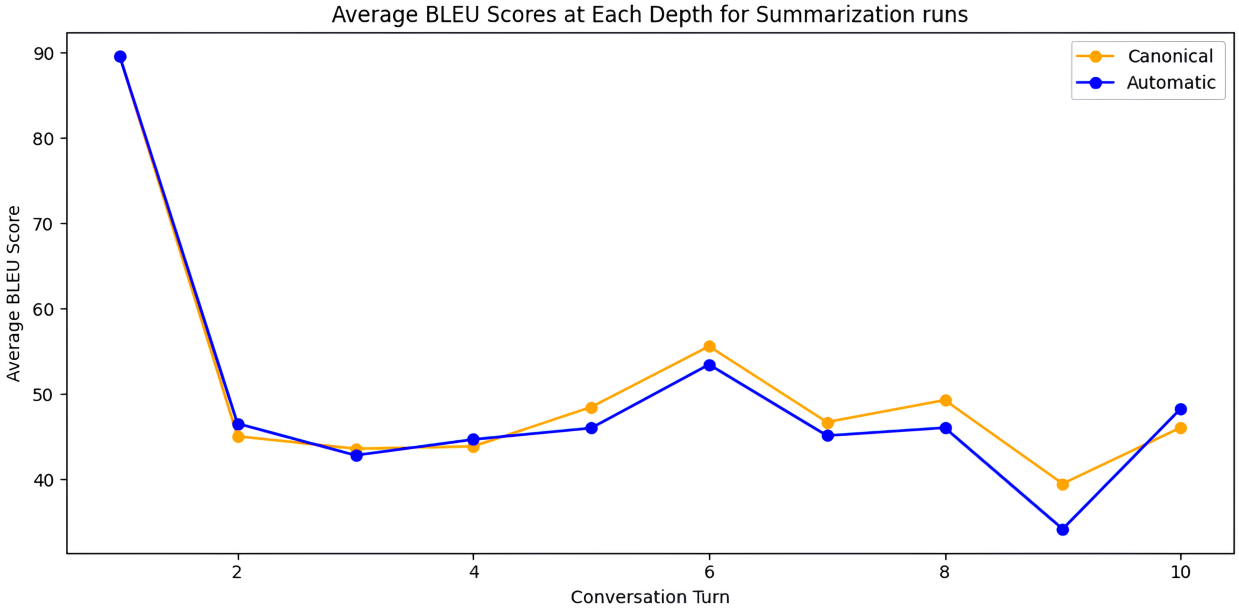

Figure 12: A graph showing the BLEU scores of the methods we ran using summarization, averaged over at different turns. The blue line represents the automatic runs, while the yellow line represents the canonical ones.

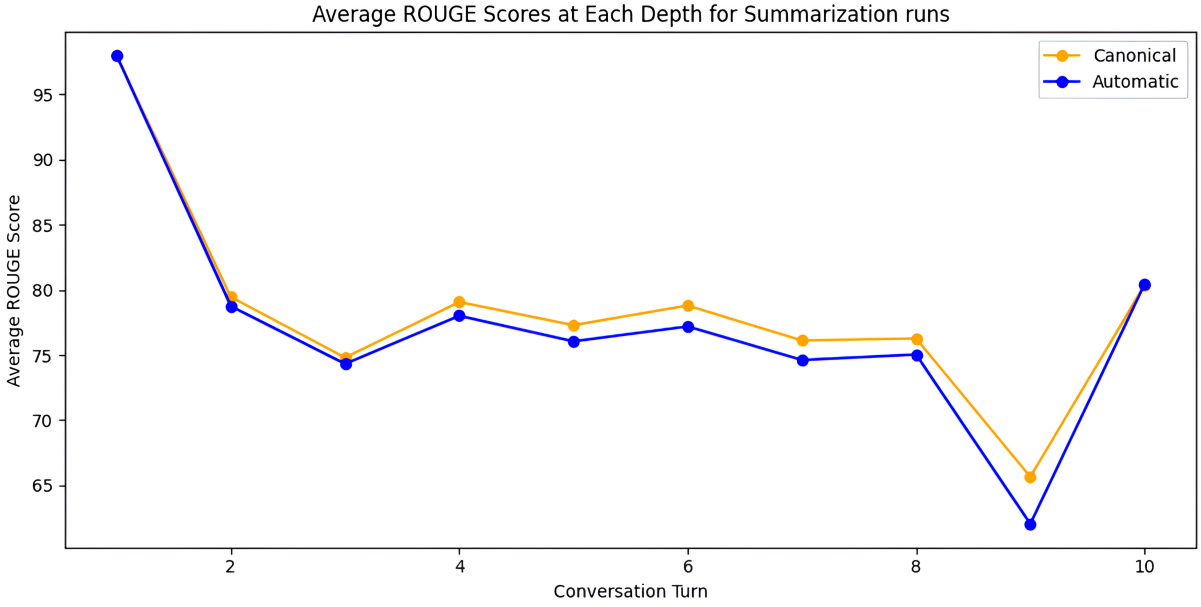

Figure 13: A graph showing the ROUGE scores of the methods we ran using summarization, averaged over at different turns. The blue line represents the automatic runs, while the yellow line represents the canonical ones.

Table 5: Table showing runs ordered by NDCG score when all the methods are run using the T5-base query rewriter we trained ourselves.

Data usage type	NDCG@3	BLEU
one short summarized paragraph	0.3596	50.63
query only	0.3494	51.77
all passages summarized short then combined	0.3464	51
type b	0.3418	52.67
all rewritten queries	0.3333	52.13
all paragraphs combined then summarized mid	0.3324	50.37
Recursive	0.3306	51.92
one mid summarized paragraph	0.3305	50.52
all paragraphs combined then summarized long	0.3298	51.61
Type a	0.3274	51.71
all paragraphs combined then summarized short	0.3251	49.39
all paragraphs summarized mid then combined	0.3177	48.77
all rewritten queries all similar sentences	0.3063	51.43

would mean that the methods using conversation history data hinder the score instead of helping it. In general, we see that our summarization methods perform similarly to the h2oloo methods, sometimes getting better scores, but never a significant improvement, and we therefore can not really say that they help the model work better when using our query rewriter. Another concern is that the metrics do not follow each other, namely the BLEU scores are sometimes actually better for the lower NDCG@3 scores shown in Table 5.

Table 6 shows the Query Only method reaches the best scores when we switch to using the TREC CAsT 2021 baseline query rewriter, again reinforcing the idea that the extra information only lowers the scores and is not something that should be used in this way. It reaches a score of 0.3939, which is 0.0457 better than when using our query rewriter. We can also see in the table that the methods we copied from h2oloo outperform the summarization variations we ran ourselves. The NDCG@3 metric follows the BLEU score metric properly here, meaning the runs with the best NDCG score also have the best BLEU score, and vise-versa.

The different order of the methods when switching query rewriters is unusual. This leads us to believe that the query rewriting methods used by h2oloo work better the more the query rewriter is trained, as the TREC CAsT 2021 baseline is trained on more data than our model is.

Nonetheless, we see that using summarization does not increase the general average scores of the methods used by h2oloo. We can on the other hand say that it helps stabilize the scores, as the Figure 11 has a much straighter curve than Figure 8, which means summarization could be a good potential way to create a process that works optimally on different turn depths.

Table 6: Table showing runs ordered by NDCG score when all the methods are run using the T5 query rewriter used as the baseline for TREC CAsT 2021.

Data usage type	NDCG@3	BLEU
query only	0.3939	52.67
type b	0.3899	52.62
all rewritten queries	0.3854	52.21
Recursive	0.3804	52.21
Type a	0.3649	52.46
all rewritten queries all similar sentences	0.3622	52.52
all passages combined then summarized long	0.3548	50.57
one short summarized paragraph	0.3524	49.89
one mid summarized paragraph	0.3492	50.39
all paragraphs summarized mid then combined	0.3477	48.77
all passages combined then summarized short	0.3443	49.74
all paragraphs combined then summarized mid	0.3405	49.05
all passages summarized short then combined	0.3251	49.66

6 Conclusion

In this thesis we have focused on exploring the influence of conversation length on retrieval scores in conversational systems. The TREC CAsT overview paper [10] showed the increase in turn depth lowers the scores. By making our own pipeline, inspired by h2oloo [25] and other papers from the 2020 competition, we created our own pipeline to test the hypothesis.

Looking at the graphs and data the h2oloo method runs gave us, we concluded that we can not say that conversation length influences a drop in retrieval quality. The scores seen in Figure 8 show an increase from turn 2 to turn 6, which would indicate the methods work better the longer the conversation is, but the scores also drop down from then to turn 10, which would indicate the opposite.

We claim that this is due to our models learning the conversation data over time and therefore giving better and better results, until a topic shift happens later in the conversation which confuses the model and makes it give lower scores.

Using summarization instead of sentence similarity did not yield an improvement in scores. The BLEU and ROUGE scores were almost identical to the h2oloo methods, while the NDCG score had slight variations. We also see the curve is more straight when compared to the h2oloo method graph, meaning the methods are more stable and better suited for working on longer conversation.

To sum up, we can not find any evidence supporting the hypothesis. We believe that a new dataset should be created which focuses on more strongly keeping to the same topic, with no shifting, to be able to run research and reach a valid and strong conclusion.

We also want to emphasise the stability of the NDCG curve when using summariation methods and point out that more research should be done in using summarization for helping retrieval methods using conversational history.

References

[1] A comparison of question rewriting methods for conversational passage retrieval. 1 2021.

[2] Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. The evaluation of sentence similarity measures. In Il-Yeol Song, Johann Eder, and Tho Manh Nguyen, editors, *Data Warehousing and Knowledge Discovery*, pages 305–316, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[3] Haya Al-Thani, Bernard Jim Jansen, and T. Elsayed. Hbku at trec 2020: Conversational multi-stage retrieval with pseudo-relevance feedback. In *TREC*, 2020.

[4] Mohammad Aliannejadi, Manajit Chakraborty, Esteban Andrés Ríssola, and Fabio Crestani. Harnessing evolution of multi-turn conversations for effective answer retrieval cod-an intelligent knowledge sharing platform view project contextual information on point-of-interest recommendation view project harnessing evolution of multi-turn conversations for effective answer retrieval.

[5] Mehdi Allahyari, Seyed Amin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys J. Kochut. Text summarization techniques: A brief survey. *CoRR*, abs/1707.02268, 2017.

[6] Avishek Anand, Lawrence Cavedon, Matthias Hagen, Hideo Joho, Mark Sanderson, and Benno Stein. Conversational search – a report from dagstuhl seminar 19461. 5 2020.

[7] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco: A human generated machine reading comprehension dataset. 11 2016.

[8] Chia-Yuan Chang, Hsien-Hao Chen, Ning Chen, Wei-Ting Chiang, Chih-Hen Lee, Yu-Hsuan Tseng, Ming-Feng Tsai, and Chuan-Ju Wang. Query expansion with semantic-based ellipsis reduction for conversational ir.

[9] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. Trec cast 2019: The conversational assistance track overview. 3 2020.

[10] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. Cast 2020: The conversational assistance track overview, 2021.

[11] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. Cast 2021: The conversational assistance track overview, 2021., 2021.

[12] Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. Can you unpack that? learning to rewrite questions-in-context.

[13] Mohit Iyyer Mark Yatskar Wen-tau Yih Yejin Choi Percy Liang Luke Zettlemoyer Eunsol Choi, He He. Quac : Question answering in context. 8 2018.

[14] Carlos Gemmell and Jeffrey Dalton. Glasgow representation and information learning lab (grill) at the conversational assistance track 2020.

[15] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. 6 2015.

[16] Faegheh Hasibi Hideaki Joko, Emma J. Gerritse and Arjen P. de Vries.

[17] Timothy C Hoad and Justin Zobel. Methods for identifying versioned and plagiarized documents the widespread use of on-line publishing of text pro-motes storage of multiple versions of documents and mirroring of documents in multiple locations, 2003.

[18] Akari Asai Hideaki Takeda Yoshiyasu Takefuji Jin Sakuma Yuji Matsumoto Ikuya Yamada, Hiroyuki Shindo.

[19] Hideaki Joko, Emma J. Gerritse, Faegheh Hasibi, and Arjen P. de Vries. Radboud university at trec cast 2021. In *Text Retrieval Conference*, 2021.

[20] Hideaki Joko, Faegheh Hasibi, Krisztian Balog, and Arjen P. De Vries. Conversational entity linking: Problem definition and datasets. pages 2390–2397. Association for Computing Machinery, Inc, 7 2021.

[21] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. pages 39–48. Association for Computing Machinery, Inc, 7 2020.

[22] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. 9 2019.

[23] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. 10 2019.

[24] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. Distilling dense representations for ranking using tightly-coupled teachers. 10 2020.

[25] Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and David R Cheriton. Trec 2020 notebook: Cast track.

[26] Sheng Chieh Lin, Jheng Hong Yang, Rodrigo Nogueira, Ming Feng Tsai, Chuan Ju Wang, and Jimmy Lin. Multi-stage conversational passage retrieval: An approach to fusing term importance estimation and neural query rewriting. *ACM Transactions on Information Systems*, 39, 10 2021.

[27] Donald Metzler and W Bruce Croft. A markov random field model for term dependencies, 2005.

[28] Ani Nenkova and Kathleen McKeown. *A Survey of Text Summarization Techniques*, pages 43–76. Springer US, Boston, MA, 2012.

[29] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. 1 2019.

[30] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert. 10 2019.

[31] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. 4 2019.

[32] Yasuhito Ohsugi, Itsumi Saito, Kyosuke Nishida, Hisako Asano, and Junji Tomita. A simple but effective method to incorporate multi-turn context with bert for conversational machine comprehension. 5 2019.

[33] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics.

[34] Ted Pedersen and Satanjeev Banerjee. Extended gloss overlaps as a measure of semantic relatedness, 2003.

[35] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. 10 2019.

[37] Siva Reddy, Danqi Chen, and Christopher D. Manning. Coqa: A conversational question answering challenge. 8 2018.

[38] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. 4 2017.

[39] Ivan Sekulić, Mohammad Aliannejadi, and Fabio Crestani. Extending the use of previous relevant utterances for response ranking in conversational search.

[40] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27:443–460, 2 2015.

[41] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. 9 2014.

[42] Svitlana Vakulenko, Nikos Voskarides, Zhucheng Tu, and Shayne Longpre. Leveraging query resolution and reading comprehension for conversational passage retrieval. 2 2021.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. volume 2017-December, 2017.

[44] Nikos Voskarides, Dan Li, Pengjie Ren, Evangelos Kanoulas, and Maarten De Rijke. Query resolution for conversational search with limited supervision. pages 921–930. Association for Computing Machinery, Inc, 7 2020.

[45] Jheng-Hong Yang, Sheng-Chieh Lin, Chuan-Ju Wang, Jimmy J. Lin, and Ming-Feng Tsai. Query and answer expansion from conversation history. In *TREC*, 2019.