MANtis: a novel information seeking dialogues dataset

Alexandru Balan

TU Delft
Delft University of Technology

MANtIS: a novel information seeking dialogues dataset

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
TRACK DATA SCIENCE

by

Alexandru Balan
born in Bucharest

Web Information Systems
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://wis.ewi.tudelft.nl

MANtIS: a novel information seeking dialogues dataset

Author: Alexandru Balan
Student id: 4722574
Email: A.Balan@student.tudelft.nl

Abstract

Nowadays, most users access the web through search engine portals. However, information needs can often be ill-defined or too broad to be solvable by a list of results the user has to scroll through, which implies that he is most likely required to refine the need by himself to reach the desired result. In recent years, researchers have attempted to tackle these issues through conversations, more specifically through *conversational search* [104]. This topic has seen an increase of interest from the research community, proven by the appearance of specialized workshops and seminars. The general public has also started to show interest, proven by the emergence of a wide range of virtual assistants, such as Google Assistant, Microsoft Cortana or Amazon Alexa. As such conversational systems seek to fulfill an information need of a user, they should be able to elicit and fully understand his requirements regardless of the domain, track the conversation as it evolves while attempting to clarify the initial information need and provide suggestions and answers that are based on concrete knowledge sources. Although various developments in domains adjacent to conversational search enabled us to better understand natural language, there is a lack of large-scale datasets that are appropriate for training models to perform conversational search tasks. Through our research, we have built a collection of over 80,000 conversations that fulfill the requirements of a conversational search dataset. We have benchmarked this dataset on three distinct tasks using multiple baselines.

Thesis Committee:

Chair:	Associate Prof. Dr. C. Hauff, TU Delft
University supervisor:	Associate Prof. Dr. C. Hauff, TU Delft
Committee Member:	Assistant Prof. N. Tintarev, TU Delft
Committee Member:	Associate Prof Dr. Z. Al-Ars, TU Delft

Contents

Contents **iii**

List of Figures **v**

1 Introduction **1**
1.1 Research objectives . . . 3
1.2 Our Approach . . . 3
1.3 Main Findings . . . 4
1.4 Outline . . . 5

2 Related work **7**
2.1 Information Retrieval . . . 7
2.2 Conversational Search . . . 15
2.3 Question Answering . . . 18
2.4 Dialogue Systems . . . 21
2.5 Conversational Datasets . . . 25

3 Building the dataset **33**
3.1 Inclusion criteria . . . 34
3.2 Initial dataset analysis . . . 35
3.3 Utterance intent labeling . . . 37
3.4 Grounding documents dataset . . . 39
3.5 Tasks . . . 41

4 Experiments **45**
4.1 Models . . . 45
4.2 Results . . . 51

5 Conclusions **63**
5.1 Future work . . . 64

Bibliography **67**

A Glossary **81**

A.1 Hyper-parameter tuning 81
A.2 Vader Score threshold 81
A.3 Intent annotation process 82

List of Figures

1.1 Example of an ideal conversational search experience (left) and an actual conversation performed between the author and Google Assistant (right). . 2

1.2 Example of a conversation from MANtIS. The user has a complex information need, with several specific details. The agent gives an initial response that could solve the issue. Afterwards, the user gives *negative feedback* regarding the solution. Finally, the agent offers another solution and refers to a link that contains more details. 4

2.1 Visualization of a traditional retrieval process in an IR system. 8

2.2 Visualization of the document vector space, as shown in [114]. The left figure shows how documents are represented in a vector space, while the right figure depicts an ideal vector space, in which documents can be clearly grouped based on their term representation. 12

2.3 Visualization of a typical LTR mechanism. 13

2.4 Graphical representation of the two types of Neural IR models. 14

2.5 Examples of conversations originating from 3 types of systems identified by Gao et al. [50]. QAS, task-oriented DS and chit-chat bots. 17

2.6 Visualization of a typical task-oriented DS 23

3.1 Distribution of conversations across the 14 domains. The green bars show the weight of each domain in terms of conversations in the dataset. The red bars 36

3.2 Evolution of the 14 domains throughout time in terms of the yearly number of conversations 36

3.3 Visualization of the utterance length per domain. Each box displays the mean number of words (vertical line inside the box) and the confidence interval. 37

3.4 Visualization of the conversation size per domain. Each bar displays the mean number of utterances per conversation and the confidence interval . 37

3.5 Dataset conversation examples. yellow shows document grounding, green displays final positive message from the original poster, pink are clarification questions and gray highlights initial information needs. All examples in the table are multi-turn. The right-most column shows the conversation labels for each utterance. 38
3.6 Intent distribution of the dataset. In total, there are 8334 distinct intent labels. 39
3.7 Distribution of un-crawlable URLs by type of file. The green bars show the share of images . The red bars show the share of web documents or URLs that are not identifiable by extension. 41
3.8 Visualization of mean document length by site. The width of the bar represents the mean, while the horizontal line represents the confidence interval. 42
3.9 Example of a conversation split by contexts and responses. The red squares show the conversation contexts, while the blue squares with gray background show the potential responses. 43

4.1 Functional diagram of DMN. 47
4.2 Functional diagram of BERT.The red squares depict the input of the network. The yellow squares depict the embedding representation of the words. The green squares represent the output sequence of the network, which is used to generate the final output. 49
4.3 Functional diagram of the multi-task learning process. Components surrounded by red squares are the ones that were added to the base DMN architecture. 51
4.4 The bottom figure shows the distribution of conversations by the number of turns in the test dataset. The top two figures show the performance of DMN and BERT on the CRR10 and CRR50 tasks based on the number of turns per conversation context. MAP is averaged over five runs with different seeds. The vertical bars in the top 2 figures represent the confidence intervals. 53
4.5 The top figure shows the distribution of conversations by category in the test dataset. The bottom figures depict the performance of DMN and BERT by category on the CRR10 and CRR50 tasks. MAP is averaged over 5 runs with different seeds. 53
4.6 Confusion matrix at a domain level for the CRR50 task, performed by DMN. Darker shades of red indicate a higher number of confusions between the two domains. Cases where the highest ranked document and the true document were in the same domain have been removed for clarity. . . 55
4.7 Performance of BERT (standard and multi-label) with respect to the precision of identifying each unique intent. 56
4.8 The bottom figure shows the distribution of conversations by the number of turns in the test dataset. The top two figures show the performance of DMN and BERT on the GDR10 and GDR50 tasks based on the number of turns per conversation context. MAP is averaged over five runs with different seeds. 58

4.9 The top figure shows the distribution of conversations by category in the GDR test dataset. The bottom figures depict the performance of DMN and BERT by category on the GDR10 and GDR50 tasks. MAP is averaged over five runs with different seeds. 59
4.10 Learning curves for the standard and MTL variants of DMN on the CRR10 and CRR50 tasks. The colored lines represent mean performance on each evaluation step, average over five runs. The shades surrounding the lines represent the confidence interval. 60
4.11 Learning curves for the standard and MTL variants of DMN on the GDR10 and GDR50 tasks. The colored lines represent mean performance on each evaluation step, average over five runs. The shades surrounding the lines represent the confidence interval. 60

A.1 The interface that the annotators were required to use in order to label utterance intents. 83

Chapter 1

Introduction

Steady progress in machine learning and, more specifically in neural networks, has paved the way for significant improvements in how information systems understand natural language and interact with people through dialogues [119, 74]. Such systems can be described as being *conversational* if they assist users through a prolonged dialogue, either in spoken or written form. Figure 1.1 showcases an example of how a conversational agent should perform in a real-life scenario, along with an example of how current conversational agents behave (in this example, Google Assistant). In the first image, the agent recognizes the feedback provided by the user and adjusts his answer accordingly. The virtual assistant in the second example does not take into account further details ("My computer runs Mac OS") and provides the same final answer as the one retrieved after the first turn.

Conversational search [104, 58] is a research area focused on the creation of agents that are capable of fulfilling a complex information need expressed by a human user. An example of such a need is depicted in Figure 1.1, as the user expresses multiple requirements as the conversation progresses ("automatically find all residual files", "my computer runs Mac OS", "I'm looking for something that is free"), as opposed to a simple request, such as "What is the weather today."

In contrast to standard search engines, which usually return a results page, a conversational search agent should be able to handle mixed-initiative interactions, which implies a "flexible interaction strategy in which each agent (human or computer) contributes what it is best suited at the most appropriate time." [4]. Therefore, a conversational agent should be able to fully understand natural language queries, respond to follow-up questions or feedback and interact with the user whenever it deems necessary. Through multiple turns, the agent should be able to construct a clearer picture of the user's information need. The conversation in the left part of Figure 1.1 shows an example of this process: the agent first tries to identify the operating system the utility should operate on, then recognizes the additional details provided by the user and supplies a potential answer. Following the negative feedback from the user, the agent adjusts its answer. The final positive feedback signals the end of the conversation.

Conversational search is a topic that has received significant attention in more recent times, proven by the appearance of several dedicated workshops (CAIR[1] 2017-

[1] https://sites.google.com/view/cair-ws/home

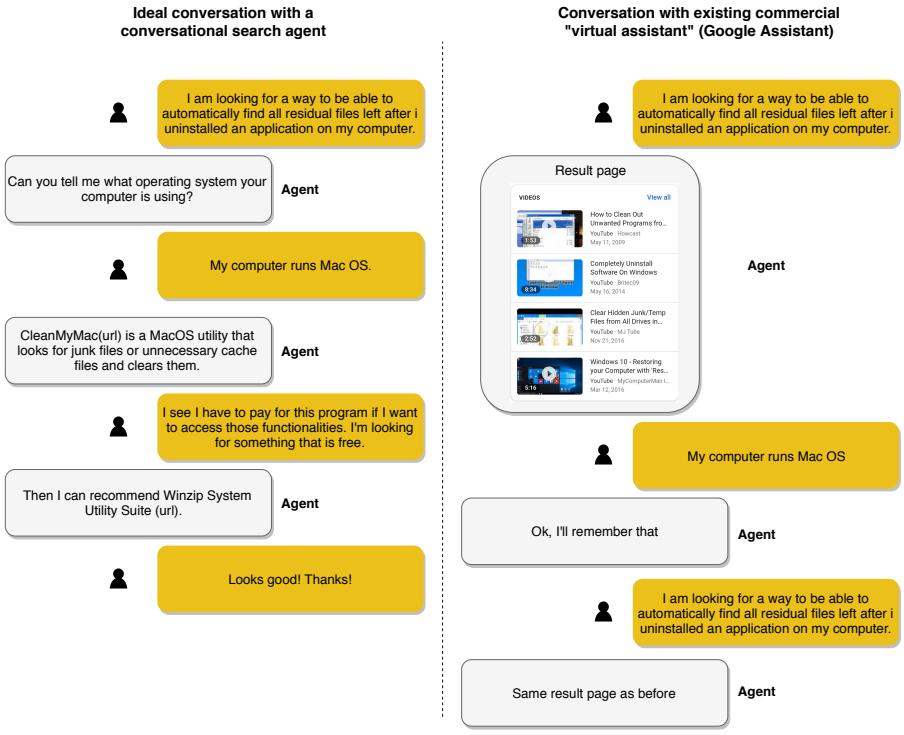


Figure 1.1: Example of an ideal conversational search experience (left) and an actual conversation performed between the author and Google Assistant (right).

2018, SCAI[2] 2017-2019), seminars[3] and the emergence of a wide range of global deployments of *virtual assistants*, such as Google Assistant, Amazon Alexa or Microsoft Cortana. However, these systems are still limited to simple interactions and short memory and share more characteristics with Question Answering Systems (QAS) than proper conversational search. In a study from 2018 by Lopatovska and Williams [81] on the usage patterns of Amazon Alexa, several participants reported that the agent "had some silly responses in relation to the questions we asked." Another study from the same year [118] measured what were the top 30 commands that were addressed to Alexa by the participants. They found that all of them were queries that did not require further refinement, such as "what's the weather", "volume down" or "what's the temperature."

As we will detail in Section 2.2 and Section 2.5, a theoretical framework for conversational search has already been established and many conversational tasks have been solved by adjacent research domains. Moreover, many efforts were directed towards building conversational datasets (that are used to train neural models to solve conversational tasks) by researchers from various domains: Dialogue Systems (DS) [7, 85], Natural Language Processing (NLP) [17, 106, 112, 39] and Information Retrieval (IR) [101, 132, 135]. However, none of these datasets fulfill all the requirements of a conversational search dataset (defined in Section 2.2), while also being large enough to be eligible for training a model. Therefore, in order to obtain a true

[2] https://scai.info/
[3] https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=19461

conversational search dataset, the knowledge, techniques and tasks that are specific to each domain need to be aggregated into one system.

1.1 Research objectives

Although the research community has released many conversational datasets, we argue that neither encompasses all the requirements of a conversational search dataset. Having built collection that does, we can then use state-of-the-art models in order to assess to what extent these architectures can be used in real-world applications. Additionally, given that our data will be eligible for performing multiple tasks, we study the effect of training the model to solve several tasks at once (process formally known as multi-task learning [22]), given that previous research has shown that this can lead to further improvements of the model [10, 44, 5, 11].

To conclude, apart from offering an *overview of existing conversational datasets*, this research contributes to the field by solving two challenges:

- Design an open-source framework that is capable of building a dataset that is appropriate for training models that can be used in a Conversational Search System (CSS).

- Use the newly created dataset to evaluate several models for a set of predefined tasks. Based on the fact that our dataset can be used for multiple conversational tasks, we also study the effect of multi-task learning on the performance of our models.

1.2 Our Approach

As a starting point for building the dataset, we have used the data provided by Stack Exchange[4]. Stack Exchange is an online collection of Q&A websites from various domains, such as computer science, literature or travel. Each site covers exactly one topic, where all the questions, answers and users are rated based on a reputation award system. This process allows all of these sites to be self-moderating, thus removing the need of an external entity to enforce rules. Starting from a publicly available data dump[5], the proposed framework fetches the chosen sites, processes and aggregates the data and returns a formatted dataset. The framework also provides a pipeline to transform the obtained dataset into training data for neural architectures by using negative sampling in a similar fashion to Yang et al. [147]. In their original form, the conversations are not annotated to indicate the intents that the users are expressing through their utterances. To achieve this, we have manually annotated a subset using a custom web interface. Furthermore, in order to extract the *concrete knowledge sources* that were used in the response, we have employed a web crawler to download the text content of the web pages that were mentioned in the conversations.

The resulting dataset can be used for performing 3 tasks: predicting the most probable next response of the conversational agent based on the current conversation con-

[4] `https://stackexchange.com/sites`

[5] `https://archive.org/details/stackexchange`

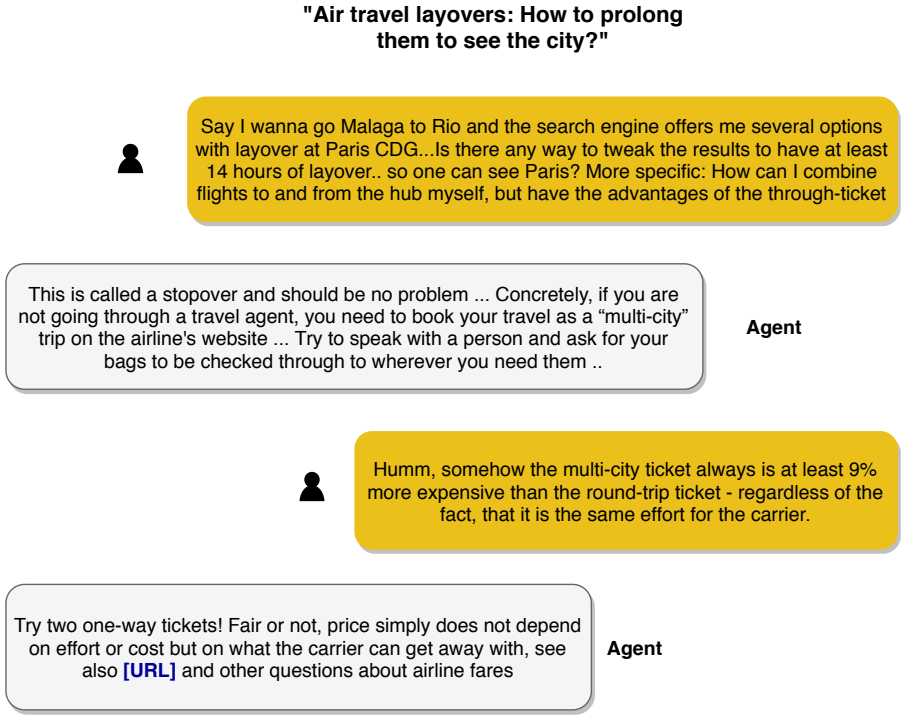


Figure 1.2: Example of a conversation from MANtIS. The user has a complex information need, with several specific details. The agent gives an initial response that could solve the issue. Afterwards, the user gives *negative feedback* regarding the solution. Finally, the agent offers another solution and refers to a link that contains more details.

text (also known as conversational response ranking (CRR) [147]), predicting the intent of a conversation participant based on the utterance [102] and predict the grounding document contained in the response of the conversational agent based on the context (which will be referred to as Grounding Document Ranking (GDR)). In the first evaluation phase, we have analyzed the performance of several models on each task independently. Afterwards, in order to verify the effect of learning multiple tasks at once has on a neural baseline, we have adapted an existing architecture, called Deep Matching Networks (DMN) [147], to accept inputs originating from different training datasets to solve different tasks.

1.3 Main Findings

Using our data processing pipeline, we have created a novel dataset called MANtIS (short for *multi-domain information seeking dialogues dataset*). It contains 80,326 two-way conversations and 411,013 utterances spanning over 14 domains. We have manually annotated using nine types of intents a number of 6,701 utterances spanning over 1,356 conversations. Using the links that were mentioned by the agent in the conversations, the web crawler was able to extract 116,061 documents. Figure 1.2 shows an example of a conversation that was captured in the dataset.

For the model training datasets, we provide two variants for the CRR and GDR

tasks: `CRR10`, `CRR50` and `GDR10`, `GDR50`, respectively. The same model performed best for all of the three proposed tasks, namely BERT, which has been previously shown to obtain state-of-the-art performance on a variety of NLP tasks [38]. Further analysis has shown that models generally perform better on domains that are more distinguishable from the others. On the CRR tasks, BERT performs better on the conversations that have a shorter history, while DMN is not affected by this factor. On the GDR task, the results show the opposite.

In the multi-task learning environment, the results have shown that jointly learning the CRR and GDR tasks improves the performance over each of the tasks learned independently. Learning to predict intents together with the other tasks did not show any improvements. To verify whether the low number of intent labeled conversations had a negative impact on the performance of the model when jointly learning intent prediction along the other two tasks, we have used BERT under a weakly supervision environment to predict labels for more conversations. The results have shown that having more intent labels slightly improves the performance of the model on the CRR task.

1.4 Outline

We describe related work extensively in Chapter 2. Chapter 3 is reserved for describing the process of building and evaluating the reusable, multi-domain dataset. In Chapter 4 we report the results of training the models on the defined tasks, both independently and in a multi-task learning setup. Lastly, we draw the conclusions of the research in Chapter 5.

Chapter 2

Related work

Conversations between humans have been studied for several decades and in order to understand conversational search, one has to first understand several neighbouring research areas to obtain a clear overview. Given that our research is based primarily in the field of IR, we will start with an overview of the field. Afterwards, we will provide an extensive survey on conversational search, focusing on the defining characteristics a system from this area should possess. Furthermore, given that the conversational search field is rather new, a deeper understanding of older neighbouring areas that tried to solve similar problems is necessary. For this reason, the survey also includes insights into QAS and DS. The last section is dedicated to researches that focused on designing and creating datasets that are partially qualified to be used for conversational search.

Several existing surveys constituted the foundation for writing this chapter. Chen et al. [25] wrote an insightful survey on Dialogue Systems, while the surveys of Bouziane et al. [14] and Allam and Haggag [2] provided valuable insights regarding QAS. For a deeper understanding of information retrieval models, one of the starting points was the survey by Hiemstra [59]. Other researches were discovered either by snowballing (following citations starting from surveys/researches) or by leveraging the search function of Google Scholar[1]. Some of the keywords that were used in order to discover previous work were: *question answering systems survey*, *dialogue systems survey*, *conversational search*, *information retrieval*.

2.1 Information Retrieval

Information retrieval usually requires a user who is interested in fulfilling a certain information need and a system that provides the means to achieve this. In traditional search, the user provides the system with a query describing the information need. Given these prerequisites, the central problem in IR is to rank the available documents in a manner that best reflects the user's needs [91]. While the information need can be expressed through various channels: video, speech, text, we will only focus on *text retrieval*.

Traditionally, such a retrieval system would first perform an indexing process on the document corpus and then, given a query in text form, the system would compute a

[1] `https://scholar.google.com/`

retrieval score for each document. The procedure is depicted in Figure 2.1. Examples of IR processes occur very often in our daily lives: every time we use a web search engine, such as Google or Bing, we trigger an IR process that returns the results we were looking for.

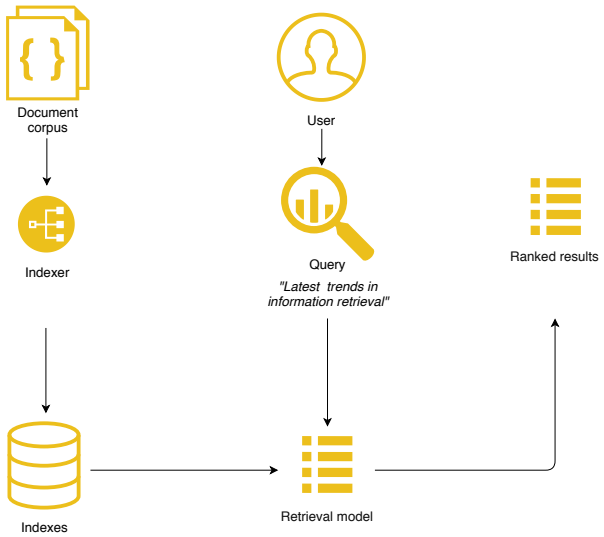


Figure 2.1: Visualization of a traditional retrieval process in an IR system.

2.1.1 Tasks

Apart from categorizing IR systems based on the form of the user query, one can also differentiate by the type of task the system performs. The most common tasks that can be encountered are *ad-hoc retrieval* and *question answering* [131]. The latter will be treated in more detail in the next section.

Ad-hoc retrieval tasks involves retrieving a list of ranked documents based on the input query. Initially, these tasks were performed on a corpus of newspapers or government documents [93]. Nowadays, these tasks are mostly performed by web search engines [9]. In these kinds of systems, the document corpus remains relatively stable, whereas the variety of queries that need to be handled can be virtually infinite, which explains the *ad-hoc* characteristic. This category of IR systems has been studied extensively for decades, as research in this area uncovered various techniques to improve the retrieval performance. In 1971, Rocchio [110] propose a system that accepts *relevance feedback* from the user in order to refine query results. This form of critique is also important for a CSS, as an agent must be able to re-orient his search efforts based on the user's feedback. However, traditional approaches have a limited space of possible feedback expressions. In a real conversation, the system should allow any form of critique. To achieve this, a natural language model is necessary to identify advanced forms of reasoning and feedback. To this end, modern neural networks incorporate attention models with large external memory [126] in order to track queries that span over several rounds of retrieval. Other researches focused on integrating external knowledge to improve query suggestion, such as click-through and web session data Cao et al. [20].

2.1.2 Evaluation

Generally, evaluation in IR can be performed from 2 different perspectives: from a *user perspective* – by measuring the level of satisfaction of the users when using the system – or a *system perspective* – by assessing the retrieval performance of a system that returned a ranking of documents based on a set of metrics [137] which, ideally, should reflect the degree to which a certain information need was fulfilled. Since the purpose of an IR system is to solve the information need of its users, the user based evaluation would seem to be the best reflection of the quality of the system. However, as stated in [137], such an evaluation is very expensive due to the larger number of users that should be involved. Therefore, in most researches, the system-based evaluation is preferred.

The challenges of evaluating IR systems have been tackled for several decades already. Robertson [109] discussed about the 2 main types of system-based evaluations: operational versus laboratory tests. In the latter, all the variables are controlled as much as possible in order to remove all extraneous variations that can alter the results. However, to answer the questions that are stemming from real-world problems, testing must be conducted in operational environments. Most of the researches that will be mentioned in this section have preferred a laboratory setting.

Generally, in order to perform any kind of traditional IR evaluation, one must possess 3 components [115]:

- A collection of uniquely identifiable *documents*
- A set of distinct *queries*
- A set of *relevance judgments*, which contains pairs of *(query id, document id)* indicating the relevance of each document with respect to a query. Generally created by a human annotator

However, when building a very large test collection, employing a large amount of human annotators can prove very expensive. Spark-Jones [123] proposed a technique called *pooling*, which involves creating a smaller subset of documents which is still sufficiently large to be representative with respect to the original collection. The relevance judgments would then be chosen based on random sampling from a pool.

Metrics

Regardless of how the test collection is built, metrics are also required in order to be able to accurately compare various systems. These metrics should be able to accurately reflect a user's satisfaction with the system. We will focus on the metrics that were used in the papers reviewed in this chapter and the ones we have used throughout our research.

The *Reciprocal Rank* is a metric that takes into account the rank of the first relevant document from a list of top-ranked results. More specifically, the reciprocal rank is the inverse of the rank of the first relevant document or 0 if there is no relevant document in the ranking. The Mean Reciprocal Rank (MRR) is the average of the Reciprocal Rank for a given set of queries. The major disadvantage of this metric is that it considers only

the position of the first relevant document. If there are multiple relevant documents for the query, one of the following metrics are preferred.

$$MRR = \begin{cases} \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_{first\_relevant}}, & \text{if relevant doc in list} \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

Precision reflects how many of the retrieved documents were relevant. It is the fraction of relevant documents out of all the the retrieved documents. The *recall* is the fraction of relevant documents that were retrieved out of the total number of relevant documents. Precision at k ($P@K$) is similar to the precision but is only considering the top k documents in the ranking. Both of these metrics do not take into account the order in which the documents appear in the ranked list. The metric that does is the *average precision* (AP), which requires the computation of the precision at every position in the ranked list. Apart from the P@k, it uses the indicator function *rel* which is equal to 1 if the document is relevant and 0 otherwise and the total number of documents $n_{relevant}$. The metric is defined by Equation 2.2. Additionally, the *mean average precision* (MAP) is the average of AP.

$$AveP = \frac{\sum_{k=1}^{n} P@k \times rel(k)}{n_{relevant}} \tag{2.2}$$

Another metric that takes into account both the precision and recall is the *F1-score* and is defined as the harmonic mean between precision and recall.

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

Three alternatives to precision and recall based measures were introduced in 2002 by Järvelin and Kekäläinen [64] and measure the cumulative gain the user gains by inspecting the list of documents up to a specific rank. *Cumulative gain* computes the sum of the relevance *rel* of each document in a list of results. However, this metric does not take into account the position of each document in the list. The *Discounted Cumulative Gain* (DCG) does take it into account by penalizing highly relevant results that appear at the bottom of the result list. The DCG for a document at position p can be defined using Equation 2.4

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

However, given that different IR systems can retrieve result lists of various sizes for different queries, a metric that would be able to take this fact into account was necessary. The *Normalized Discounted Cumulative Gain* (NDCG) uses a normalization term, called Ideal DCG (IDCG) to achieve this. The IDCG uses a list of relevant documents sorted by relevance relative to the given query to (called REL_p) obtain the maximum possible DCG up to position p. Therefore, the NDCG can be described by Equation 2.5.

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

, where $IDCG_p$ is defined by the following equation

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{2.6}$$

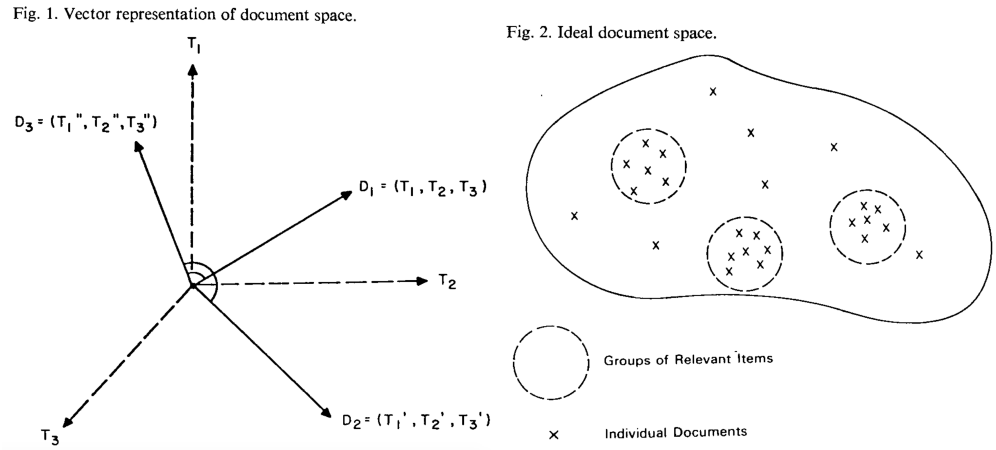
2.1.3 Traditional IR models

In an effort to formalize the retrieval process, the IR community has designed theoretical models that can encompass the assumptions behind an IR system [59]. Historically, it is considered that there are 3 main types of traditional IR models [59]. The first one is represented by the *Exact match model*. This model is the first one to be defined and also the simplest. As opposed to more recent models, those belonging to this category perform only exact matches, based on Boolean operations [141]. Therefore, the task of finding all documents that contain *information retrieval* and *survey* will be transposed into the query *information retrieval AND survey*, which will only retrieve documents that contain both words. Although easy to implement and understand, these models cannot provide a ranked list of documents.

Starting from Luhn's proposal [87] to "transform information into arrays of normalized idea building blocks and then to discover similarities", the *Vector Space Model* proposes a vector representation for the query and the documents and a relevance ranking based on the similarity of the vectors, usually achieved by measuring the cosine of the angle. However, it assumes that the similarity between the query and the document is correlated with relevance, which might not always prove to be true. The *Probabilistic Models* have strong foundations in probability theory and provide a stronger theoretical footing and clearer assumptions than the other two models [117]. In a system based on this model, the ranking of documents is based on the probability of relevance. In his research regarding probabilistic retrieval, Robertson [108] has shown that, under certain assumptions, "a ranking of documents in order of decreasing probability of relevance ... will be the best that is obtainable on the basis of those data".

Vector Space Models

Salton et al. [114] lay the fundamental concepts of the vector space model. They define a document or a query as a t-dimensional vector comprised of t weighted terms. The similarity between two documents or a document and a query can be computed as the cosine of the angle between the two represented vectors, as can be observed in Figure 2.2. However, computing the similarity is not the biggest challenge, but rather deciding upon the weights of each term in a vector. They define two term weighting systems: The *term frequency weighting system* (TF) is based on the observation that words that appear often in a document have a bearing on the content. However, if the same term appears in more documents, it loses its usefulness for the current content representation. This observation is the basis for the *inverse document frequency weighting system* (IDF). Combining these two observations lead to the *term frequency - inverse document frequency weighting system* (or *TF-IDF*), which states that if a "rare" word has a high number of occurrences in a document, it should be attributed a high weight in the document vector. The weighing scheme mathematical details will be presented in the following subsection.



(a) Vector representation of a document space (b) Ideal document space

Figure 2.2: Visualization of the document vector space, as shown in [114]. The left figure shows how documents are represented in a vector space, while the right figure depicts an ideal vector space, in which documents can be clearly grouped based on their term representation.

Probabilistic models

The probabilistic model was designed in an effort to "introduce arithmetic (as opposed to logic alone) into the problem of indexing" [88] in an effort to establish a firmer theoretical footing in IR. Models following this framework no longer consider document relevance as a binary attribute (relevant / not relevant), but rather as a probability. However, despite being one of the oldest formal models in IR, probabilistic models did not exceed in performance other IR models of the time (mostly due to the strict mathematical assumptions the model required) until BM25 was released [40].

BM25 [107] is a probabilistic model that ranks a list of documents based on the query terms that appear in each document. It follows a tf-idf weighing scheme similar to the one used in the vector space model. The key difference resides in the scoring function of each document, which follows a probability theory based formula instead of the cosine similarity. Given a query Q and its terms $q_1, q_2, ..., q_n$ and a document D, the scoring function of BM25 is defined by Equation 2.7. The leftmost term represents the IDF scheme, which assesses the information value a term possesses by normalizing the number of documents in which the term occurs ($n(q_i)$) by N - the total number of documents in the corpus. The rightmost term is the TF component, which takes into account the number of times a term appears in document D ($f(q_i, D)$), also known as the term frequency. This term is normalized by the length of the document $|D|$ divided by the average document length in the corpus $avgdl$. Further regularization is applied using the hyperparameters k_1 and b. Several variations of BM25 have also been developed, such as BM25F [107], which weighs different parts of the document (header, main text, anchors) with different degrees of importance. BM25 is a popular IR model used in the search engine industry, with key players such as ElasticSearch [2]

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

using it as the default ranking function.

$$BM25(D,Q) = \sum_{i=1}^{n} log\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \cdot \frac{f(q_i,D) \cdot (k_1 + 1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \tag{2.7}$$

Equation 2.7. The BM25 [107] scoring function

2.1.4 Learning To Rank

Before diving into the neural IR architectures, one must first understand the Learning To Rank (LTR) paradigm. LTR is an approach that is built on top of existing traditional IR models, such as BM25 or various vector space models. Generally, these methods use the input attributes of BM25, combined with other handcrafted features and a machine learning algorithm to re-rank the list of results that was generated by a traditional IR model for a certain query. A detailed visualization of this process is depicted in Figure 2.3. As can be observed, the LTR procedure closely resembles standard retrieval, with the exception of the rightmost part, which encompasses the re-ranking procedure.

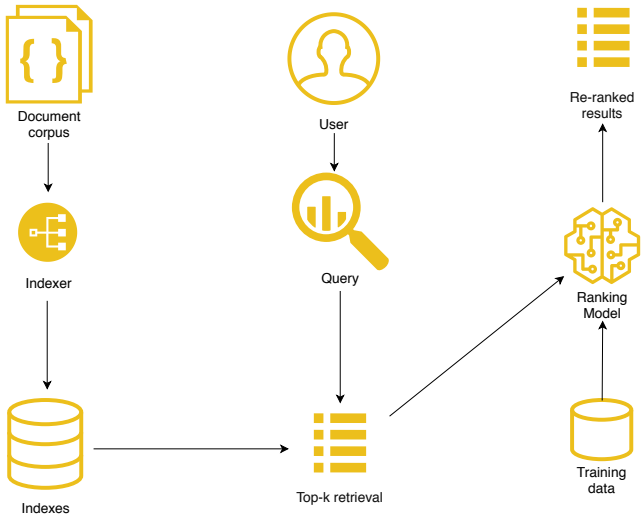


Figure 2.3: Visualization of a typical LTR mechanism.

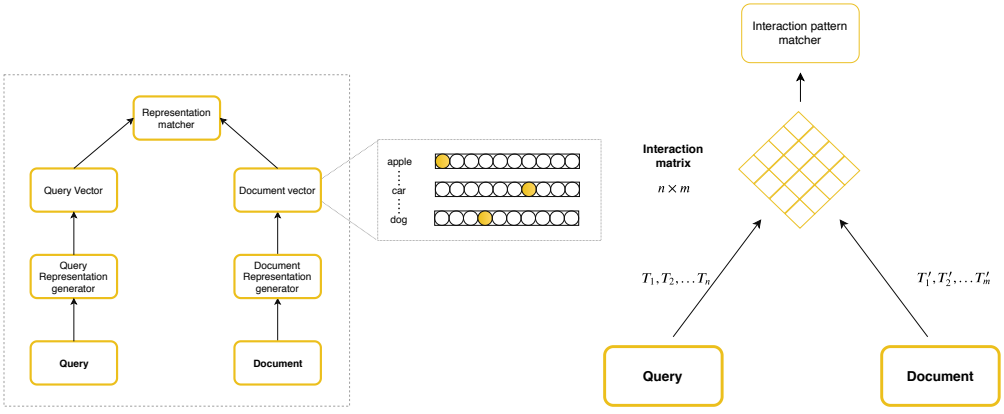
As Svore and Burges [130] mention in their research, the parameter tuning for BM25 and BM25F can be difficult and time-consuming. For this reason, they propose a new LTR framework called LambdaBM25, a neural-based ranking algorithm that accepts the same input attributes as BM25F (TF, IDF and document length). However, instead of optimizing the hyperparameters of the BM25F function by running an exhaustive search in order to obtain the best combination, the algorithm uses LambdaRank [18] to learn (and minimise) the function based directly on the input attributes. Their new model outperforms BM25F for documents that have multiple kinds of fields (URL, title, body, anchor, click data etc.) and for documents that possess only the body, anchor or click data. For matching between small passages of text, such as the title, BM25F performs similarly to LambdaRank.

2.1.5 Neural IR

Whereas LTR approaches generally require the creation of appropriate handcrafted features as input, neural IR models can leverage representation learning [75] to automatically discover the representation for a certain dataset of raw data. This enables the discovery of new, complex patterns that are not necessarily bound by human intuition.

Researches in this area distinguish between 2 types of Neural IR approaches [54, 55]: representation-focused and interaction-focused. In the representation-based approach, one must first generate representations (usually in the form of vectors) for both the query and the document that captures the distribution of the information each one contains and then apply a matching algorithm to obtain an estimation regarding their mutual relevance [91]. In the case of the interaction-based methods, local interactions between different pieces of text are built (usually through interaction matrices), which are then passed to a neural model that learns hierarchical interaction patterns in order to perform matching. Guo et al. [55] benchmarked 18 neural models that use either of the approaches on 4 datasets and found that interaction-focused networks generally perform better.

By analyzing the graphical representation of the 2 approaches in Figure 2.4, it can be immediately observed that neural models no longer work with predefined features, as was the case for the traditional IR models or their LTR extensions. Queries and documents are now represented in a different space, most often through vectors. In our research, we use *word embeddings* to obtain the vector representations of all the words by means of unsupervised learning, as explained in the work of Mikolov et al. [89]. Given that utterances in a conversations are inherently connected one to another, we have chosen to use only interaction-focused neural models throughout our experiments.



(a) Representation-focused (b) Interaction-focused

Figure 2.4: Graphical representation of the two types of Neural IR models.

Multi-task learning

Multi-task learning is a branch of machine learning in which multiple tasks are solved at the same time by the same model by exploiting both the common and different

characteristics of each task. It has been proved both empirically [10, 44] and theoretically [5, 11] that learning multiple tasks at the same time can significantly improve the performance of the model as opposed to learning every task separately, given that the tasks are chosen appropriately. More importantly, multi-task learning has been applied for both natural language processing [31] and information retrieval tasks [98, 23] with positive results. This approach improves general performance of models for several reasons: it enables information transfer between tasks, which translates into having more knowledge regarding each task [22], it helps the model filter out idiosyncrasies in the data when using unrelated tasks [111] and, most importantly, multi-task learning behaves as a regularizer by introducing inductive bias [22]. The first and third reasons are of utmost importance for our research as they drive the entire intuition behind our experiments in this area.

As mentioned, task selection is highly influential on the outcome of a multi-task learning setup. One of the most common approaches is to choose a related task as an auxiliary task. Caruana [22] used 8 tasks that predicted various characteristics of the road in order to predict the steering in an autonomous car, while Girshick [52] use multi-task learning to jointly learn the class and the spatial location of an object in an image. Another approach to task selection is called *adversarial* and implies using a task that is opposite to the main task and using an adversarial loss [41], which seeks to obtain the highest obtainable training error. This approach has seen promising results in *domain adaptation* tasks by using a gradient reversal layer [49] which, as its name suggests, reverses the gradient of the adversarial task by multiplying with a negative constant. By using this layer, the model is forced to learn representations that cannot distinguish between various domains. In our research, we have adopted the first approach in order to identify auxiliary tasks.

2.2 Conversational Search

As we have stated previously, conversational search is a research area that wishes to improve the current search paradigm by solving a user's information need through conversation, as opposed to standard, single-turn retrieval. However in order to build such a system, called a conversational search system, one must first hypothesize how conversational systems should function in general. Allen et al. [3] suggest that a conversational agent should track the current state of the conversation and update its current understanding of the information need based on the user's responses. Christakopoulou et al. [29] emphasises the need of an elicitation system to be able to understand the user's preferences. Moore [94] states that an agent should be able to ask the user to repeat or rephrase a piece of information in the case of a misunderstanding and detect when a conversation has reached an end.

Having defined these characteristics of a general conversational agent, several researches focused on conceptualizing what are the defining traits of an ideal conversational search agent. Radlinski and Craswell [104] proposed a theoretical framework, starting from an extensive definition of a CSS and when such a system would be useful. The proposed conversational search model encompasses various types of interactions, ranging from simple query searches up to complex situations, in which the user provides different types of feedback. Furthermore, the researchers propose a list of

properties that could be used to measure the extent to which a given system is conversational. Azzopardi et al. [8] build upon the existing research by abstracting the main actions, decisions and tasks that occur during a conversation between a user and a conversational search agent.

Summarizing the findings of the two researches [104, 8] results in a set of requirements that a CSS should be able to fulfill when interacting with the users:

- The CSS is able to *elicit and understand* the user's information need.
- The CSS is able to identify the user's *intent* and, conversely, express its own.
- The CSS is able to remember what has been said previously, enabling the user to reference past information. It has been shown that context preservation is essential for maintaining a high level of user satisfaction with regards to a conversational system [70].
- The CSS is able to progressively *refine and clarify* the initial information need based on *feedback* from the user.
- The CSS is able to provide *answers, suggestions, summaries, recommendations, explanations, reasoning* and divide the problem into sub-problems, given its extensive knowledge in *different domains* and based on *concrete knowledge sources*.
- The CSS is able to *take initiative, ask questions* back and decide which types of actions are best suited in the current conversation context.

Exploring recent developments in conversational AI [50] (which encompasses various types of conversational agents) quickly reveals that, although many breakthroughs were achieved by improving neural network architectures, current systems are not yet able to fulfill all the requirements of a CSS.

Gao et al. [50] have performed an extensive survey on neural approaches to conversational AI and their commercial applications, showing that current systems fall into three main categories:

- Conversational QA Agents, which are designed to solve an information need of the user by directly answering their question, without further rounds of refinement. These agents usually have access to large-scale knowledge bases but their architecture does not support extended conversations. For example, Choi et al. [28] created a dataset for training Machine Reading Comprehension (MRC) models that generate an answer based on the retrieved passages. However, in these kind of datasets, the agent does not attempt to take initiative in order to clarify the information need and is focused on always providing a direct answer. Another example of a QA agent created by a company is ActiveQA[3], an agent created by Google that transforms the questions in order to retrieve the best answer.

- Task-oriented DS, such as slot-filling systems, can handle more complex situations but are limited to a predefined set of operations, as they are bound by

[3] https://github.com/google/active-qa

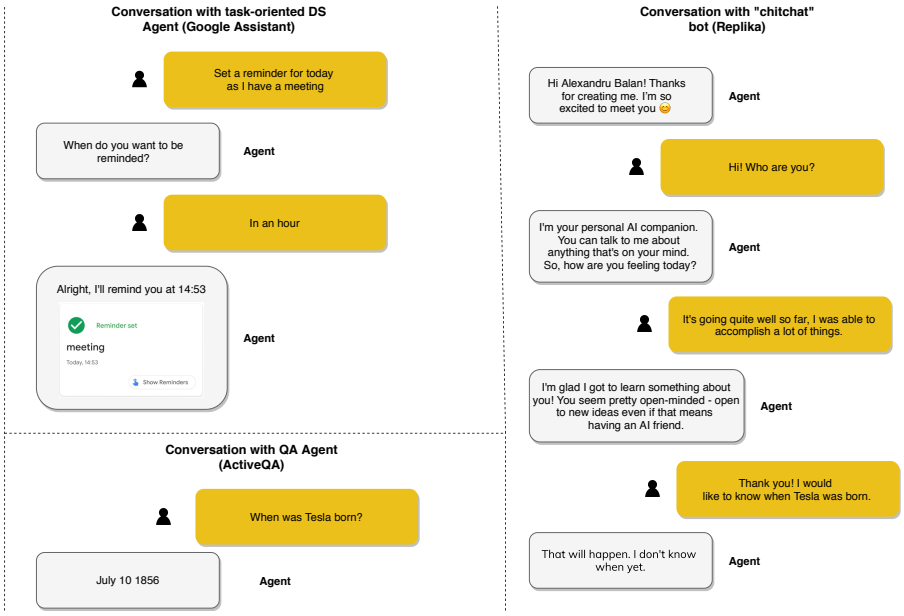


Figure 2.5: Examples of conversations originating from 3 types of systems identified by Gao et al. [50]. QAS, task-oriented DS and chit-chat bots.

a finite list of domains in which they can provide answers. Commercial task-oriented DS can often be found in smartphones in the form of "virtual assistants", such as Amazon Alexa[4] or Google Assistant[5]. However, as we have explained previously, they are limited to a set of tasks, such as providing weather information or setting alarms.

- On the other side of the DS spectrum, *social chatbots* facilitate natural generic interactions between humans and computers. However, these systems are focused on social interactions and their responses are usually not grounded in the real world. Replika [6] is an example of an emotion-aware conversational agent that is able to generate responses based on the emotion or the domain of the conversations. However, by analyzing transcripts of conversations [46], it can be observed that these architectures lack the *search* component of a conversation, as they are not trained to return facts or grounding. Figure 2.5 shows examples of systems from all of the 3 categories identified by Gao et al. [50].

Although a theoretical framework for CSS has been established [104, 8] and subproblems of CSS, such as predicting the user's intent based on their utterance [102] or ranking responses from agents [147], have been tackled successfully by using neural architectures, Gao et al. [50] show that all existing commercial solutions lack at least one of the necessary properties.

The importance and challenges of conversational search have been discussed extensively in the report from the Third Strategic Workshop On Information Retrieval

[4] https://developer.amazon.com/en-US/alexa
[5] https://assistant.google.com/
[6] https://replika.ai

(SWIRL 2018) [33]. As part of the initial questionnaire of the workshop, participants were asked to express what topics they currently consider to be of utmost importance. Conversational search was chosen as the most important one.

More importantly, the workshop participants identify several obstacles that might occur when developing conversational systems. One is that "Reusable datasets may be difficult to design or acquire due to the personalized, interactive nature of the task", which accentuates the need for new, improved datasets. Moreover, there was a consensus over the fact that advancements were made in all 3 domains that compose the conversational search field: Information Retrieval, Natural Language Processing and Dialogue Systems. This includes the creation of a multitude of conversational datasets, as we will detail in Section 2.5. However, all of these datasets either lack some of the requirements of a CSS or are too small to be used for training a model that can perform conversational search tasks.

2.3 Question Answering

Question answering is a research area that combines knowledge and techniques from IR and NLP. However, while IR is usually focused on document retrieval, QA is focused on providing the exact answer to the user. This idea started from the fact that, in many cases, the user is looking for the exact answer, rather than a list of documents to browse through [60].

Starting from this intuition, the TREC-8 Question Answering track [138] was one of the first major endeavours into evaluating QAS. The participants would receive a collection of text documents and 200 questions that can be solved with short answers from at least one of the documents. The documents were composed mostly of newspaper articles from various domains. The task to be performed was to return a ranked list of $(document\_id, answer)$ pairs for each question such that the selected string contained the answer to the question. The evaluation was performed by human assessors who judged whether the answer solved the question or not. In total, 45 runs were submitted, which shows that there was already significant interest in the area.

Conceptually, a QAS involves a pipeline formed of 3 main components [83]: (a) a *question processing subsystem* that processes and converts the question in natural language form into a query that is understandable by the retrieval system, (b) a *document retrieval subsystem* that searches for the candidate documents, (c) an *answer extraction subsystem* that extracts the snippet containing the direct answer from one of the candidate documents. This last module is the one that is distinguishing a QAS from a traditional IR system.

Depending on the area of expertise these systems are designed for, we can distinguish between *open-domain* QAS and *domain-specific* QAS [60]. Open-domain question answering can handle questions about any topic and usually rely on large, open-domain knowledge sources such as Wikipedia [24]. Closed domain question answering systems are designed to solve questions under a specific domain, such as movies, weather or music. This involves the development of specific ontology to capture the relationships between domain-specific concepts [99].

2.3.1 Domain-Specific Question Answering

One of the first known QAS to be released was BASEBALL [53], a system that was designed to answer questions related to baseball games played in the USA over one season. Given a question, the system would analyze it using available linguistic knowledge and transposed into a query that was used against a structured database containing baseball-related information. Although sophisticated at the time, the system had two major limitations: it was limited to one specific domain with no possibility to extend and it was designed to communicate with a structured database instead of a large, unstructured collection of data. Similar systems appeared in the years after, such as the LUNAR system [144], which answered questions related to the analysis of rocks found on the moon by the Apollo mission and was able to respond to over 90% of the questions asked by the scientists. An extensive analysis of these early systems can be found in the work of Androutsopoulos et al. [6].

In an effort to make these types of systems more reusable, more modern systems started to propose frameworks that allow easy plug-in of various ontologies and data sources. AquaLog [82] is one of the earlier solutions that accepts queries in natural language and an ontology as inputs and outputs the available semantic markup. Firstly, the query is being transformed into a <subject, predicate, object> triple using GATE [34], a framework for NLP. This allows them to identify relationships, terms, questions and indications (such as who/when/which etc.) and place the query in a specific category. Having made the representation, the system tries to match the triple with a candidate in the ontology by computing the string similarity. AquaLog also takes advantage of WordNet [90] in order to try additional combinations with synonyms when the initial matching fails. If only some of the terms in the triple are matched completely, a list of multiple-term candidates is displayed to the user and he is then expected to choose the appropriate one. This feedback system allows AquaLog to constantly learn new jargon and add them to the ontology. An example is the query "Who collaborates with the knowledge media institute?". Assuming that the system is not able to disambiguate the term "collaborate", it will present a list of candidate relations out of which one is "has-affiliation-to-unit". If the users chose that relation, a new mapping is added between the two terms so that the next time the system can recognize it. To evaluate the system, 76 questions were created by 10 human annotators and the system was able to handle correctly 48.68% of them. Although the result might not seem impressive, it is worth noting that there were no linguistic constraints imposed on the participants. By analyzing further, the researchers discovered that 69% of the errors were caused by insufficient linguistic coverage. Therefore, improving the coverage would yield significantly better results. PowerAqua [84] is a system that has evolved from AquaLog to enable multi-ontology QA by merging information from multiple ontologies in order to retrieve the best answer. By using a similar methodology with the previous research, the system was able to now handle 69% of questions correctly.

Other related work from the same period use the same pipeline. PANTO [139] takes a question as input and executes a SPARQL query on a given ontology. As AquaLog, it uses the same triple representation, which is obtained using parse trees, and uses WordNet to improve the matching. FREyA [35] is another example, which takes a question as input and turns it into an ontology annotation (class, instance, property or literal). As opposed to previous researches, this system aims for a deeper

understanding of a question's semantic meaning using semantic trees generated by the Stanford Parser [71]. In case the system does not find the definitive answer, suggestions are generated to be presented to the user. By using this feedback loop, the system can solve the ambiguity of the question and improve its performance.

From a conversational search perspective, using domain-specific question answering techniques is impractical, given a CSS should be able to handle open-domain dialogues. However, these techniques can be used as part of larger open-domain systems in order to improve their performance. IBM Watson [48] is a well-known example, given its participation in the Jeopardy! contest. While the system is considered to be open-domain, some of its subsystems are taking advantage of structured data, such as ontologies. Kalyanpur et al. [66] used YAGO (Yet Another Great Ontology) [125], which has over 100,000 concepts, and other ontologies included in DBpedia [7]. The structured data is used for several tasks, such as detection of spatial ("This port is close to ...") or temporal relations ("Fire was discovered before the birth of... ") and answer type coercion. For this last task, the DeepQA [48] uses a multitude of TyCor algorithms [96] to verify whether a candidate answer belongs to the same type as the original question. Firstly, the algorithm performs Entity disambiguation and matching (EDM) [96] in order to map a potential answer to a resource in YAGO. Afterwards, predicate disambiguation and matching (PDM) [96] is being performed in order to identify the lexical types that are compatible with the input question. This task was evaluated on 3508 Jeopardy! questions and an improvement of up to 8% was observed when the structured data was used.

2.3.2 Open-domain Question Answering

One year after the addition of the QA track as part of TREC-8, we can already observe several notable open-domain QAS. Abney et al. [1] created a system that receives a natural language query as input and outputs a ranked list of potential answers. In the first stage, the system acts exactly as an IR system, given that it returns a list of passages from the top-ranked documents. To retrieve the top documents, a customized variant of the SMART system [113] was used. Afterwards, the passages were scored depending on the sum of IDF weights of consecutive words that it has in common with the user's query and the number of common bi-grams. In the second stage of the system, potential answers are extracted from the output of the first stage and classified by type (person, location, quantity etc.). Furthermore, the category of the user's query is identified by comparing the terms composing the query against a list of predefined keywords (for example, the presence of *Who* shows that the answer is of type *Person*). In the end, in order to rank the possible answers, the system checks if there is a match between the category of the query and the type of the answer. The top-ranked passages are those in which entities from the answer appear multiple times. The system was the third-best run in the TREC-8 Question Answering Track [138] for responses of at most 50 bytes with a Mean Reciprocal Rank (MRR) of .356 and the 2nd best run in the category with responses of at most 250 bytes with an MRR of .545.

Although very crude when compared to modern systems, it paved the way for other improved approaches. Harabagiu et al. [57] followed approximately the same system

[7] https://wiki.dbpedia.org/

architecture. However, by adding several knowledge-based NLP techniques in the pipeline, such as (1) Named Entity Recognition (NER), (2) semantic classification of the question by using WordNet and other taxonomies and (3) phrasal parsing using the Brill Part of Speech Tagger [15], they were able to greatly improve the performance of the system. In the same TREC-8 competition, the system was able to obtain an MRR of .555 for the 50-byte run and .646 for the 250-byte run, placing on the 2nd and 1st place, respectively.

Another approach that can be explored when building a QAS is to take advantage of existing question-answer conversations, instead of processing large collections of data in order to obtain an answer. This approach has a stronger IR component and is similar to our system, as we also take advantage of an existing collection of conversations. Surdeanu et al. [128] constructed a ranking engine using a large collection of community-generated QA conversations from Yahoo! Answers [8], which covers a large array of topics, making it suitable for open-domain QA. Given an input query, the system should be able to provide a ranked list of potential answers and choose the highest-scoring one. To compute the score, several features are used: (1) the similarity between the query and the answer, computed using BM25 [107], (2) the probability that question Q is a translation of answer A using IBM Model 1 [16] (3) the frequency and density of question terms that are present in the answer and (4) the correlation between the query-answer pair and the log of a search engine corpus. The results show that there is an increase of MRR from 56.06 when using only BM25 to 64.65 when using all the mentioned features.

More recent QAS started to take advantage of the evolution of neural networks. Rao et al. [105] propose a system in which answer selection for QA is done as a pairwise ranking task. The model contains 2 LSTMs, each taking a (question, answer) pair as input and outputs a score that represents the semantic distance between the 2. One network receives a positive pair and is expected to return a larger similarity score while the other a negative pair and should return a smaller score. By this approach, the network should approximate a function that, in the end, should reward the positive pair better. The evaluation of the model showed state-of-the-art performance on the TrecQA dataset, achieving a Mean Average Precision (MAP) of .78 and MRR of .834. By looking at the evolution of QA as a field of research, it can be observed that the general performance of systems has steadily increased, generally due to the development of more complex models, that can capture more easily the complexity of textual data.

2.4 Dialogue Systems

A Dialogue System (DS) is a system that is specifically designed to converse with a human in a coherent manner. A dialogue system can be categorized by their purpose: task-oriented system, which is designed to help the user solve a specific task from a specific domain, and non-task-oriented (chatbots), which is focused on basic conversations with humans on various domains.

[8] `https://answers.yahoo.com/`

QAS Type	System	Year	Used techniques
Domain specific	BASEBALL [53]	1961	Transpose question into query that was used against structured DB
	LUNAR [144]	1971	Similar to BASEBALL, applied to analysis of rocks on the moon
	AquaLog [82]	2005	Convert question into triple to match with an entry in the ontology. Relies on feedback from the user for incomplete results
	PANTO [139]	2007	Convert question into triple by using parse trees & WordNet. Executes SPARQL query on ontology
	FREyA [35]	2010	Creates triple using semantic trees for deeper understanding of question. Relies on feedback (suggestions) for incomplete results
	PowerAqua [84]	2012	Similar techniques with AquaLog. Matching is done by merging information from multiple ontologies.
Open-Domain	AT&T QA	2000	Fetches top documents and extracts best passages based on question structure and category.
	SMUNLP1	2000	Similar to AT&T, but enriched with multiple NLP techniques (NER, POS tagging etc.)
	Surdeanu et al. [128]	2008	Fetches potential answers using IR and NLP techniques based on a Yahoo! Answers dataset
	Rao et al. [105]	2016	Joint ranking task using 2 LSTMs in order to distinguish between positive (question, answer) pairs and negative ones

Table 2.1: Overview of discussed QAS

2.4.1 Task-oriented Dialogue Systems

In task-oriented systems, the agent is trained to be specialized in a specific domain, such as online shopping [146] or disease diagnosis [142]. Usually, such systems are designed as a pipeline with multiple components, each having a specific role. Firstly, such a sistem must have a *natural language understanding* (NLU) component, which detects the user intent from the utterance and/or extracts word-level information through slot filling. The latter is the more challenging problem, as it usually implies semantically labeling each word in the utterance: the input is the sequence of words and the output is a sequence of slots/concepts. Deep convex networks [36] have been applied successfully for both tasks in [37] and, more recently, using attention-based Recurrent Neural Networks (RNNs) in [79], reaching an F1-score of 95.78.

The *Dialogue state tracker - DST* (or belief tracker) is another core component of a task-oriented dialogue system. It makes an estimation of the goal of the user after every iteration of the dialogue and maintains these states as a form of dialog progress. The Dialog State Tracking Challenge (DSTC) [143] is the most important corpus for building belief trackers and is responsible for the creation of a variety of statistical techniques, such as rule-based approaches [140] or maximum entropy models [76]. More recently, Mrkšić et al. [95] proposed a neural model called Neural Belief Tracker, in which NLU and DST tasks are being using jointly with a DNN or a CNN. Their model achieved state of the art performance without relying on any hand-crafted

semantic lexicons. The next core component is tightly coupled with the state tracker and is handling *policy learning*, which predicts the next action based on the current state of the dialogue. For example, in an online shopping situation, if the previous state of the dialogue was "Recommendation", then the corresponding policy is triggered. Usually, policy learning is done via supervised or reinforcement learning and, more recently, through deep reinforcement learning [32], where the system obtained double the performance of a supervised learner in a negotiation scenario against bots.

Finally, a Dialogue System requires *natural language generation* in order to translate an action into a natural language utterance. Traditionally, this is achieved using *sentence planning*, in which semantic symbols are first processed using tree-like structure or templates and then converted into the final utterance via surface realization. The entire described process can be observed in Figure 2.6. An early system that implements the entire process is detailed in the work of Stent et al. [124]. As in the case of the other components of the pipeline, in more recent times, neural architectures started to be designed for natural language generation tasks. Dušek and Jurcicek [42] used a sequence-to-sequence (seq2seq) generation technique, combined with beam search and a n-best list re-ranker, all based on RNN. This allowed the creation of both natural language sequences and deep dependency trees from the input.

Throughout the research, we will see many elements from these task-oriented DS being used in conversational search. However, they are not equivalent, as a CSS is focused on the search component and is designed to handle an open-domain environment, where the user can drastically change its goal throughout the conversation.

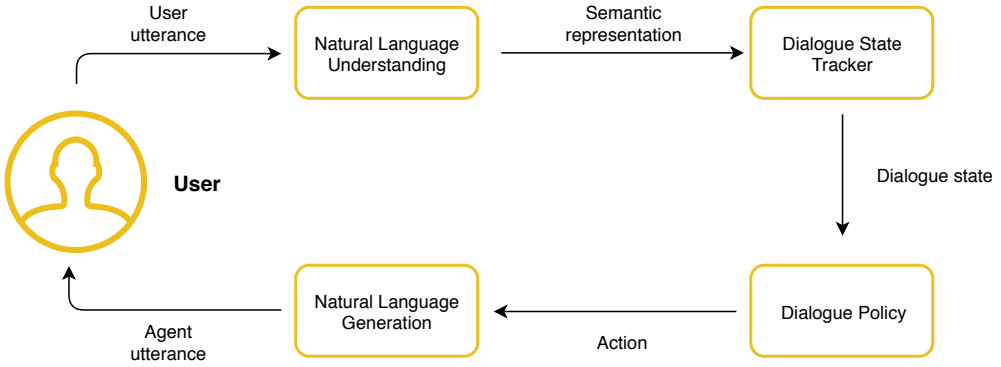


Figure 2.6: Visualization of a typical task-oriented DS

2.4.2 Non-task-oriented Dialogue systems

On the other side of the spectrum of dialogue systems are the non-task-oriented ones (also known as chatbots). Unlike their task-oriented counterparts, these systems focus on open-domain conversations and less on fulfilling a specific task. These systems can be implemented by 2 methods: generative or retrieval-based. The former has the power to generate responses that have never appeared in the conversations before, while the latter is usually better in environments when information is important.

When using *generative methods*, one of the foundations required to build generative models is the sequence-to-sequence model (*seq2seq*) [129]. The architecture accepts an input sequence $X = (x_1, x_2, ..., x_n)$ and outputs a sequence $Y = (y_1, y_2, ..., y_n)$

such that $p(y_1, y_2, ..., y_n | x_1, x_2, ..., x_n)$ is maximized. More specifically, the model follows an *encoder-decoder framework*: the encoder handles reading X and transforming it into a *context vector c* through an RNN (usually LSTM or GRU), the decoder estimates the probability p of Y by using input c. The method can also incorporate an attention mechanism that conditions each word in Y on a different context vector c. An example of a research that uses this model is [120], who took advantage of an encoder-decoder framework for short conversations on Weibo data [9]. The system was able to generate grammatically and contextually correct responses in 75% of the cases. Tian et al. [133] provide a very insightful empirical study, in which they examine the impact of context information on the performance of DS using generative methods. They have found that hierarchical models, which use a model for utterance-level information and another for the inter-utterance (context) level are superior to standard models, which use a simple model to capture the entire context information. Moreover, they propose a model which applies weighting to the context vector c depending on an attention score that models the context-query relevance. The newly created model outperforms all the baselines that were under revision. To conclude, although this approach has the power to generate new responses, a system built using these approaches cannot be considered by itself a CSS, as usually these kinds of systems are focused on simple conversations (such as chit-chat) and they don't have the necessary grounding to solve complex information needs.

Retrieval-based methods focus on choosing the most appropriate response from a pool of possible responses. In other words, a model that implements this method performs matching between a query and a response. Initially, retrieval-based methods focused on single-turn conversations, when the agent would have to match only one query. Deep Neural Networks(DNN)-based models [86] focused on co-occurrence of words between the 2 sequences on a local level (*infection-antibiotics* are likely to co-occur). Afterwards, these local interactions would be combined into a hierarchical structure in order to detect words that are semantically close (*Travel in Paris* and *Travel in Berlin* can be combined into *Travel*). In recent years, using *multi-turn conversations* has become more popular, as neural architectures now have the power to capture more context. Moreover, in multi-input conversations, the responses seem more natural as they are based on more background context. As we will see in the rest of this research, these types of systems come very close to CSS. One example is the work of Lowe et al. [85], which built the Ubuntu Dialogue Corpus, a large dataset of multi-turn conversations built using Ubuntu chat logs. This dataset was an essential starting point for the creation of the our dataset. To test the performance of the dataset, the authors use two neural models (RNN and LSTM) on a context that is built using all the previous utterances and the candidate response under analysis in order to compute the matching score between the response and the previous context. A later research [145] leveraged the same dataset in order to do response matching. Most of the researches until then would encode the entire context into a vector from the start and then perform the matching. However, starting from such a coarse granularity implies some degree of information loss. For this reason, the authors propose a new matching model which constructs a context vector for each utterance and performs the matching for each such vector. More specifically, the model creates a word-word similarity

[9] http://ntcir12.noahlab.com.hk/stc.htm

matrix (using word embeddings) and a sequence-sequence similarity matrix (using the hidden states of a GRU) for each (utterance, potential response). The information is then fused into a final vector by a series of convolution and pooling operations. The new model achieves significant improvements over the baselines tested by the creators of the Ubuntu Dialogue Corpus (increase in MAP from 0.485 to 0.529 and in MRR from 0.527 to 0.572).

DS Type	System	Year	Used techniques
Task-oriented	Wei et al. [142]	2018	Use Markov Decision Process (MDP) and Reinforcement Learning to build a DS to converse with patients collect symptoms based on a dataset collected from a medical forum
	Yan et al. [146]	2017	General framework for building a DS for online shopping, which can assist clients in completing purchases, searching products or answering questions.
	Cuayáhuitl et al. [32]	2015	Applies Deep Reinforcement Learning (DRL) policy in a strategic board game, where artificial agents negotiate over resources. The system learns to offer and reply by interacting with the other agents.
Non-task-oriented	Shang et al. [120]	2015	Uses Encoder-Decoder Framework with attention using a GRU model to *generate* appropriate responses to question. Based on data dump from a Chinese microblogging website.
	Lu and Li [86]	2013	*Matches* questions with appropriate responses based on co-occurence of words at a local level and by leveraging a hierarchy of local decisions to capture the semantic similarity.
	Wu et al. [145]	2017	Performs response *matching* by building by context vectors of various granularities by performing the matching for each utterance in the context.

Table 2.2: Overview of discussed DS

2.5 Conversational Datasets

This section is dedicated to researches that have built datasets that are intended for use in systems that share many properties with CSS. For each example, we will show the main characteristics, the methodology and the tasks that were performed using the dataset in order to understand what kind of tasks could be appropriate for out dataset.

The *Shaping Answers with Rules through Conversations* (`ShARC`) [112] is a conversational dataset built for machine reading purposes. It contains conversations that relate to a rule or regulation regarding traffic, benefit programs or grant descriptions ("Do I need to carry on paying National Insurance?"). In order to obtain these conversations, annotators received a scenario that described the context ("I am working for an employer in Canada"), the regulation that applied and the history of current follow-up questions and an answer. The task of the worker was to either pose the first, under-

specified question, respond with "Yes" or "No" to an existing question or emit a new follow-up question if the answer is not yet straightforward (for example, if the rule requires at least 1 of 3 characteristics and the response regarding the first one was "No", the user should ask details regarding the second characteristic that would fulfill the rule). To control the quality of the annotations, the authors enforce two mechanisms. First, a system agent is inserted during the conversation to respond with "Yes" or "No" in order to control the ratio of answers. In cases where the rule is in the form of *condition1 OR conditions 2*, users might tend to focus on the first condition and respond with "Yes" directly, which will immediately end the conversation. Secondly, some annotators will receive only parts of the ongoing dialog, which implies that, again, the annotator should find it hard to give a definitive answer. The resulting dataset contains 32436 utterances spanning over 948 rule texts. However, given that the answers to the questions had to be answered on a Yes or No basis, this implies the conversations have no grounding and no utterance intents (apart from *Follow-up Question*). Furthermore, all conversations originate from the same main domain.

`CoQA` [106] is a conversational Question Answering dataset obtained by splitting 8,000 conversations into 127,000 questions with answers. Starting from an evidence (" The Virginia governor's race"), pairs of questions and answers (Q_1,A_1),(Q_2,A_2) are derived ("What are the candidates running for?" - "Governor" or "Where?" - "Virginia") based on the conversation history. At test time, the trained model does not receive any evidence and has to build the entire construct by itself. The dataset is built using crowd-workers and the collection process is both verified by inter-annotator agreement and self-moderation (one annotator can flag the question of another as being vague or the answer as incorrect). Disagreements can also be discussed in a separate chat window. Furthermore, some annotators are integrated into a game where they receive a question that was already answered and have to predict an answer. The data collection spans over 7 domains, such as *Literature* or *News*. Several conversational and reading comprehension models were benchmarked against human judgements for response generation and response selection tasks and found that the best model is 23.4 points behind human judgements, which obtained an F1-score of 0.888. As the previous dataset, `CoQA` is designed for QA tasks and cannot be transposed to a CSS. However, as opposed to `ShARC`, the dataset is more scalable and the conversations originate from multiple independent domains.

`Frames` [7] is a conversational dataset designed for task-oriented DS and contains 1,369 dialogues with a mean of 15 turns per conversation. The data collection was performed using 12 participants in pairs of two over a period of 20 days. To gather the data, the authors used a Wizard-of-Oz approach [67], in which one of the conversation participants acts as a DS: has access to a search interface connected to a database and, given a user query, it decides what to say next. The conversations occurred on Slack[10], where each participant was allocated a task template ("Find a vacation between [START_DATE] and [END_DATE] for [NUM_adults] adults. You leave from ...") and an available "wizard". The values were either populated using the database or by using random numbers that were not stored, meaning that some of the tasks were not achievable. In these cases, the agent would have to recommend alternatives that would fit the constraints of the database. The conversations were then split into frames

[10] `https://www.slack.com`

(states), which summarize semantic information regarding the conversation history, such as constraints, user requests, user comparison requests. After obtaining the conversations, the data was annotated with various labels: slot types (such as database fields - START_DATE or global properties, such as the PRICE), slot values, references to other frames, dialogue acts (which is equivalent with our definition of user intents). The trained models are used for slot filling and frame tracking (predicting the next frame based on the current history) tasks. While this dataset contains an extensive list of annotations, it lacks any form of response grounding, is focused on a single domain and is not large-scale enough to be used for a CSS.

The *Multi-Domain Wizard-of-Oz* (`MultiWOZ`) [17] is another example of a dataset oriented towards task-oriented DS. At a size of 8,438 conversations and 113,556 utterances, it is significantly larger than `Frames` and spans conversations over seven domains. The data collection process is performed by using a custom interface that connects human wizards to users in a similar fashion to `Frames`. However, as opposed to the previous dataset, crowd-workers are employed instead of expert annotators, which allowed the authors to gather a much larger amount of data. The rest of the collection process is similar to `Frames`, with the users receiving a template and the wizards the means to solve the information needs (if possible given the information in the database the wizard has access to). However, in an effort to mimic an actual conversation, users were gradually given parts of the template, in order to solve sub-goals one by one. The models trained on this dataset were evaluated on two DS specific tasks: dialogue management and response generation. Although it solves some of the issues that the previous dataset was having regarding the requirements of a CSS, the variety of domains is limited (all of the seven domains are related to travelling) and there is still a lack of grounding for responses.

Wizard of Wikipedia (`WoW`) [39] is an example of a dataset that is at the crossroads between task-oriented DS and non-task-oriented DS: while the conversation setup is based on a Wizard-of-Oz approach, the user does not have a clear information need from the start. In the beginning, the user is a "curious learner" which talks freely to the wizard. The goal of the conversation is to have a detailed discussion about a chosen topic that interests one of the participants, while also keeping the conversation "engaging and fun". Instead of a small-scale database, the wizard has access to an interface that retrieves articles from Wikipedia that can be relevant to the given topic under discussion. The final dataset contains 22,311 dialogues with a total of 201,999 utterances spanning over 1365 domains. The models evaluated on the dataset are meant to replace the wizard in the dialogue. This process happens in 3 stages: *knowledge retrieval* - given a large knowledge base, the model should be able to retrieve the first paragraph of the top 7 articles for a given query, *knowledge attention* - choose what sentences should be used to create the next utterance in the dialogue, *utterance prediction* - given the model performed the previous 2 tasks, the final stage should predict what should be the next utterance in the dialogue. Humans were then asked to rate the performance of each model, along with rating actual conversations that occurred between people on a scale of 1 to 5. The best performing model, which combined memory network architectures [126] for knowledge retrieval with Transformer architectures [136] for generating outputs, obtained an average rating of 3.43, as opposed to 4.13 for human conversations. Although this dataset is truly open domain and large-scale, the lack of utterance labels, response grounding and clear information need from the start of the

conversation makes it incompatible with a conversational search dataset.

The *Ubuntu Dialogue Corpus* (UDC) [85] is a dataset extracted from various logs from Ubuntu-related IRC chat rooms. It contains almost 1 million multi-turn dialogues, spanning over 7 million utterances composed of over 100 million words. It is designed to be used for building DS based on neural network models, which can take advantage of the significant amount of unlabeled data. It contains conversations between humans interacting on the platform, with a minimum number of 3 turns and an average of 7.71 and an average number of 10.34 words per utterance.

To identify the dialogues, each message is identified by a unique (time, sender, recipient, utterance) tuple. Afterwards, conversations are formed by matching senders and recipients. Recipients are identified by checking whether the first word of the utterance matches a username that was recently active in the chat. To track the initial question, the algorithm starts from the first response of a user, identifies a recipient and considers the initial question as the most recent utterance of the recipient. In case multiple people respond to the same question, each of them is treated separately as an independent dialogue. In order to test how neural architectures perform on this dataset, the authors benchmarked the performance of an RNN and an LSTM model against a simple TF-IDF baseline on the task of selecting the next best response after an utterance. To build the training dataset, for each conversation, all possible *conversation contexts* of length at least three are generated (a context is the entire conversation history up to the current utterance). Therefore, for a conversation with five utterances, there are three conversation contexts. In the testing phase, the system has access to a context and is required to construct the next best response for it. Furthermore, to make the task more difficult, negative responses are also added and flagged accordingly in the training set. Using various recall measures, the researchers discovered that LSTM outperforms all the other architectures on all metrics. They have also noticed that the performance increases continuously with the increase of the training size, which shows the importance of having a large-scale dataset.

Although this research provided a large-scale dialogue dataset, designed to be used with advanced neural architectures, it does not have all the necessary properties to be used for a CSS: the dataset does not have any conversations with intent labels (therefore, the system cannot detect what kind of intent the user has in the current context), it is specific to only one domain and the responses have no grounding.

MISC (*Microsoft Information-Seeking Conversation*) [132] is a dataset designed for Conversational Information Retrieval that includes audio-video recordings and their transcripts, recordings of search and other computer-related actions and surveys on the emotions and efforts felt by the users during the conversations. The conversations occur between pairs of volunteers. One of them is a "seeker", who is given an information need but no means to answer it directly (no internet connection) and an “intermediary”, who has the means of finding any answer (via a computer connected to the Internet) but does not know the task. Each pair had to communicate through an audio link and had 10 minutes to solve each task. Each participant had to complete a questionnaire to assess the personality traits in order to have a reference against which to analyze and interpret the results. During the tasks, the screens were recorded in order to understand what parts of the page were under supervision and which queries were used. A camera was also tracking the faces in order to capture eye movement and emotions. In the end, each volunteer was questioned with regards to the necessary effort and the

level of engagement. In the end, the experiments generated 88 conversations (only 42 were completed, the others were skipped after 10 minutes), with 857 words per task on average and an average solving time per task of 8 minutes 20 seconds. As compared to UDC, it can be observed that the mean number of words used is roughly 10 times higher on average, which is an indication of the fact that people are much more expressive when talking than when writing.

The purpose of this research was to understand how an ideal natural conversation should look like. Moreover, it can be used to understand how the relationship between the agent and the user evolves throughout the process of searching, given the fact that the dataset also provides emotion data. However, this dataset is not appropriate for building an actual CSS, as it does not have utterance labels and the answers are not grounded. Moreover, given the size of the dataset is so small, it cannot be used for training any kind of complex conversation-oriented neural architecture.

Trippas et al. [134] conducted a study similar to MISC, where 13 pairs of people would try to solve a task together in at most 10 minutes: one received the task but had no means to solve it while the other did not know the task but had access to a search engine. Their screens were also recorded and transcripts were created. However, as opposed to the previous research, annotators were asked to label each turn of the conversation depending on the intent, such as *Query Repeat* or *Intent Clarification*. The tasks were also split by complexity and the results showed that the more complex a task would become, the longer the length of the utterance would become. The analysis has also shown that the users who knew the task needed less time per turn that the ones who were asked to use the search engine and that, as the complexity of the tasks grew, the number of times the search engines was used increased. The result of performing the experiment is a dataset that contains only 39 conversations. Due to the small size of the dataset, the purpose of the study is to further understand how humans behave during the search process. Moreover, apart from the insufficient size, the responses of the agent are not grounded and, thus, cannot be used for building a CSS.

One of the researches that constituted a solid starting point for ours is the one performed by Qu et al. [101], which resulted in the creation of the MSDialog dataset. It contains 35,000 dialogues (totaling 300,000 utterances) between users with an information need and people that can provide answers on an online forum regarding Microsoft products [11]. All conversations have at least three turns and at most 10, with a mean of 8.94 and a mean utterance length of 75.91 words. As opposed to UDC, dialogues can have more than two participants, with an average of 3.18 per dialog. Every conversation that was extracted had to have at least one correct answer, which was identified using the forum community, which is able to vote on which is the correct answer. Furthermore, a subset of 2199 dialogues totaling 10020 utterances was labeled using 12 distinct intents by employing Mturk [12] crowdsourcing workers. The intent classes were built starting from the taxonomy of Bhatia et al. [13] and encompass a large spectrum of intents, such as greetings or gratitude, information requests, requests for further details or positive/negative feedback. Most of the classes that were defined in this research were also used in ours. In a following research, Qu et al. [102] took advantage of MSDialog to perform user intent prediction. Firstly, the conversations

[11] https://answers.microsoft.com/

[12] https://www.mturk.com/

were transformed into multiple types of features: (1) content features by using TF-IDF to obtain similarities between a response and the initial utterance in order to capture the relevance, (2) structural features, such as the position of the utterance or the length and (3) sentiment features in order to better identify certain types of intents which are strongly correlated with sentiments, such as Positive/Negative Feedback or Gratitude.

The authors propose two neural architectures to solve the tasks: a CNN and a BiLSTM model, which take as input an utterance from the dialogue, transforms it into word embeddings using a pre-trained model and outputs one or more intent labels. The architectures can either take advantage of only the features that were mentioned previously or, additionally, can incorporate context representation of the dialogue up until the utterance under supervision and use it as an additional feature to further understand the importance of the current utterance. The experimental results show that the CNN model with context information and representation achieves superior results to all the other baselines (Random Forest, AdaBoost and standard CNN and BiLSTM) in terms of Accuracy, Precision, Recall and F1-Score. Furthermore, the authors evaluated how the best model trained on MSDialog performs on the UDC annotated subset. The results in Table 2.3 show that, although the generalization performance is lower than for MSDialog, the model still outperforms the baselines.

Dataset	Model	Accuracy	Precision	Recall	F1
MSDialog	Random Forest	0.6268	0.7657	0.5903	0.6667
	AdaBoost	0.6399	0.7247	0.6030	0.6583
	CNN-Context-Rep	**0.6885**	**0.7883**	**0.6516**	**0.7134**
UDC	Random Forest	0.4405	0.**6781**	0.4077	0.5092
	AdaBoost	0.4430	0.5913	0.4187	0.4902
	CNN-Context-Rep	**0.4708**	0.5647	**0.5129**	**0.5375**

Table 2.3: Performance of various models on MSDialog and UDC, as reported by Qu et al. [102]

Although many researches have proposed datasets that are appropriate for various conversational-related tasks, none exhibit *all* of the characteristics necessary to build a CSS. Furthermore, the ones that demonstrate most of the features are too small to train any model. Table 2.5 contains a list of datasets that are designed to solve tasks that are related to conversational search, ordered by the number of CSS characteristics they fulfill. Table 2.4 showcases some of the tasks that can be accomplished using each of the mentioned datasets.

Name	Task
ShARC [112]	Next utterance prediction, Follow-up Question Generation
CoQA [106]	Reading comprehension
Frames [7]	Dialogue State Tracking, Natural Language Generation
MultiWOZ [17]	Dialogue State Tracking, Dialogue-Context-to-Text Generation, Dialogue-Act-to-Text Generation
WoW [39]	Next utterance prediction
UDC [85]	Next utterance prediction
MSDialog [101]	Next utterance prediction, User intent prediction
MISC [132]	Information Seeking task
SCS [135, 134]	Search task with different levels of cognitive complexity (Remember, Understand, Analyze)

Table 2.4: Overview of the tasks that were performed on each dataset

Name	Description	Size	Strengths	Weaknesses
ShARC [112]	Dataset focused on follow-up questions	948	Multi-turn, follow-up questions	Simple answers - Yes No, single-domain, no grounding
CoQA [106]	Long conversations split into QA pairs	8,000	Long conversations, multi-domain	QA only, simple information needs, no grounding
Frames [7]	Goal-oriented DS dataset	1,369	Multi-intent, complex information needs	Not scalable, single domain & no grounding
MultiWOZ [17]	Conversational corpus	8,438	7 domains, multi-intent, multi-turn, utterance labels	No grounding
WoW [39]	Open-domain conversational corpus based on Wikipedia	22,311	Scalable, truly open-domain, multi-turn, grounded.	"Chit-chat" oriented - no clear information need from the start, no utterance labels
UDC [85]	Large conversational dataset based on AskUbuntu [13]	930,000	Very large, multi-turn, complex information needs, follow-up questions, multi-intent	Single-domain, no grounding, 80% of conversations have artifacts [73], no utterance labels
MSDialog [101]	Large conversational dataset based on Microsoft Products forum [14]	35,000	Large, multi-turn, complex information needs, follow-up questions, multi-intent, utterance labels [a]	Single-domain, no grounding
MISC [132]	Recordings of conversations between humans	88	Multi-turn, complex information needs, follow-up questions, multi-domain	Very small, no grounding, no utterance labels
SCS [135, 134]	Study on human interactions during conversations	39	Possesses most of the characteristics necessary for a CSS	Very small, no grounding

Table 2.5: Overview of dialogue datasets including their size and main characteristics.
[a] *There are labels for a sample of 2,199 dialogues.*

Chapter 3

Building the dataset

As seen in Table 2.2, existing conversational search datasets usually lack coverage of multiple domains, do not address complex information needs or lack grounding of the answers present in the agent responses. A significant portion of this research is dedicated to the creation of a new dataset that can address the issues that are currently present in conversational datasets. In order to create a large-scale collection of data, we must resort, as some of the creators of the datasets in Table 2.2, to using existing data sources, rather than using crowdsourcing to artificially create new conversations. We have chosen the community-driven question-answering platform Stack Exchange [1] as our data source for extracting conversations due to the following reasons:

- The data dump is publicly available [2].
- It is large-scale - over 20M Questions [3].
- It covers various domains (so-called *sites* - 174 as of 11/2019).
- The information needs are complex as, usually, users who resort to asking a question on one of the sites have not been able to find the desired information via a simple web search.
- The platform is self-moderated, which means that people can rate comments and answers, which makes filtering of spam and offensive discussions much easier.
- The built-in interface allows extended interactions through comments on the same answer.

For our dataset, we consider 14 diverse domains: apple, askubuntu, dba, diy, electronics, english, gaming, gis, physics, scifi, security, stats, travel and worldbuilding. We have chosen the sites based on the topic (we aimed for a high diversity) and their relative size (we filtered out sites that had archive files smaller than 100MB, which was an indication of a low number of conversations). The number of domains was restricted by the time and resource constraints, given that adding new domains requires

[1] `https://stackexchange.com/`

[2] `https://archive.org/details/stackexchange` data dump from 2019-03-04

[3] According to `https://data.stackexchange.com`

some manual annotation work. However, the code [4] for automatically extracting the dataset is publicly available and can be easily modified to include any of the 174 available domains.

3.1 Inclusion criteria

Usually, on a single question-answering thread on Stack Exchange, there can be multiple potential answers. Each answer has its own comment space where the discussion can be extended further. We treat every such answer space as an independent conversation, which occurs between exactly one *information seeker* and one *information provider*. A conversation is included in the dataset if:

- The conversation takes place between exactly two users, without any interference from anyone else. Conversations with three or more participants are not allowed. Kummerfeld et al. [73] have found that in the Ubuntu Dialogue Corpus [85] (which derives two-way conversations from chat rooms involving multiple people) 80% of the conversations have either missing or extra messages, with only 48% of them having explicit direct mentions. Therefore, by focusing on conversations that only occurred between two users, we discard the possibility of wrongly separating the conversations into sub-threads and, therefore, remove the need for applying conversation disentanglement [73].

- The conversation consists of at least two utterances per user. Conversations that are shorter than two utterances per user are considered fit for question answering, rather than conversational search.

- *At least* one of the responses originating from the information provider contains a hyperlink (thus, providing grounding for the answer).

- The conversation has not been marked as *Spam* or *Offensive* by one of the users of the platform.

- The conversation has not been marked as *Edited* or *Deprecated*. This is usually marked in a certain manner in the conversation (e.g *EDIT:* or *DEPRECATED:*). Discarding these types of conversations can avoid artifacts in the dataset, such as answers that are placed directly in the original question after edit.

- If the final response of the conversation is coming from the information seeker, it must express positive feedback.

The rationale behind the last condition is that, if the final response is coming from an information seeker and is not a positive feedback, this means that the conversation is either ongoing or the seeker did not fulfill his information need. Therefore, these conversations do not qualify for being added to the dataset. In order to verify whether the last response from the information seeker was positive, we sampled from each domain a total of 1400 conversations (100 conversations per domain) where the

[4] https://github.com/alexanderblnf/conversational-search-dataset

last utterance originated from the information seeker and labelled it as positive feedback/other. Subsequently, we computed the VADER score [62], which returns a value between -1 and 1 based on the polarity of a given text. A low score indicates a negative sentiment, while a high score is indicative of a positive one. Based on the labels and the score, we used a decision stump to discover what is the optimal VADER score threshold (per site) to separate the positive feedback from other formulations. Based on the newly obtained thresholds, we discarded all conversations that had the score under that value.

In order to certify to what extent the presence of a hyperlink is a valid indicator for document grounding, we have sampled 150 conversations from the dataset and manually verified whether the link was indeed pointing towards a document that is related to the answer. This was the case for 88% of the samples, which we consider to be a sufficiently high percentage to stop improving upon the grounding logic.

3.2 Initial dataset analysis

Based on the aforementioned criteria, we were able to extract a total of 80,326 conversations, spread across 14 sites as illustrated in Figure 3.1. Tech-related domains occupy the top three domains by size, with *askubuntu* being the biggest. What is also noticeable is the fact that the weight of each domain in the dataset is generally proportional to the size of each site (by the number of questions) - for example, askubuntu is the biggest domain while worldbuilding is the smallest. Our filtering criteria was quite stringent, as only 4.77% of question threads were transformed into conversations that qualify for our dataset.

To understand how these sites have evolved over the years, we have looked at the number of conversations generated by each site every year (we consider the beginning of a conversation as being the timestamp at which the agent first responded to a question), with the exception of 2019, for which we do not have the complete data. By analyzing Figure 3.2, it can be observed that, since its release in 2009, the platform has increased substantially in terms of the number of conversations. For a large part of its history, *askubuntu* has been the biggest site of the 14. However, with the appearance of more sites, its share of the conversations has been steadily decreasing, with *electronics* becoming the largest generator of conversations.

When analyzing the average size of an utterance by site in Figure 3.3, we have noticed that domains that have questions that require extensive descriptions and responses are the ones that have the highest utterance size. For example, the discussions on *worldbuilding* revolve around designing fictional worlds, which, intuitively, cannot be summarised in a couple of sentences. Responses on *askubuntu* or *apple* tend to be short, as they are usually targeted towards concrete functionalities or commands.

The average size of a conversation can be indicative of the complexity of a question. Our analysis in Figure 3.4 has shown that the majority of conversations have two turns per user (60%). We have noticed that highly technical domains, such as *electronics* or *physics* tend to contain longer conversations (in terms of the number of turns), which can indicate that the agent might require further information before fully answering a question. On the other hand, questions related to domains such as *travel*

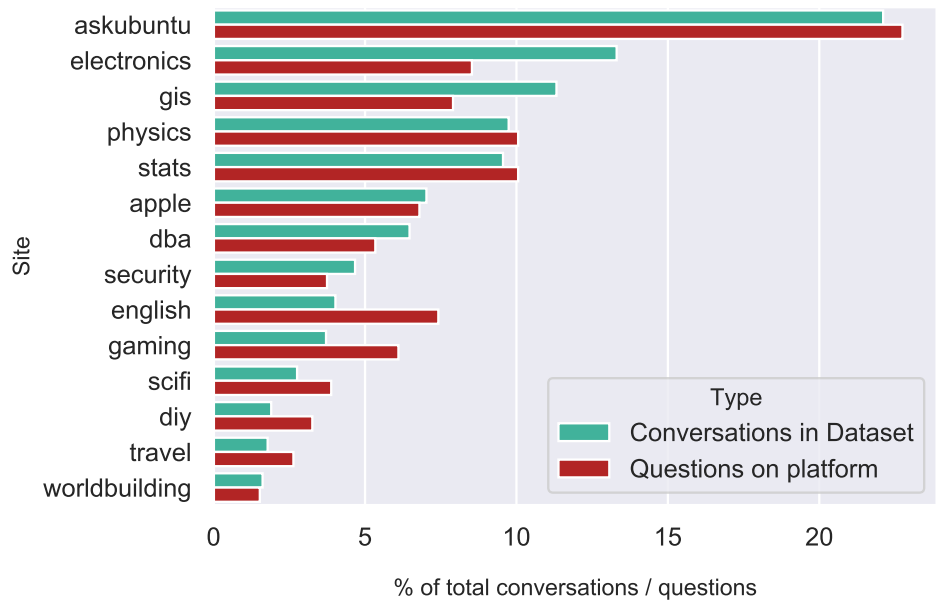


Figure 3.1: Distribution of conversations across the 14 domains. The green bars show the weight of each domain in terms of conversations in the dataset. The red bars show the weight of each domain in terms of number of questions that are present on the platform (as measured in September 2019).

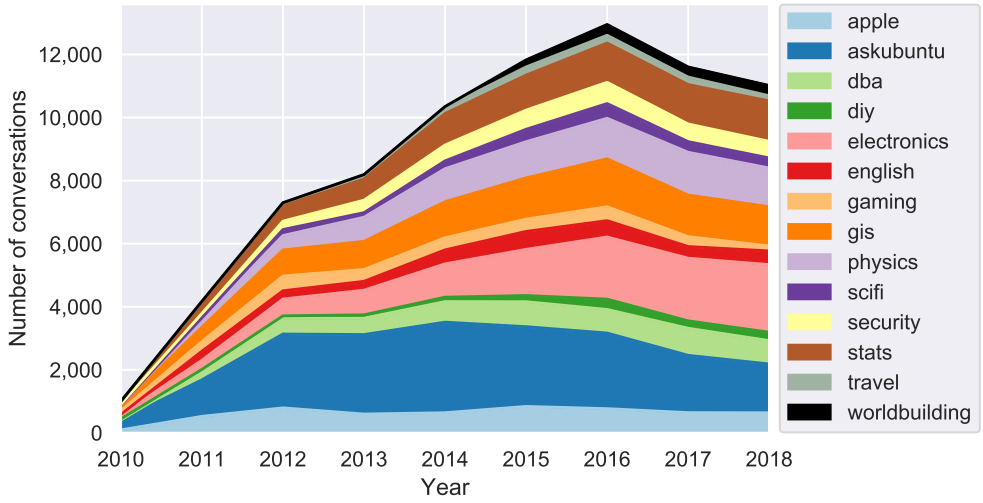


Figure 3.2: Evolution of the 14 domains throughout time in terms of the yearly number of conversations

or *english* might have more straightforward answers, given that the users require less turns on average to answer a question.

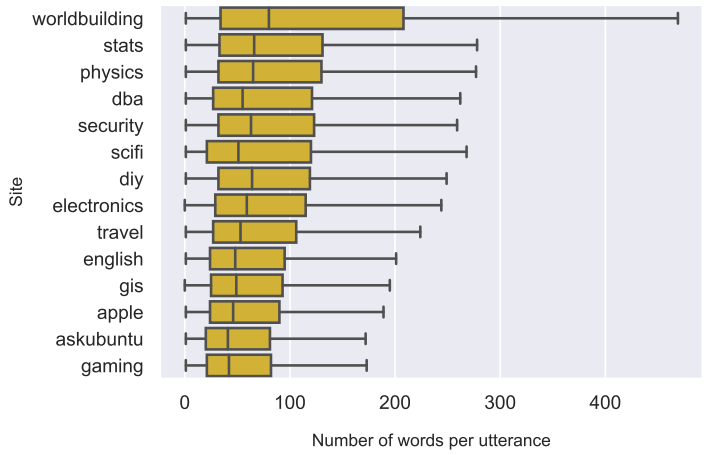


Figure 3.3: Visualization of the utterance length per domain. Each box displays the mean number of words (vertical line inside the box) and the confidence interval.

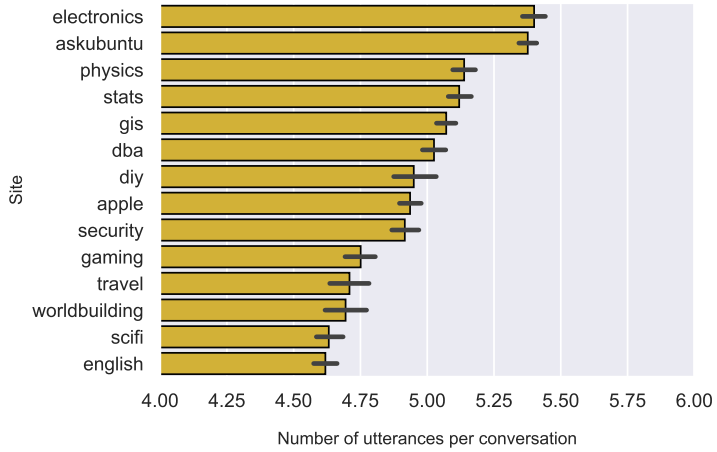


Figure 3.4: Visualization of the conversation size per domain. Each bar displays the mean number of utterances per conversation and the confidence interval

3.3 Utterance intent labeling

As previous research has shown [135], integrating underlying utterance information such as relevance feedback can have positive effects on the performance of the agent. In order to detect these types of *user intents*, we have sampled 1,356 conversations from `MANtIS`, covering the entire range of domains in accordance with their relative size. Each utterance in the conversation was manually labelled according to the intent of the user who emitted it, resulting in a total of 6,701 intent labels. Inspired by the work of Qu et al. [102], which performed the same labeling task, we have defined nine types of intent labels. These can either describe a question (*Information Request*, *Follow Up Question*, *Original Question*), an answer (*Potential answer*, *Further Details*),

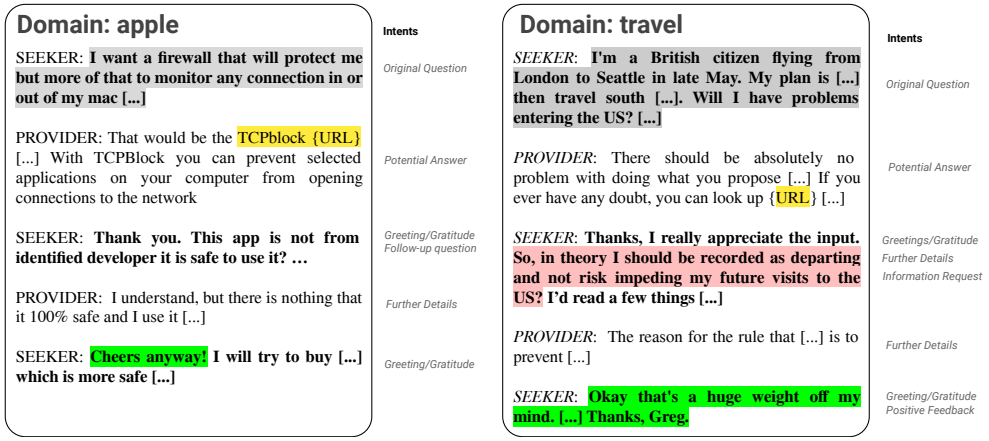


Figure 3.5: Dataset conversation examples. yellow shows document grounding, green displays final positive message from the original poster, pink are clarification questions and gray highlights initial information needs. All examples in the table are multi-turn. The right-most column shows the conversation labels for each utterance.

express a form of greeting gratitude (*Greetings/Gratitude*) or illustrate some form of feedback (*Negative Feedback*, *Positive Feedback*). Any other type of intent that does not fit in any of the aforementioned descriptions is labeled as *Other*. A more extensive description of each label, accompanied by an example, is available in Table 3.1

Category	Description	Example snippet
Original Question (OQ)	A user asks a question that initiates the conversation	Hello! I was wondering what is [...]
Further Details (FD)	A user provides more details.	The information you need is [...]
Follow Up Question	Seeker asks one or more follow up questions.	I really have one more simple question [...]
Information Request (IR)	A user asking for clarifications or further information.	Your advice is not detailed enough [...]
Potential Answer (PA)	A potential solution, given by the information provider.	To change the PIN on your phone, you [...]
Positive Feedback (PF)	Seeker provides positive feedback about the offered solution.	That was exactly what I needed. Thanks!
Negative Feedback (NF)	Seeker provides negative feedback about the offered solution.	[..] the fix did not work.
Greetings / Gratitude (GG)	A user offers a greeting or expresses gratitude.	Thank you for all the responses!
Other (O)	Anything that does not fit into the above categories.	:) :) :) . *shrug*

Table 3.1: Description of the intent annotation scheme.

As in Qu et al. [102], an utterance can be associated with multiple labels, given that one person can express multiple intents through the same message in a conversation. For example, a person can both show gratitude and give positive feedback when a response answers his question. A real example of such a conversation is presented in Figure 3.5. Initial labeling experiments consisted in employing *crowd-workers* to perform the task. However, the observed accuracy of crowd-workers when labelling pre-annotated gold standards was, on average, under 50%. Therefore, the procedure was then applied using two expert annotators, which performed the task on an interface that was built specifically for this purpose. To measure the annotator agreement, a subset of 151 utterances was reserved for both the annotators to label and were randomly inserted during the process, disguised as normal utterances. The metric that was used to measure the agreement is the Krippendorff α [72] and the resulting value was 0.71, which indicates a satisfactory agreement between annotators.

The distribution of intents in Figure 3.6 reveals that *Further Details*, *Potential An-*

swer and *Original Question* are the most common intents. These results are to be expected, given that every conversation should have one question and one possible answer. Moreover, *Further Details* should also occur in most conversation, since we are only considering conversations that have at least four turns, which means the information need was not solved immediately and further interactions were necessary. Furthermore, 20.5% of utterances were labelled with more than one intent and 18.7% of these were labelled with more than two intents. The most common combinations of intents were *(Further Details, Information Request)*, *(Further Details, Greetings / Gratitude)* and *(Positive Feedback, Greetings Gratitude)*. These facts confirm the multi-intent characteristic of our dataset.

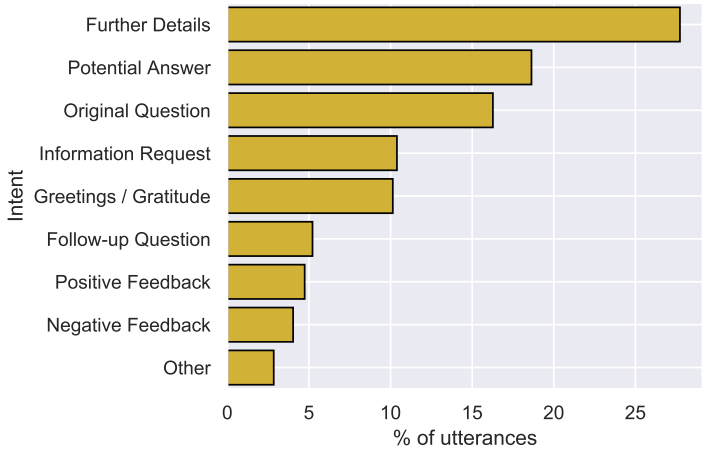


Figure 3.6: Intent distribution of the dataset. In total, there are 8334 distinct intent labels.

3.4 Grounding documents dataset

As explained by Baeza-Yates et al. [9] in their book on modern information retrieval, ad-hoc retrieval is one of the most common tasks that are performed by IR systems. This implies matching two pieces of text - typically a user query, which expresses his information need, and a target document. In conversational search, the user can express various information needs throughout the conversation and the agent should be able to provide responses, accompanied by a grounding document that explains the intuition behind the answer.

In its current form, `MANtIS` does not contain actual documents that solve the information need of the user, but rather agent responses which contain a link to a grounding document. For this reason, we have decided to enrich `MANtIS` with the actual grounding documents by building a system that crawls the URLs that exist in each conversation. Therefore, for each existing URL in the dataset, we attempt to extract the associated document. Given that links can point towards other types of files than text documents (images, zipped archives etc.), the crawler verifies the request header of each link to make sure that the destination URL is an HTML document and dis-

cards everything that is not. For the document extraction phase, we have used *newspaper3k*[5], a Python library specialized in extracting articles from links. Timeouts between requests are also enforced in order to make sure that our crawler is not flooding websites that appear often throughout conversations (for example, we have noticed it was quite common that users post links that indicate other StackExchange threads and that, throughout our initial experiments, we were temporarily blocked from accessing the website). Although the crawler is able to extract documents for all URLs available in the conversation, the current analysis and task will be performed only on the documents mentioned by the agent.

Out of a total of 197,293 URLs identified in agent responses, we have managed to extract documents for 58.8% of them. Out of the URLs we had associated documents for, 82.5% were mentioned in the first response of the agent, with 11.9% mentioned in the second one. This indicates that most of the grounding of answers occurs at the beginning of the conversation.

Further analysis provided important insight on the reason why our document extraction success ratio was relatively low: out of the total number of URLs for which we were not able to fetch documents, 47% had no extension or an extension indicating a web document (*.html, .htm, .asp*), which qualified them for web crawling, while 32.07% had an extension that indicated a PDF or image file, which are not compatible with our current system. The remaining 20% are composed of various types of files, such as archives (*.zip, .tar*), executables (*.exe, .dmg*) or file extensions specific to certain programming languages (*.py, .sh, .java, .js*).

At a domain level, we have noticed the most common domain for which we were unable to fetch URLs was by far imgur [6], a popular image sharing website, with a share of 20.5%. The next domain in terms of occurrences was Github[7] with a share of 1.79%.

Having these insights, we then proceeded to analyze how the share of each URL type evolved throughout the years by looking at the utterance time the link was included in. Figure 3.7 depicts how the distribution of un-crawlable URL types changed from 2009 to 2019. Initially, links that would qualify for our web crawler (indicate a web document extension or show no extension at all) were by far the most common (roughly 70% of the URL pool). However, as time grew by, images became the predominant type of URL for which we were not able to retrieve documents, reaching a share of almost 50% in 2019. In a report by the International Trademark Association[8] regarding Wayback Machine [9], a historic digital archive of the World Wide Web, they state that the typical lifespan of a web page, as measured by the online archive, is 44-75 days. This implies that for the first years, when the crawler-compatible links were the most common, there is a high probability that the crawler was not able to retrieve documents because the web pages did not exist anymore.

Since one agent response can contain multiple URLs, further analysis was necessary in order to discover how many conversations have at least one associated documents. We have found that for 74.36% of conversations we have managed to extract at

[5] https://pypi.org/project/newspaper3k/

[6] https://imgur.com/

[7] https://github.com

[8] https://www.inta.org/Advocacy/Documents/INTAWaybackMachine2009.pdf

[9] https://archive.org/web

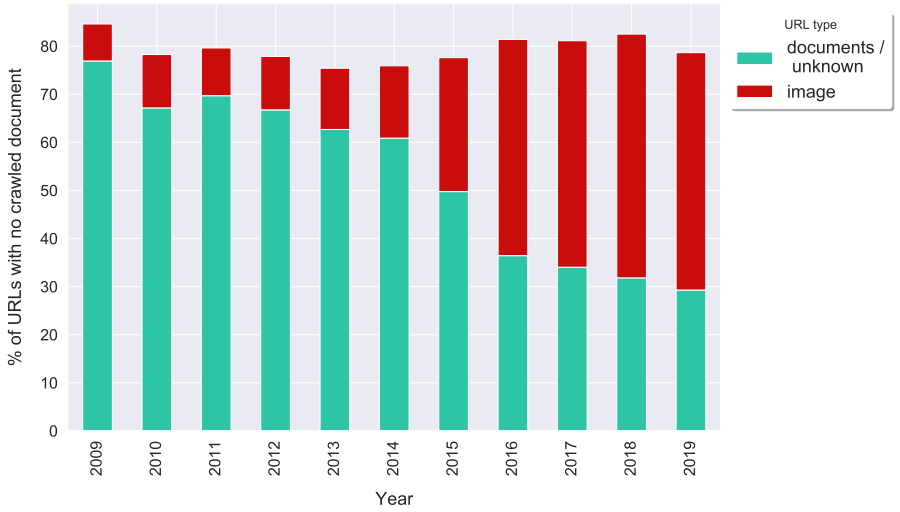


Figure 3.7: Distribution of un-crawlable URLs by type of file. The green bars show the share of images . The red bars show the share of web documents or URLs that are not identifiable by extension.

least one document. When analyzing the document sizes by site, we see significant differences between various domains. Given that *worldbuilding* had the largest utterances in terms of the number of words used, we expected that the grounding documents were also quite extensive. We expect the same behaviour for *scifi*, which can be considered a neighbouring domain given that both refer to fictional elements. The visualization in Figure 3.8 confirms the fact that sites with long utterances are also characterized by long grounding documents: the top six domains by mean utterance length are the same top six domains by document length.

3.5 Tasks

The dataset is suitable for a suite of conversational tasks: conversation response ranking (CRR) [148, 85, 106], user intent prediction [101, 102] and grounding document ranking (GDR).

3.5.1 Task definition

Before providing insights into our task setup, let us define each task in a more detailed manner.

Conversation response ranking Let $D = (C_i, r_i, y_i)_{i=1}^{N}$ be a conversational dataset containing N triplets. The first component is the conversation context C_i containing all of the utterances $\{u_1, u_2, ...u_t\}$ that appeared in the conversation up to time t, which is the time when an agent response occurred. In our experiments, we only consider conversation contexts that hold at least three utterances, as a conversation with 1 user question and 1 agent response is considered a QA task. For example, if a conversation spanned over eight utterances, three conversation contexts will be generated: one containing three utterances, one containing five (the previous three, together with

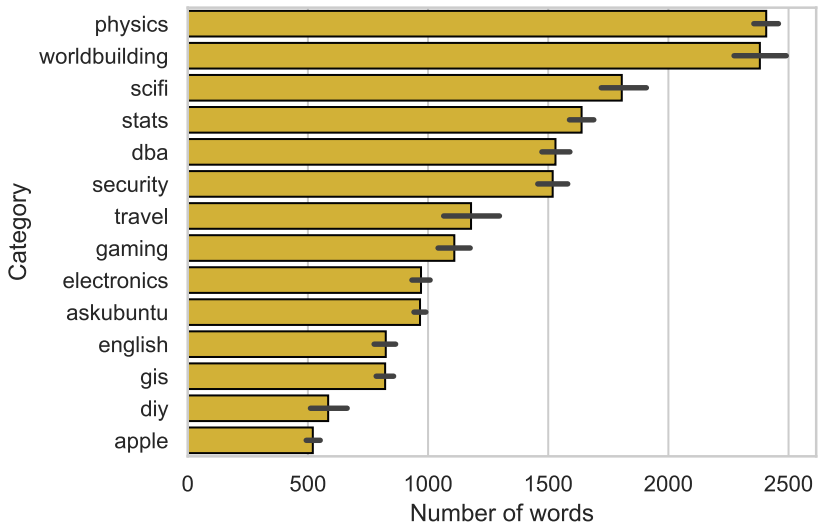


Figure 3.8: Visualization of mean document length by site. The width of the bar represents the mean, while the horizontal line represents the confidence interval.

the following agent response and a new utterance originating from the user) and one containing seven. The second component is response r_i, which can either be the first agent response that occurred after time t (also known as *true response*) if $y_i = 1$, or , if $y_i = 0$, r_i is a negative sample obtained by means of a sampling function. The task of a model is to learn a ranking function that can order a set of candidate responses (all the r_i that belong to the same C_i) by their likelihood of being the true response. An example depicting a conversation split by contexts and potential responses is depicted in Figure 3.9.

User intent prediction Based on a set of utterances u_i with the associated list of intent labels, a model has to learn to predict the intent(s) of an unseen example.

Grounding document ranking The GDR task definition is similar to CRR. Let $D = (C_i, d_i, y_i)_{i=1}^{N}$ be a dataset containing N triplets: a conversation context C_i, a document d_i obtained based on the URLs present in r_i if $y_i = 1$ or a negative sample if $y_i = 0$. The task of the model, similar to the CRR task, is to learn a ranking function that can order a set of candidate *documents* by their likelihood of being the true document.

3.5.2 Task setup

For the CRR task, we have designed 2 variants of the dataset: `CRR10` and `CRR50`. For each conversation containing n_a agent utterances, we generate $n_a - 1$ *conversation contexts*, as the case in which there is only one agent response is not taken into account due to the fact that a context with one question and one answer is considered appropriate for QA tasks. In total, we have generated 118,353 conversation contexts. We consider the last agent response after the current context to be the *true response* (the one that should always be ranked first by a ranking model). For each ground truth reply, we generate a set of negative samples, as done previously by researches performing the

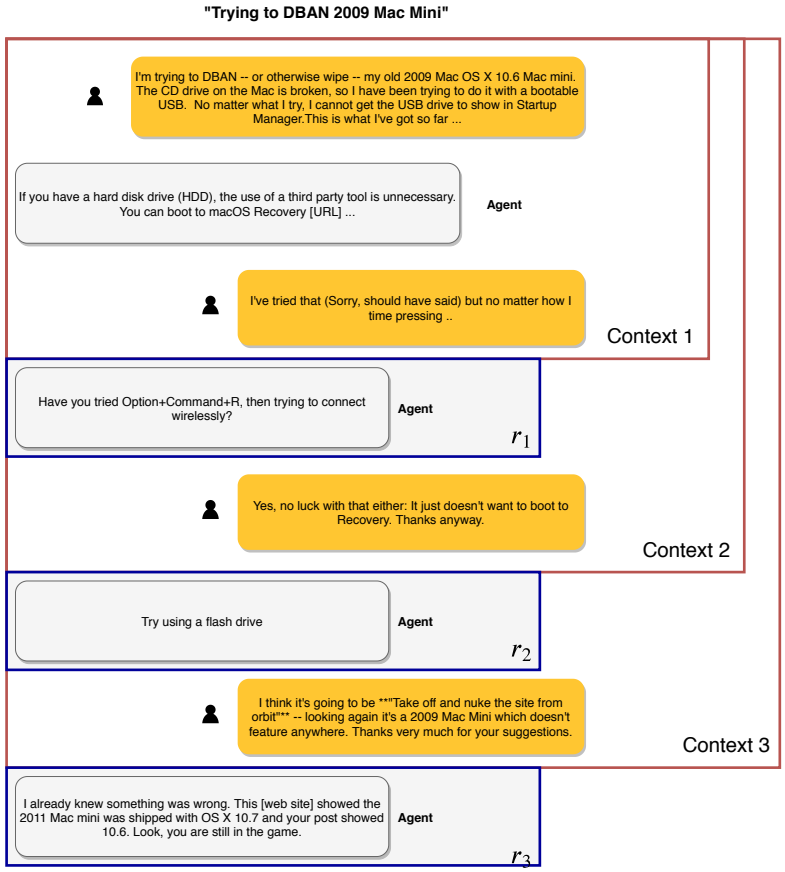


Figure 3.9: Example of a conversation split by contexts and responses. The red squares show the conversation contexts, while the blue squares with gray background show the potential responses.

same task [147]. To obtain these samples, we have used BM25 [107] on a corpus consisting of all agent replies from the training and validation splits defined in Chapter 3 to obtain the retrieval score of all agent replies in relationship to a context. We then randomly sample 10 (for `CRR10`) or 50 (for `CRR50`) negative samples from the top 1000 replies by retrieval score. For `CRR10`, a BM25 pool is created for each domain and, for a given context, all the negative samples are originating from the same domain. For `CRR50`, given that some domains have as few as 1000 conversations and we are sampling 50 negative responses per context, we have decided to create a single common pool for all the domains, as opposed to sampling from the same domain as the context belongs in, in order to ensure a high level of diversity. Furthermore, choosing this approach gives us the opportunity to study how the baselines perform when having to distinguish between documents coming from other sites.

For the GDR task, the procedure is almost identical, except for the fact that the agent replies are now replaced by the grounding documents that were mentioned in the reply. Given that an agent reply can contain multiple grounding documents, this means that, in this case, there can be multiple ground truth documents for the same conversation context.

For evaluation purposes on the CRR and GDR tasks, we have split the dataset into three subsets: *training*, *validation* and *test*. As suggested by the name, the training set is used for training the model, the validation for evaluating the fit of the model during the time the model is tuning its parameters and the test set for the final evaluation of the model. Following the same methodology as Yang et al. [148], we have split the dataset chronologically, with *training* containing the oldest conversations and *test* the newest. Training contains 70% of the conversations, while validation and test each contain 15%.

Since the user intent prediction is a simple classification task, the setup of this task is different than for the others. The input that is being passed to the models consists of rows containing only the utterance labels and the actual utterance. The output is a set of intents for an unseen example. Since the number of intent labels is rather small, we have opted for a 10-fold cross-validation evaluation, as opposed to the 3-way dataset split we have used for the previous tasks.

Table 3.2 shows a summary of all the tasks that we have derived from the dataset, including the inputs and outputs and the evaluation procedure for each task.

Task	**Input**	**Output**	**Evaluation**
CRR	context potential answer	1 - if the entry contains the true answer 0 - otherwise	train/validation/test dataset split
GDR	context potential grounding document	1 - if the entry contains the true document 0 - otherwise	train/validation/test dataset split
Intent Prediction	utterance	set of intent labels	10-fold cross-validation

Table 3.2: Summary of the tasks that we have defined, along with a description of the inputs and outputs and the evaluation methodology.

In conclusion, starting from a theoretical framework [104, 8], we were able to extract a set of requirements which was then transposed into a stringent list of criteria. Using these criteria and a dump of conversations from 14 StackExchange sites, we were then able to extract over 80,000 conversations between humans. Furthermore, based on the links that were mentioned in the agent responses, we were able to extract the referred documents. This enabled us to provide grounding for most of the conversations in the dataset. In order to include intents to the conversations, we have manually annotated 1,356 conversation with 9 types of intents, reaching a total of 6,701 intent labels. Finally, given the available data, we were able to derive 3 tasks: conversation response ranking, grounding document ranking and user intent prediction.

Chapter 4

Experiments

In this research, our focus was solely on ranking and classification methods and we decided to leave generative methods for further research. In this chapter, we will first detail the models that were used for solving the tasks, followed by a discussion of the results. In the final part, we will present a method that takes into account the labelled data from these 3 tasks in a multi-task learning setup.

4.1 Models

Our dataset is compatible with three tasks with different characteristics that require different approaches. Given this observation, we have applied several methods involving different types of models to discover which one obtains the highest performance. Furthermore, one of the models is also extended in order to be used in a setup where the tasks are learned jointly. This section describes all of the models that were used to perform the tasks both independently and in a multi-task learning setup. An overview of all the models and their compatibility with each task is presented in Table 4.1.

Model	CRR	GDR	Intent Prediction	MTL
AdaBoost			✓	
BERT	✓	✓	✓	
BERT MultiLabel			✓	
BiGRU			✓	
BM25	✓	✓		
DMN	✓	✓		✓
GradientBoost			✓	
Logistic Regression			✓	
SVM			✓	

Table 4.1: Models that were used throughout all of our experiments. The ✓signifies that the model on the corresponding row was used to solve the task on the corresponding column.

4.1.1 Base tasks

Response Ranking & Grounding Document Ranking

BM25

We have chosen 3 models for the ranking tasks based on three different rationale. The first baseline is *BM25* [107], which was detailed in Chapter 2. It is a baseline that works out of the box without extensive parameter tuning and was chosen due to it being a popular standard model for IR related tasks [130, 92, 21], including CRR [148].

In our experiments, we have used the *gensim*[1] implementation of BM25. To find the optimal values for k_1 and b, we have used grid search. Table 4.2 shows the optimal values for each parameter with respect to the task that was performed.

Task	k_1	b
CRR10	2.1	0.2
CRR50	2.1	0.5
GDR10	2.1	0.5
GDR50	2.1	0.5

Table 4.2: Optimal values for the k_1 and b parameter of BM25 with respect to each task.

DMN

The next model is a Deep Matching Network (*DMN*) [145], a neural model focused on interactions and specialized on conversational tasks, which has previously shown good performance when applied on a conversational response ranking task using the MSDialog dataset [148], obtaining better results than BM25. Since BM25 only performs lexical matching, we expect DMN, which performs higher-level pattern matching, to outperform it for our dataset as well.

DMN, as its name suggests, is a neural network that relies on matching between parts of text by creating interaction matrices between every past utterance in a conversation and the current potential answer. The input of the network is, therefore, represented by the current response (also known as potential answer) r_i^t on one side and the dialogue context on the other. In the next step, two interaction matrices are built for each pair of utterance u_i and response r_i: matrix M_1 is a *pairwise similarity matrix* and it stores the interactions between the word embeddings of u_i and r_i. M_2 stores the *hidden representation similarity matrix* using a bidirectional gated recurrent unit (BiGRU) that encodes r_i and u_i into hidden representations by modelling the neighbor context information around words from two directions. The BiGRU is neural network we have also used for the intent prediction task and will be detailed in the corresponding section

The two matrices are then passed to a convolutional neural network (CNN) in order to learn high level matching patterns by alternating between convolution and max-pooling operations. The newly obtained representations are passed to another BiGRU

[1] https://radimrehurek.com/gensim/summarization/bm25.html

to learn the dependencies and temporal relationships of utterances in the conversation. Finally, all combinations of u_i and r_i that were processed through this pipeline are gathered into a fully connected layer that outputs a matching score between the context and the response. The graphical representation of the entire process can be observed in Figure 4.1.

We have used the same base implementation of DMN[2] as Yang et al. [148]. In our experiments, we used word embeddings generated using word2vec[3] pre-trained on the training set. More specifically, a sentence is turned into a sequence of tokens $S = \{s_1, s_2, ...s_n\}$ of size n. Each utterance was truncated to a sequence length of $n = 50$, which was the default value in the implementation. We did not notice any significant improvements when raising the sequence length to higher values, such as $n = 100$. Padding occurs when the sentence is shorter than n. In our implementation of DMN, the inputs are randomly sampled from the dataset at each iteration of the training phase until the evaluation performance of the validation set is no longer increasing (the maximum number we have encountered throughout our experiments was 1300 iterations). The learning rate is set to 0.0001. The model that is used for testing is using the weights that were saved at the iteration that showed the highest performance on the validation dataset. The training loss for both the CRR and GDR tasks is the categorical cross-entropy loss. The Adam optimizer [68] was chosen for tuning the hyper-parameters during training.

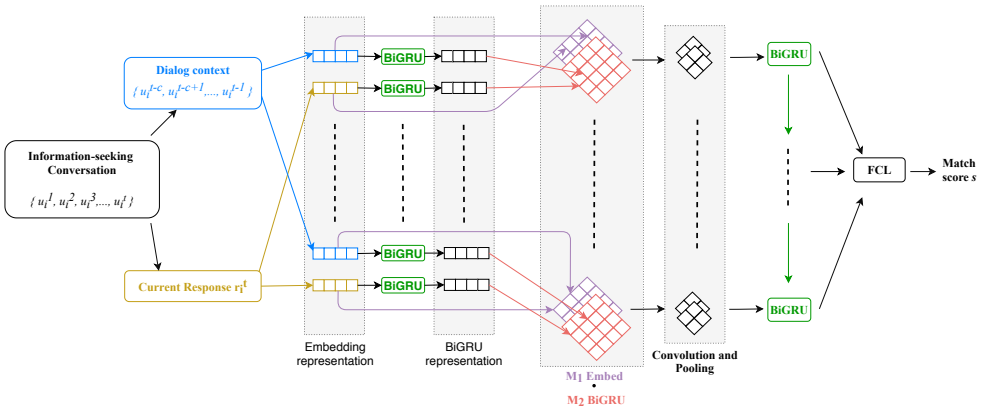


Figure 4.1: Functional diagram of DMN.

BERT

As the *strong* baseline, we have chosen fine-tuned BERT [38], which is the newest architecture of the three and is a language representation model that uses bidirectional representations on text data by conditioning both the left and right parts of a context at the same time. Currently, it achieves state-of-the-art performance on a variety of NLP tasks. For example, in the MS Marco leaderboard[4], which is a Machine Reading Comprehension (MRC) dataset, all of the models in the top 10 for the passage retrieval

[2] https://github.com/yangliuy/NeuralResponseRanking
[3] https://github.com/dav/word2vec.git
[4] http://www.msmarco.org/leaders.aspx

task are variations of BERT, while four of the models in the top 10 (including the first) for the Q&A task are also flavours of the same model.

BERT conditions both the left and right context on all of its layers, as opposed to the previous state-of-the-art models, such as OpenAI GPT [103], which uses a left to right architecture. Furthermore, its architecture allows the model to be used for a wide range of tasks with state-of-the-art performance [38]. One of the key assets of BERT is its ease-of-use: the model is available already pre-trained on a large corpus of documents containing 3.3B words [38] and can be extended for various NLP tasks by just adding one additional output layer for fine-tuning. Because of this characteristic, the fine-tuning is also fast: all of the experiments that were conducted by Devlin et al. [38] on pre-trained BERT can be replicated in a couple of GPU hours. The network accepts a list of tokens as input: the first token is always *[CLS]*, a special classification symbol, followed by a list of tokens representing the first sequence, followed by *[SEP]* - a token which denotes a separator and finally, a token representation of the next sequence. These tokens are then transformed into three types of embeddings: token, sentence and position, that are summed up to form the final input embeddings. These are then passed to a series of layers which are based on the Transformer architecture [136], which uses an attention mechanism that matches different parts of the same sequence in order to identify the relevant context. Each layer performs the attention computations on the word representations of the previous layer in order to obtain a new representation. The final hidden representation corresponding to the *[CLS]* token stores the aggregate sequence representation. The described process is detailed in Figure 4.2

The pre-training phase is being performed using 2 novel unsupervised prediction tasks. First, given that BERT uses bidirectional conditioning, this would allow the words to "see themselves" in a context with multiple layers. To avoid this, for each training batch, 15% of the input tokens are chosen randomly, out of which 80% is masked using a *[MASK]* token that replaces the actual word, 10% are replaced with a random word and 10% are left unchanged. Only the selected tokens are used for predicting the true word. Afterwards, the model solves a next sentence prediction task in order to understand the relationship between sentences. More specifically, the model receives 2 masked sentences from the previous task, with a probability of 50% that the second sentence is the one following the first. Adapting the pre-trained model to a variety of NLP tasks is done by just adding an additional output layer (usually a fully connected layer) on top of the existing architecture, which implies that only a handful of parameters need to be learned from scratch.

In our experiments, we have used a pre-trained model of BERT and fine-tuned the fully connected layer on top of the model output. The implementation of BERT[5] randomly samples inputs from the datasets until all the samples from the dataset have been exhausted. The learning rate is set to e^{-5}, while the context and response are truncated based on the following heuristic: given s_a and s_b two sequences (in our case, s_a is the context and s_b is the response), at each step, remove a token from the sequence that is currently the longest until the sum of their number of tokens fits within a given limit, which in our experiments is 200 tokens. The training loss for both CRR and GDR tasks is the mean squared error loss. As was the case for DMN, the Adam optimizer is

[5] https://github.com/huggingface/transformers

used for hyper-parameter optimization.

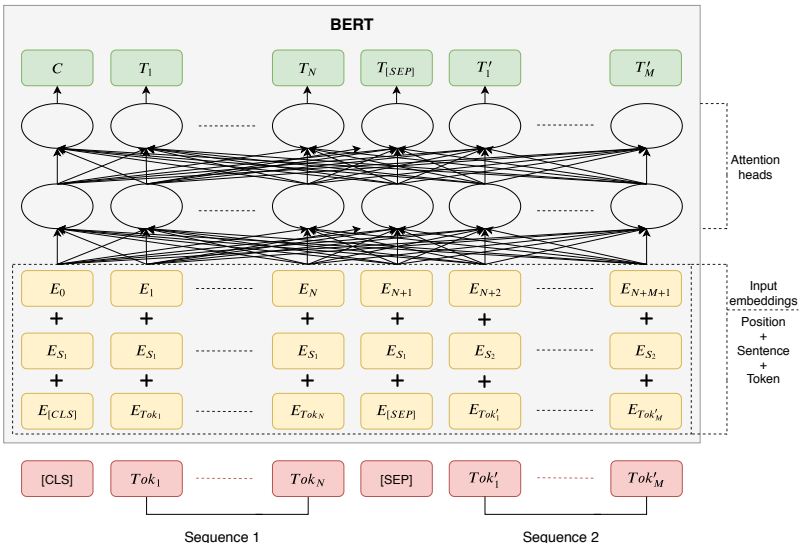


Figure 4.2: Functional diagram of BERT.The red squares depict the input of the network. The yellow squares depict the embedding representation of the words. The green squares represent the output sequence of the network, which is used to generate the final output.

User Intent Prediction

As was shown in Table 3.2, the user intent prediction task uses only one input, namely a user utterance, and outputs a set of intent labels. From the 3 aforementioned baselines, only BERT can also be used for this task, as its architecture can also accept [CLS] and only one sentence, without the need of specifying [SEP] and a second sentence. DMN requires two sequences as input, while BM25 is not suitable for classification.

By default, the implementation of BERT we have used does not support multi-label classification. Therefore, we also extend the model to include support for these types of tasks. More specifically, the fully connected layer at the end of the architecture has now nine outputs (the number of unique intent labels) instead of one and the mean squared error training loss function is replaced with the binary cross-entropy loss, given that the former does not support multi-label classification. In the standard BERT implementation, the set of intents for each entry is encoded as a single integer, which means that the model cannot assign different probabilities to different types of intents. For this reason, in the version of BERT that supports multi-label classification, we use a multi-label binarizer to represent every set of intents as a binary vector with a size equal to the number of distinct intents in the dataset. This will enable the model to output a probability for each possible type of intent.

Long Short-term Memory (LSTM) [61] is an RNN which has been used by many researchers for text classification [26, 78, 150]. Its popularity over other RNNs is due to the fact that it can preserve long-term dependencies better, which is crucial for long sequences of text. It accomplishes that using multiple gates, which have a role in asserting what information is maintained and what is discarded at a certain

point in time. A standard LSTM can only remember information from the past, while BiLSTMs combine information from both directions, which is essential when trying to understand a sequence of text. More recently, Cho et al. [27] introduced Gated Recurrent Unit (GRU), an RNN which tries to solve the same long-term dependencies as LSTMs through a similar architecture. However, it accomplishes that using two gates instead of three, which makes the model more efficient. BiGRUs are GRUs that combines information from both directions, similar to BiLSTMs. Given the added efficiency and based on the research of Jozefowicz et al. [65], which have shown that BiGRUs perform similar to BiLSTM on a variety of tasks, we have used a BiGRU as the second neural model for our intent prediction task.

The rest of the models are machine learning algorithms, which have been used previously by other researches for text classification: *SVM* [127, 30], *AdaBoost* [97, 116, 56], *Gradient Boosting* [56, 43] and *Logistic Regression* [51, 77]. The One-vs-Rest classification strategy is applied for each algorithm, which reduces the main problem of multi-label classification to multiple binary classification problems. This strategy implies that a classifier is instantiated for each class, where the positive samples are the instances of the class and the negative are instances from all the other classes. Given an unseen example, the resulting combined model predicts the labels for which the corresponding classifier outputs a positive result. We use the *scikit*-learn [6] framework as it contains implementations for all of the learning algorithms. The hyperparameter tuning is performed using grid search. For the learning algorithms, we represent each utterance using a bag-of-words representation and then apply TF-IDF term weighting following related work on text classification [19].

4.1.2 Multi-task learning

As we have detailed in Section 2.1.5, multi-task learning is a process through which a model jointly learns multiple tasks. Given the tasks are chosen sensibly, this approach has shown that it can improve performance over learning the tasks independently.

BERT was released at the end of 2018 and it took the research community several months to prove its state-of-the-art performance on a variety of tasks. When we have first started this research, BERT was not as popular as it currently is. Given that DMN was released earlier and already showed good performance on the CRR task, it was the preferred model for our multi-task learning setup.

The intent prediction task can be performed by matching an utterance with a set of intents. However, that is not compatible with the input that DMN accepts, as this network is specialized in matching conversation contexts with responses. For that reason, we will predict only the intent of the last utterance of a given context. This implies that DMN no longer requires the use of interaction matrices for this specific task, as the potential response is no longer used due to the fact that the network should only leverage the context in order to predict an intent. Therefore, the embedding and BiGRU representation are concatenated and transmitted directly to a fully connected layer, which outputs a set of intents.

The grounding document ranking task is very similar to the CRR task, given that they both require matching between a context and a response or a document. For this

[6] http://scikit-learn.org/

reason, both tasks share most of the components of DMN, with the exception of the last layer, which contains one fully connected layer that outputs a response matching score for the CRR task and another to output the document matching score for the GDR task. The diagram in Figure 4.3 depicts the multi-task learning process, given the aforementioned considerations. Each task can be plugged in or out of the architecture, which gives us the opportunity to study them in various combinations.

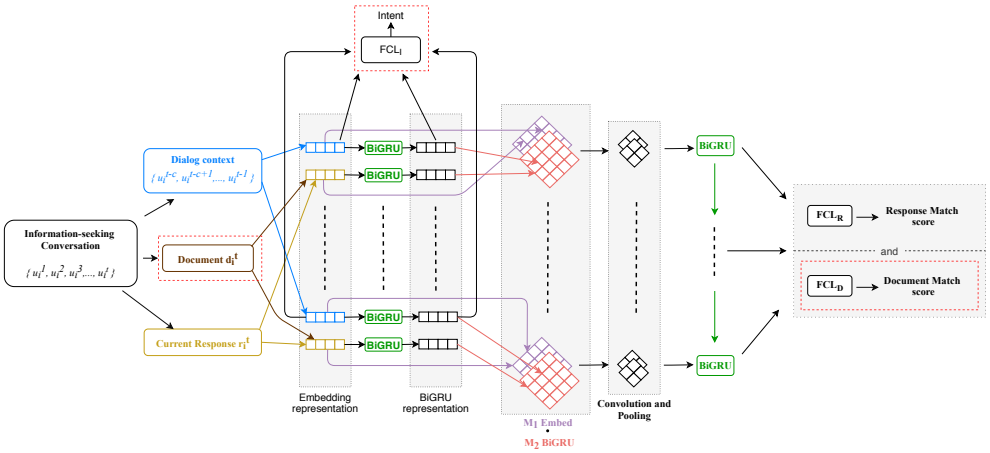


Figure 4.3: Functional diagram of the multi-task learning process. Components surrounded by red squares are the ones that were added to the base DMN architecture.

In our experiments, we assessed how training the models in a multi-task learning environment impacts the performance on the CRR and GDR tasks. The performance assessment on the intent prediction is omitted and only its influence on the other tasks is measured due to the fact that not all available utterances are used for predicting intents in this case. Given that we have two variants for both the CRR and GDR tasks, we perform all the experiments for both: `CRR10` + `GDR10` (which will be called `MTL10`) and `CRR50` + `GDR50` (called `MTL50`).

4.2 Results

In this section, we first assess the performance of each model for each task, accompanied by an in-depth analysis with respect to various conversational characteristics, such as the performance of each model by the size of the context or by domain. Afterwards, we report the results of jointly learning various combinations of the tasks for the multi-task version of DMN. Given that solving the user intent prediction task alongside any of the other tasks did not yield any significant improvements, we enrich the intent dataset with additional labels using a weakly supervised approach.

4.2.1 Base tasks

Conversation response ranking

To evaluate the performance of the 3 baselines on the CRR task, we have used 2 metrics that are traditionally used in IR for ranking: *mean average precision* (MAP) and

normalized discounted cumulative gain at 10 (NDCG@10). MAP is the "most standard measure among the TREC community" [117] and provides a measure of quality of retrieval across different recall levels in the form of a single figure. A simplified interpretation of the MAP value is the following: a value of 0.1 signifies that, on average, for a given query, only one in 10 documents is relevant. A value over 0.5 indicates that, on average, there are more relevant documents that irrelevant in the ranked list of results. NDCG measures the same properties as MAP. However, due to the logarithmic denominator in the equation of the metric, NDCG favors less the documents in the lower ranks.

The results in Table 4.3 show that BERT is the best performing model, with an absolute increase in MAP of 0.026 and a 0.019 increase in NDCG for `CRR10` over the second-best performing model, DMN. For `CRR50`, the performance difference increases to 0.077 for MAP and 0.062 for NDCG. As expected, BM25 has the worst performance, with differences in performance up to 0.46 for MAP as compared to DMN. An important aspect to note is that all of the models experience a severe decrease in performance when the number of negative samples is increased to 50. BERT is the model that is registering the lowest relative performance decrease, however, it still loses more than 22% of its performance when switching from `CRR10` to `CRR50`. These results show that under realistic conditions, where a retrieval system might have to choose among hundreds or thousands of possible responses, current approaches fail.

	`CRR10`		`CRR50`	
	MAP	nDCG@10	MAP	nDCG@10
BM25	0.318 (-)	0.476 (-)	0.163 (-)	0.195 (-)
DMN	0.756 (.0084)	0.817 (.0063)	0.513 (.018)	0.591 (.02)
BERT	**0.782** (.0012)*	**0.836** (.0009)*	**0.59** (.0019)*	**0.653** (.0014)*

Table 4.3: Baseline results on the test set for the conversational response ranking task. We run neural baselines 5 times and report the average and standard deviation (in brackets). * refers to significant to $p < 0.05$ compared to second highest scoring baseline using the *Student's t-test*

Figure 4.4 shows the performance of DMN and BERT over five runs based on the number of turns each context has. It can be observed that BERT's performance decreases as the context size increases, while DMN's remain relatively stable. This phenomenon can be explained based on how each network truncates utterances: DMN keeps the first 50 words of each utterance, regardless of the context size, while BERT maintains the first 200 words from the concatenation of the context and response, where the longest sequence is truncated more. In our specific case, given that the response should usually be much shorter than the context (given that the context contains at least 3 utterances), this means that, as the number of utterances increases, BERT will start to lose more and more context. Therefore, we expect that the performance of BERT will increase if the word limit is raised. Due to time constraints, we leave further parameter exploration with BERT for future work. However, most of the contexts in our dataset have only 3 turns, which explains why BERT's performance is already high.

Figure 4.5 displays the performance of the same neural models over 5 runs with

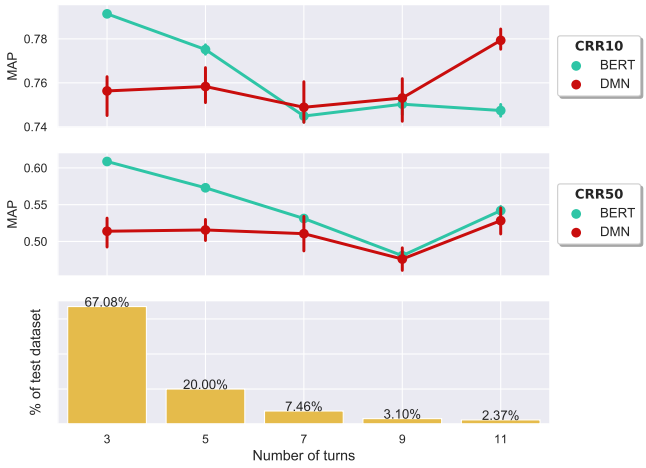


Figure 4.4: The bottom figure shows the distribution of conversations by the number of turns in the test dataset. The top two figures show the performance of DMN and BERT on the `CRR10` and `CRR50` tasks based on the number of turns per conversation context. MAP is averaged over five runs with different seeds. The vertical bars in the top 2 figures represent the confidence intervals.

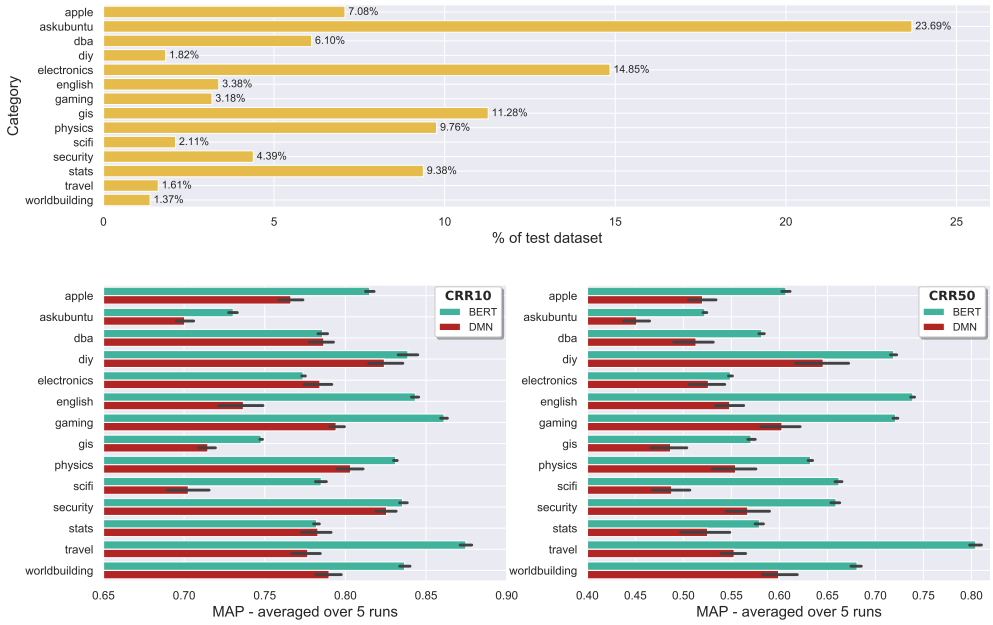


Figure 4.5: The top figure shows the distribution of conversations by category in the test dataset. The bottom figures depict the performance of DMN and BERT by category on the `CRR10` and `CRR50` tasks. MAP is averaged over 5 runs with different seeds.

regards to each of the Stack Exchange sites on both variations of the CRR task. For both DMN and BERT, conversations originating from *askubuntu* (which is the biggest domain in terms of conversations) have the worst performance. One area where BERT

truly shines is for domains that have small context sizes, as we have come to expect given the results in Figure 4.4: *travel*, *english*, *worldbuilding* and *gaming* are the domains for which BERT obtains the best performance which, according to Figure 3.3, are also domains with a low number of utterances per conversation.

The `CRR50` dataset can have, for the same context, potential answers stemming from various domains. As the conversations in Figure 3.5 shows, there are domains in `MANtIS` that have only a few concepts in common with the others. For example, conversations in *apple* use technical terms, such as firewall, while conversations in *travel* often refer to cities or travel documents. *Askubuntu*, on the other hand, presents more similarities with the former (for example, a firewall is also configurable in Ubuntu[7]).

Based on this observation, we hypothesize that a model receiving a context from a domain that has fewer concepts in common with the others, such as *travel*, should be able to identify the true answer with more ease than if the context originated from a technical domain, such as *apple*, which are more common in our dataset. This assumption is validated by the results for `CRR50` in Figure 4.5. For example, BERT's performance on *travel*, which is the only domain in the broader tourism category, is superior to any of the others, obtaining a precision of 0.804, an absolute increase of 0.06 as compared to the second-best scoring domain, *english*. The same observation is true for DMN's performance on *diy*, the only domain related to home improvement, which reached a precision of 0.64. On the other hand, more technical domains, such as *physics* or *electronics*, tend to have more concepts in common, and this fact is reflected in the low precision for both DMN and BERT. We have further observed this behaviour for DMN in the confusion matrix in Figure 4.6, which compares the domain of the top-rated document by the model for a context with the true answers. For example, contexts from *askubuntu* are often associated with potential answers from *apple*, *dba* or *gis*, as they all originate from the broader computer software domain. The same can be said for *physics* and *electronics*, which are both technical studies.

User Intent Prediction

As mentioned previously in Section 3.5.2, for user intent prediction, we evaluate all of the models using 10-fold cross-validation. As performance metrics, we use the following: precision (the number of correctly predicted labels divided by the predicted labels), micro and macro F1. We have chosen precision as it reflects the capacity of the model to detect the true positives and avoid false positives. The F1-score, apart from detecting true positives and false positives, is also useful for detecting false negatives, as it uses both precision and recall for computing the score. The difference between micro and macro F1 resides in the fact that the macro variant will compute the score independently for each class (thus, treating the classes equally), while micro-average takes into account the contribution of each class to the dataset. All of the metrics we are using have been used previously by other researchers for the intent prediction task [101]. A higher value for any of the metrics indicates higher performance in solving the task.

The results in Table 4.4 show that BERT is the best performing model, with a 0.133 absolute improvement in terms of precision over the next best performing algorithm,

[7] `https://askubuntu.com/questions/106952/interactive-firewall`

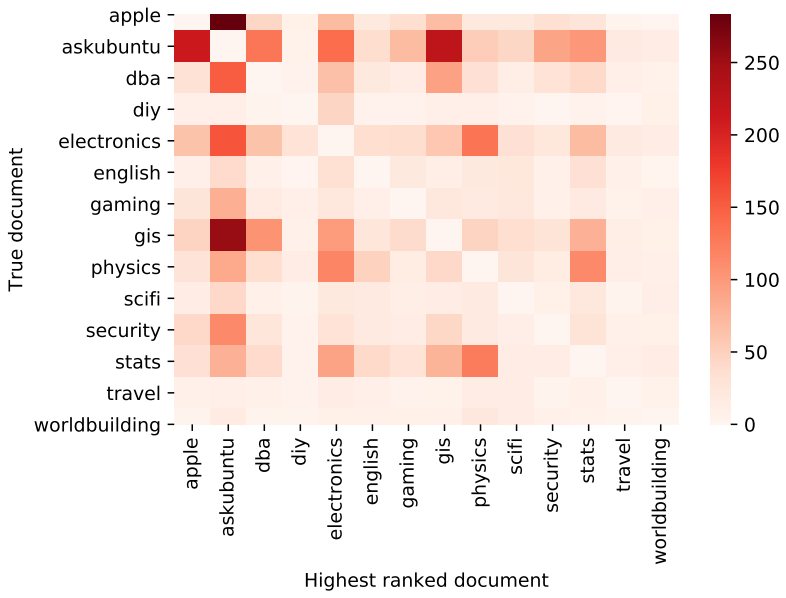


Figure 4.6: Confusion matrix at a domain level for the `CRR50` task, performed by DMN. Darker shades of red indicate a higher number of confusions between the two domains. Cases where the highest ranked document and the true document were in the same domain have been removed for clarity.

Classifier	Precision	F1-Micro	F1-Macro
LR	0.486 (.017)	0.469 (.014)	0.348 (.014)
SVM	0.532 (.021)	0.534 (.019)	0.455 (.018)
BiGRU	0.574 (.016)	0.563 (.015)	0.478 (.027)
AdaBoost	0.641 (.015)	0.585 (.012)	0.480 (.010)
GradBoost	0.657 (.017)	0.611 (.013)	0.491 (.011)
BERT Standard	**0.790** (.013)*	0.750 (.015)	0.591 (.030)
BERT MultiLabel	0.787 (.008)	**0.759** (.008)*	**0.617** (.025)*

Table 4.4: Results for the user intent prediction task, average and standard deviation of the cross-validation (k=10). * refers to significant to $p < 0.05$ compared to second highest scoring baseline (more specifically, GradBoost - the difference between the two BERT variants is not significant) using the *Student's t-test*.

Gradient Boosting. The multi-label version of BERT shows minimal improvements over the standard flavour in terms of F1-score. It can also be noticed that both boosting algorithms perform better than the BiGRU neural architecture, which is consistent with the work of Qu et al. [101], where their corresponding neural model (BiLSTM) performed worse than standard machine learning algorithms on the same task when no context information was available.

In order to further investigate the differences between the standard and multi-label variants of BERT, we have plotted the classification precision for each unique intent. Figure 4.7 shows that both BERT variants perform similarly for intents that have a high number of occurrences in the dataset (the top five intents by occurrence, as seen in Fig-

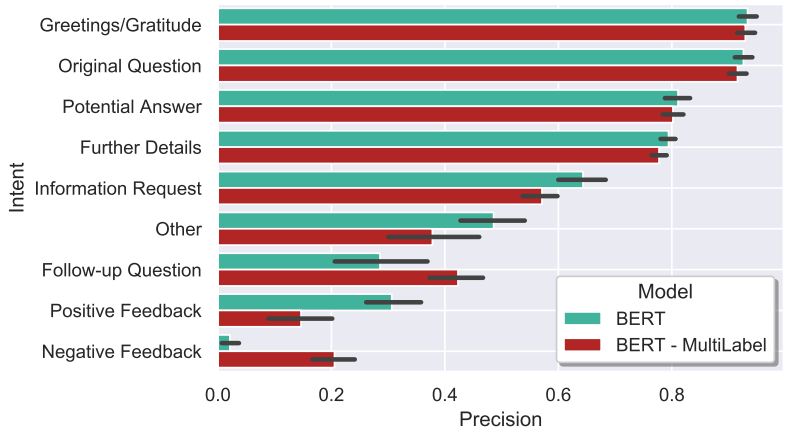


Figure 4.7: Performance of BERT (standard and multi-label) with respect to the precision of identifying each unique intent.

ure 3.6, are also the top 5 intents by classification performance). Although *Greetings / Gratitude* is not the most common intent in our dataset, the two models achieve the highest performance in predicting it correctly. This is due to the fact that this intent is communicated through a limited set of natural language expressions (such as *Thank you / Thanks* for gratitude or *Hello / Greetings* as a form of greet).

Feedback has proved to be challenging for both BERT variations, as they show subpar performance for identifying the *Negative Feedback* and *Positive Feedback* intents. The multi-label variant of BERT managed to correctly identify the negative feedback in 20.5% of the cases, while the standard variant in 2.1% of the cases. For the positive feedback, the standard variant correctly predicted 30.6% of the cases, while the multi-label BERT obtained 14.6%. We believe that one of the reasons for this weak performance on identifying feedback is the low representation of these intents (according to Figure 3.6, *Positive Feedback*, *Negative feedback* and *Other* are the least common intents in our dataset). The problem of dataset imbalance has been discussed by other researchers [63] and it has been shown that models generally favour the majority classes over the minority, which implies that minority classes will often be misclassified as one of the majority classes. This phenomenon is easily observable in the case of the negative feedback. In a conversation, negative feedback can be often accompanied by further details about the information need or by a form of gratitude to show appreciation for the agent's effort (*Thank you anyway*). This assumption is backed up by our findings regarding the most common errors that involve the *Negative Feedback* intent, which are presented in Table 4.5. It can be observed that the negative feedback is often mistaken with further details, which is an intent that is much more common in the dataset (further details labels represent 27% of the total number of labels, while negative feedback represents only 4%).

Grounding document ranking

When evaluating the three baselines for this task, we have used the same two metrics that were used for the conversational response ranking task, namely MAP and

True Label(s)	Predicted Label(s)	Model	% of total errors involving NF
FD, NF	FD	BERT	24.92%
		BERT - MultiLabel	22.81%
NF	FD	BERT	18.31%
		BERT - MultiLabel	12.08%
GG,NF	FD, GG	BERT	7.8 %
		BERT - MultiLabel	7.38 %
FD,GG,NF	FD, GG	BERT	7.8%
		BERT - MultiLabel	4.36 %

Table 4.5: Most common four errors involving the Negative Feedback intent.

nDCG@10.

	GDR10		GDR50	
	MAP	nDCG@10	MAP	nDCG@10
BM25	0.228 (-)	0.403 (-)	0.075 (-)	0.062 (-)
DMN	0.604 (.012)	0.677(.035)	0.431 (.011)	0.496 (.011)
BERT	**0.695** (.0021)*	**0.767**(.0018)*	**0.475** (.008)*	**0.53** (.0085)*

Table 4.6: Baseline results on the test set for the GDR task. We run the neural baselines 5 times and report the average and standard deviation (in brackets). * refers to significant to $p < 0.05$ compared to second highest scoring baseline using the *Student's t-test*

.

The results in Table 4.6 show that BERT outperforms DMN on this task as well (with an absolute improvement of 0.09 on the GDR10 task and roughly 0.04 on the GDR50 task for both metrics), with BM25 falling behind with a large margin (0.376 decrease in MAP and 0.274 decrease in NDCG for the GDR10 task as compared to DMN and a 0.356 decrease in MAP and 0.434 in NDCG).

Performing the same analysis on the performance of DMN and BERT by context size in Figure 4.8 yields a completely different result as compared to the CRR task. In this case, documents can be much larger than the context, which means that BERT will truncate the documents more than the context. This translates into the fact that the model can leverage the information provided by the context better. DMN is also seeing a sharper increase in performance as the context size increases (from 0.6 MAP for a context size of three to 0.75 for a context size of nine on GDR10, while for GDR50 the increase is from 0.41 to 0.67). This might be due to the fact that the GDR task is more difficult than CRR (documents are longer than responses) and the model needs more context to correctly approximate the ranking function.

From a domain perspective, the performance of DMN and BERT in Figure 4.9 reveals several insights. In this case, a larger context is advantageous for both DMN and BERT. For this reason, conversations from *askubuntu*, which has the second-highest number of utterances per conversation (according to Figure 3.3 - 5. 3 utterances per conversation), are now one of the most accurately ranked by both models, whereas for the CRR tasks they proved to be the most difficult. The same observation is applicable

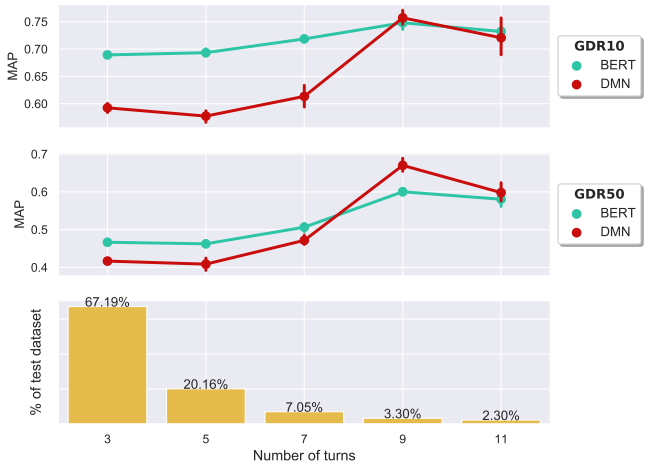


Figure 4.8: The bottom figure shows the distribution of conversations by the number of turns in the test dataset. The top two figures show the performance of DMN and BERT on the GDR10 and GDR50 tasks based on the number of turns per conversation context. MAP is averaged over five runs with different seeds.

for *electronics*, which contains, on average, the largest conversations (5.35 utterances per conversation). Furthermore, both of these domains have a relatively low number of words per document (under 1000), as opposed to other domains with large contexts but also large documents, such as *physics* (over 2500 words per document, 5.2 utterances per conversation) or *stats* (1600 words per document, 5.15 utterances per conversation).

As we have previously seen, when the negative samples are retrieved from a pool that is common for all domains rather than from the same site (in this case, GDR50), domains that are more distinctive tend to have a higher performance. A relevant example is *diy*, which has seen a significant increase in ranking performance on the GDR50 task over GDR10 for both models, even though the number of negative samples increased five times.

What is also noticeable for the variations of GDR, as compared to CRR, is that performance varies more between domains. For DMN, the standard deviations for both CRR variants are 0.04 and 0.05, while for BERT the reported values are 0.04 and 0.08. For the GDR variants, the standard deviations increase for DMN to 0.07 and 0.06 and for BERT to 0.08 and 0.1.

4.2.2 Multi-task learning

The results in Table 4.7 show that the MTL variant which implies performing the GDR and CRR tasks jointly performs slightly better than the standard variant on the CRR tasks. This is especially true on the MTL10 task, as the difference between the two diminishes for the more difficult task and is no longer statistically significant. When integrating the intents, we observe a decrease in performance over the baseline for both tasks.

Analyzing the learning curves in Figure 4.10 for both CRR variants, we notice

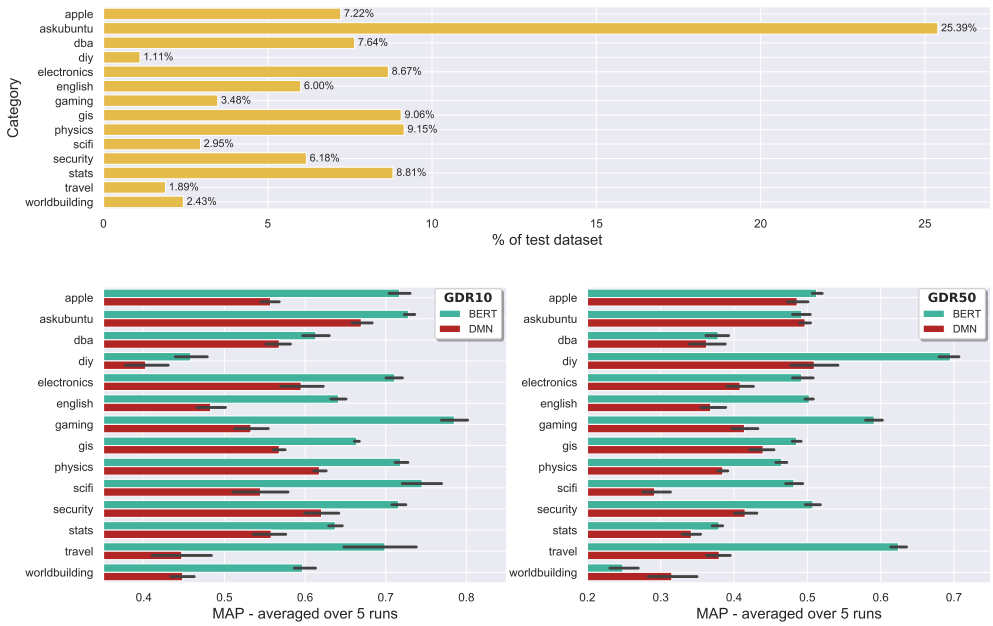


Figure 4.9: The top figure shows the distribution of conversations by category in the GDR test dataset. The bottom figures depict the performance of DMN and BERT by category on the `GDR10` and `GDR50` tasks. MAP is averaged over five runs with different seeds.

	`MTL10 - CRR`		`MTL50 - CRR`	
	MAP	nDCG@10	MAP	nDCG@10
CRR	0.756 (.0084)	0.817 (.0063)	0.513 (.023)	0.591 (.02)
CRR + Intents	0.735 (.024)	0.8 (.0185)	0.496 (.029)	0.575 (.027)
CRR + GDR	**0.766 (.0071)***	**0.824 (.0053)***	**0.519 (.018)**	**0.596 (0.0158)**
CRR + GDR + Intents	0.75 (.016)	0.812 (.012)	0.518 (.016)	0.595 (.015)

Table 4.7: Performance of the model for the CRR task in a multi-task learning environment. Metrics averaged over five runs.* refers to significant to $p < 0.05$ compared to the base task using *Student's t-test*.

that, initially, the GDR + CRR variant converge significantly faster to a higher MAP. However, after a certain number of iterations, the performance difference between it and the baseline decreases. It is also noticeable that the addition of intents in the learning process increases the confidence interval.

The results in Table 4.8 on the performance of MTL variants on the GDR task show a similar pattern to the other. The GDR + CRR variant performs significantly better than the baseline on both `MTL10` and `MTL50` tasks with a much higher margin than we have seen in the previous case. However, once again, jointly learning using the intents did not yield any improvements.

The learning curve for the GDR tasks in Figure 4.11 show that the models converge much faster to the optimal solution, due to the lower variety in documents, with the

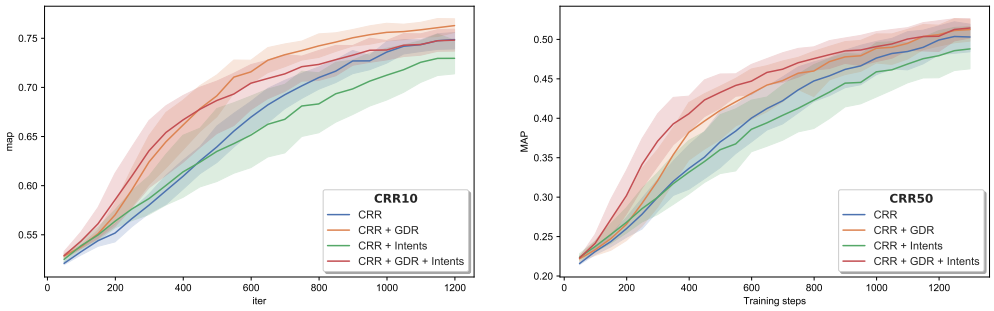


Figure 4.10: Learning curves for the standard and MTL variants of DMN on the `CRR10` and `CRR50` tasks. The colored lines represent mean performance on each evaluation step, average over five runs. The shades surrounding the lines represent the confidence interval.

	`MTL10 - GDR`		`MTL50 - GDR`	
	MAP	nDCG@10	MAP	nDCG@10
GDR	0.598 (.011)	0.691(.009)	0.431 (.011)	0.496 (.011)
GDR + Intents	0.587 (.011)	0.682(.008)	0.426 (.005)	0.492 (.005)
CRR + GDR	**0.672 (.009)***	**0.75 (.0071)***	**0.487 (.010)***	**0.557 (0.009)***
CRR + GDR + Intents	0.656 (.004)	0.737 (.004)	0.486 (.011)	0.556 (.011)

Table 4.8: Performance of the model for the GDR task in a multi-task learning environment. Metrics averaged over five runs. * refers to significant to $p < 0.05$ compared to the base task using *Student's t-test*.

CRR + GDR performing much better than the others. Given this fact, we hypothesize that the GDR task improves performance only up to a certain number of iterations in an MTL environment. Therefore, if we would use a scheduled learning environment in which one task is prioritised more over the other as the number of iterations increase [69], we would expect to see an increase in performance across the board.

Figure 4.11: Learning curves for the standard and MTL variants of DMN on the `GDR10` and `GDR50` tasks. The colored lines represent mean performance on each evaluation step, average over five runs. The shades surrounding the lines represent the confidence interval.

Given that jointly learning using user intents did not yield improvements over the standard learning procedure for any of the tasks, we hypothesize that one of the reasons for this behaviour is the lack of intent labels. For this reason, given that BERT obtained the best results on the intent prediction task, we have used it in a weakly-supervised environment (more specifically, using bootstrapping [149]) to obtain more intent labels based on the existing dataset of intents. To ensure that the task adds as little noise as possible, we have used a decision stump to set a confidence threshold for the prediction of each type of intent label such that at least 95% of the predictions on that specific label are accurate. The stump is applied on the subsets obtained previously through cross-validation. Since only the multi-label variant of BERT provides separate confidences for each type of intent, we have used this variant in our task. This approach enabled us to add an additional 140,000 labels to the existing intent pool. Using this newly obtained dataset (named Intents$^+$), we have run all the previous MTL experiments to see whether having more intent labels provides better results.

	MTL10 - CRR		MTL50 - CRR	
	MAP	nDCG@10	MAP	nDCG@10
CRR + Intents$^+$	**0.752 (.0177)**	**0.813(.013)**	**0.522 (.01)**	**0.6 (.008)***
CRR + GDR + Intents$^+$	**0.766 (.0072)***	**0.824 (.0054)***	0.515 (.015)	0.592 (.013)

Table 4.9: Performance of the model for the CRR tasks in a multi-task learning environment using the extended dataset of intents. Metrics averaged over five runs. Values in **bold** signify a mean performance increase over the MTL task that uses the initial set of intents. * refers to significant to $p < 0.05$ using *Student's t-test*.

The results in Table 4.9 show that, on the CRR task, adding more intent labels has a positive effect, especially in the case where all the 3 tasks are jointly learned. Given that the mean performance increased all across the board with a dataset obtained through weak supervision which only covers roughly 25% of conversations, we hypothesize that using human annotators to obtain an intent dataset that encompasses all conversations and does not add noise would increase the performance significantly.

	MTL10 - GDR		MTL50 - GDR	
	MAP	nDCG@10	MAP	nDCG@10
GDR + Intents$^+$	0.581 (.0105)	0.676(.0083)	0.43 (.0074)	0.494 (.0084)
CRR + GDR + Intents$^+$	0.663 (.013)	0.743 (.01)	0.488 (.006)	0.561 (.008)

Table 4.10: Performance of the model for the GDR tasks in a multi-task learning environment using the extended dataset of intents. Metrics averaged over five runs.

In the case of the GDR task, as can be observed in Table 4.10, there are no noticeable improvements in the ranking performance by adding more conversations with intent labels.

To conclude, starting from the three tasks that we have defined earlier, we have evaluated a multitude of models of different complexity. One model, namely BERT,

has shown the highest performance on all of the tasks, which is consistent with our expectations and the model's state-of-the-art performance on a variety of related tasks. Further analysis of the models' performance by context size has shown that, for the CRR task, BERT is negatively impacted by conversations with larger contexts. For the GDR task, the opposite is true, as a larger context size is beneficial for both DMN and BERT. Analysis by domains on the 50 negative sample variations of the CRR and GDR tasks show that contexts originating from sites that have fewer concepts in common with the others are easier to match with their corresponding potential answers.

Training DMN in a multi-task learning setup showed significant improvements when jointly learning the CRR and GDR tasks. Using the base intent prediction task alongside the others did not show any performance gains. However, by enriching the dataset with more labels through a weakly supervised setup, minor improvements were registered, which indicates that, if human annotators would be employed to obtain more intent labels, a higher degree of improvement in performance could be observed.

Chapter 5

Conclusions

Over the past 5 years, conversational search has grabbed the attention of the research community proven by the appearance of dedicated workshops[1,2] and researches [104, 8] appeared that tackle challenges associated with this topic. The popularity and widespread deployment of task-oriented conversational agents, such as Google Assistant and Amazon Alexa, has proved that there is a need for advanced conversational systems that can change the way we interact with retrieval systems. However, as research as shown [81], these commercial systems are far from ideal, as they are not able to engage in complex, multi-turn conversations.

Furthermore, although significant advances were made in various adjacent domains, such as DS, NLP and IR, as we have shown in Section 2.5, the research community has yet to release a conversational search dataset that fulfills all the necessary requirements to build a CSS. This fact was the main motivator of performing our research.

Using our proposed methodology, we have built a flexible framework capable of constructing a large-scale dataset with over 80,000 conversations that encompasses all the requirements of a conversational search dataset. To ensure the properties are respected, we have established strict acceptance conditions in the data collection pipeline. This fact reflects in the percentage of conversations that were retrieved from the StackExchange websites, as we retrieved only 4.77% of all possible conversations. The development effort to add up to 160 more domains from StackExchange is minimal, given that enough computational resources are provided. Due to time and resource limitations, we have added only 14 domains.

We have also augmented the conversations with 6,701 utterance labels by means of human annotation and 116,061 grounding documents associated to URLs mentioned by the users by using a web crawler, which is also provided within the framework. While a researcher interested to use this framework is free to use any other approach to build datasets for training a model, we provide a built-in functionality to split each conversation into contexts and generate 10 or 50 negative samples for each.

Using these training datasets, we have been able to devise 3 tasks: conversational response ranking (predict the next most likely *agent response*, given a context), user intent prediction and grounding document ranking (predict the next most likely *docu-*

[1] `https://scai.info/`

[2] `https://sites.google.com/view/cair-ws/home`

ment, given a context). For the first and second tasks, we have used DMN, a baseline that is specialized in conversation-related tasks and BERT, a state-of-the-art model that can be used for various NLP tasks. For both tasks and both flavours of the training datasets, BERT has shown the best performance. The performance difference we have noticed between the 10 and 50 negative samples is significant. This implies that, for a real-life scenario where there can be hundreds or thousands of possible answers, the current solutions might prove insufficient. For the intent prediction task, we have compared 2 variants of BERT (standard and multi-label classification compatible) with several learning algorithms and a recurrent neural network. Once again, BERT has shown the best performance (with no significant difference between the 2 variants). The fact that BERT has outperformed any other model we have tested is not surprising, given that the same model is considered state-of-the-art for many other tasks.

For the multi-task learning tasks, we have experimented only with DMN. The results show that combining the CRR and GDR tasks yields better performance than any of the 2 tasks combined, reaching a performance comparable to that of BERT on the GDR task. This is in line with other researches that used conversational datasets for multi-task learning setups and noticed significant improvements [121] Having a dataset that encompasses all the requirements for training conversational search models paves the way for multiple research directions, which will be discussed in the following section.

5.1 Future work

Detection of grounded conversations Currently, our data processing pipeline detects document grounding by simply checking if the utterance contains a link. However, as it was reflected in our analysis of the web pages for which the web crawler was not able to extract documents, there were many links that pointed towards images or web pages that no longer exist. Given these insights, a solution might be to integrate the crawler directly into the data collection pipeline for the conversations. This can enable us to automatically filter out conversations for which there is no grounding.

Intent labelling Currently, only a fraction of conversations have intent labels (1,356 out of 80,326). Although in our initial testing, using crowd-workers has yielded unsatisfactory results, we have not investigated further how we could improve the quality of their annotations. Other researches that constructed conversational datasets have successfully used crowd-workers by turning different data collection processed into games. Reddy et al. [106] have transformed the answer collection step into a game of predicting answers: workers would first try to guess the answer to the question and then the original was shown to them. This process increased the human F1-score by 5.1%. In our case, verifying utterance labels could be done through a similar game of predicting intents.

Negative sampling technique While our current strategy of employing BM25 for negative sampling gave us the opportunity to study how various models perform when trained using our dataset, the fact that BM25 relies only word matching to compute the similarity between two pieces of text introduces a bias in how we sample potential

responses as it restricts the possible space of samples to the ones that have a high tf-idf score. Fan et al. [45] propose a "learning to teach" paradigm, in which the negative documents and the order in which they are presented is decided by using a policy gradient model, based on previous states of the neural model under training. This shifts the paradigm from a static sampling approach to an active one, which can modify the heuristic at every step in the learning process. Applying the defined policy on an image classification task showed an improvement in both performance and training time, as the model was able to converge much faster. More importantly, the theoretical framework on which the research is based can be transposed to any type of task.

Curriculum and scheduled multi-task learning In our experiments, the multi-task learning model would receive batches with equal distribution between tasks. However, there are other approaches to learning the model several tasks. One of them is curriculum learning [12], which involves sequentially learning a task by starting with less complex examples and progressing towards more difficult ones. Applied on multi-task learning, curriculum learning has shown to improve the performance [100] over jointly learning on random samples, given that the order of the tasks are chosen appropriately. The other approach is called scheduled multi-task learning [69] and involves jointly learning tasks, but as the learning progresses, one task becomes "more important" than the other (e.g. uses more batches). In their research Kiperwasser and Ballesteros [69] propose a framework that jointly learns syntax and translation on a German to English corpus, while gradually putting more emphasis on translation. Their results show that if the probability to offer more batches from the translation task increases exponentially during training, the overall performance of the system increases over the standard, uniform distribution of batches. Given that our GDR task tends to reach maximum performance significantly faster than the CRR task, we hypothesize that if our network might improve its performance if it gradually receives more CRR-related batches as the learning process advances.

Combining retrieval and generative-based methods In our experiments, we have only used retrieval-based methods to solve our ranking tasks, as generative methods generally show a lower performance. However, no public dataset is large enough to be able to provide appropriate responses for any type of query, especially given that new queries can appear at any moment. Based on this intuition, Song et al. [122] looked into whether combining the superior performance of retrieval methods with the potentially infinite response space of generative methods can yield better results on a dialogue system. Their findings suggest that a combination of the two improve performance over each type of method taken independently.

Multi-task learning BERT Currently, BERT is considered the state-of-the-art neural model for NLP tasks, as it has outperformed any other architecture in a variety of tasks [38]. Therefore, given our experience with DMN, we expect that the performance of BERT will increase further if the network is trained in a multi-task learning environment. Liu et al. [80] have already constructed a multi-task learning framework over the same implementation of BERT we are using and they have obtained a 2.2% absolute improvement over standard BERT for multiple Natural Language Understanding tasks.

Addition of other domains to the dataset As stated previously, StackExchange contains over 174 domains. Due to time and resource limitations, we have used only 14 domains. However, our framework easily allows the addition of new domains by just specifying the download link for the raw data archive. This increase in the volume of data generates new challenges, which brings us to the next direction of future work.

Data collection framework improvements Throughout this research, the data collection procedure was performed using a single thread. However, given that conversations can be treated independently from one another, the task can be integrated into a multi-processing pipeline that can significantly decrease the execution time necessary for collecting the data [47]. This is especially important if one decides to add more domains, such as StackOverflow[3], which is the biggest StackExchange site and is roughly 60 times larger (in terms of the size of the raw data) than our biggest domain in the dataset, AskUbuntu[4]. The same observation applies to the pipeline that builds the model training datasets (`CRR10`, `CRR50`, `GDR10` and `GDR50`), given that the generation of negative responses depends solely on the current context of the conversation. For the larger training sets, the execution time can currently take up to a day.
The other improvement is related to our web crawler, which is currently is limited to extract HTML and text files. However, we have noticed that there are 8,225 PDF files which could be easily extracted by adding a *PDF converter module*[5] to our crawler.

[3] `https://stackoverflow.com/`
[4] `https://askubuntu.com/`
[5] `https://pypi.org/project/pdfminer.six/`

Bibliography

[1] Steven Abney, Michael Collins, and Amit Singhal. Answer extraction. In *Proceedings of the sixth conference on Applied natural language processing*, pages 296–301. Association for Computational Linguistics, 2000.

[2] Ali Mohamed Nabil Allam and Mohamed Hassan Haggag. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS)*, 2012.

[3] James Allen, George Ferguson, and Amanda Stent. An architecture for more realistic conversational systems. In *Proceedings of the 6th international conference on Intelligent user interfaces*, pages 1–8. ACM, 2001.

[4] JE Allen, Curry I Guinn, and E Horvtz. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications*, 14(5):14–23, 1999.

[5] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[6] Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(1): 29–81, 1995.

[7] Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. Frames: A corpus for adding memory to goal-oriented dialogue systems. In *SIGDIAL*, pages 207–219, 2017.

[8] Leif Azzopardi, Mateusz Dubiel, Martin Halvey, and Jeffery Dalton. Conceptualizing agent-human interactions during the conversational search process. In *The Second International Workshop on Conversational Approaches to Information Retrieval*, 2018.

[9] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[10] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4(May):83–99, 2003.

[11] Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines*, pages 567–580. Springer, 2003.

[12] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[13] Sumit Bhatia, Prakhar Biyani, and Prasenjit Mitra. Classifying user messages for managing web forum data. 2012.

[14] Abdelghani Bouziane, Djelloul Bouchiha, Noureddine Doumi, and Mimoun Malki. Question answering systems: survey and trends. *Procedia Computer Science*, 73:366–375, 2015.

[15] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 152–155. Association for Computational Linguistics, 1992.

[16] Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

[17] Paweł et al. Budzianowski. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*, pages 5016–5026, 2018.

[18] Christopher J Burges, Robert Ragno, and Quoc V Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200, 2007.

[19] Raphael Campos, Sérgio Canuto, Thiago Salles, Clebson CA de Sá, and Marcos André Gonçalves. Stacking bagged and boosted forests for effective automated classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 105–114. ACM, 2017.

[20] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 875–883. ACM, 2008.

[21] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193. ACM, 2006.

[22] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[23] Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Multi-task learning for boosting with application to web search ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1189–1198. ACM, 2010.

[24] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, 2017.

[25] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35, 2017.

[26] Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, 72:221–230, 2017.

[27] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[28] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. In *EMNLP*, pages 2174–2184, 2018.

[29] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 815–824. ACM, 2016.

[30] Fabrice Colas and Pavel Brazdil. Comparison of svm and some older classification algorithms in text classification tasks. In *IFIP International Conference on Artificial Intelligence in Theory and Practice*, pages 169–178. Springer, 2006.

[31] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[32] Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. Strategic dialogue management via deep reinforcement learning. *arXiv preprint arXiv:1511.08099*, 2015.

[33] J Shane Culpepper, Fernando Diaz, and Mark D Smucker. Research frontiers in information retrieval: Report from the third strategic workshop on information retrieval in lorne (swirl 2018). In *ACM SIGIR Forum*, volume 52, pages 34–90. ACM, 2018.

[34] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. Gate: an architecture for development of robust hlt applications. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 168–175. Association for Computational Linguistics, 2002.

[35] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Extended Semantic Web Conference*, pages 106–120. Springer, 2010.

[36] Li Deng and Dong Yu. Deep convex net: A scalable architecture for speech pattern classification. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[37] Li Deng, Gokhan Tur, Xiaodong He, and Dilek Hakkani-Tur. Use of kernel deep convex networks and end-to-end learning for spoken language understanding. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 210–215. IEEE, 2012.

[38] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[39] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *ICLR*, 2019.

[40] Sándor Dominich. *Mathematical foundations of information retrieval*, volume 12. Springer Science & Business Media, 2012.

[41] Hao-Wen Dong and Yi-Hsuan Yang. Towards a deeper understanding of adversarial losses. *arXiv preprint arXiv:1901.08753*, 2019.

[42] Ondřej Dušek and Filip Jurcicek. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 45–51, 2016.

[43] Claudia Ehrentraut, Markus Ekholm, Hideyuki Tanushi, Jörg Tiedemann, and Hercules Dalianis. Detecting hospital-acquired infections: a document classification approach using support vector machines and gradient tree boosting. *Health informatics journal*, 24(1):24–42, 2018.

[44] Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6 (Apr):615–637, 2005.

[45] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. *arXiv preprint arXiv:1805.03643*, 2018.

[46] Denis Fedorenko, Nikita Smetanin, and Artem Rodichev. Avoiding echo-responses in a retrieval-based conversation system. In *Conference on Artificial Intelligence and Natural Language*, pages 91–97. Springer, 2018.

[47] Alexandra Fedorova, Margo I Seltzer, Christopher A Small, and Daniel Nussbaum. Performance of multithreaded chip multiprocessors and implications for operating system design. 2005.

[48] DA Ferrucci. Introduction to this is watson. *IBM Journal of Research and Development*, 56(3):235–249, 2012.

[49] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[50] Jianfeng Gao, Michel Galley, Lihong Li, et al. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298, 2019.

[51] Alexander Genkin, David D Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

[52] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[53] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM, 1961.

[54] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64. ACM, 2016.

[55] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval. *arXiv preprint arXiv:1903.06902*, 2019.

[56] Amit Gupte, Sourabh Joshi, Pratik Gadgul, Akshay Kadam, and A Gupte. Comparative study of classification algorithms used in sentiment analysis. *International Journal of Computer Science and Information Technologies*, 5(5):6261–6264, 2014.

[57] Sanda M Harabagiu, Marius A Pasca, and Steven J Maiorano. Experiments with open-domain textual question answering. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*, 2000.

[58] Marti A Hearst. 'natural'search user interfaces. *Communications of the ACM*, 54(11):60–67, 2011.

[59] Djoerd Hiemstra. Information retrieval models. *Information Retrieval: searching in the 21st Century*, pages 1–17, 2009.

[60] Lynette Hirschman and Robert Gaizauskas. Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300, 2001.

[61] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[62] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[63] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[64] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[65] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350, 2015.

[66] Aditya Kalyanpur, Branimir K Boguraev, Siddharth Patwardhan, J William Murdock, Adam Lally, Chris Welty, John M Prager, Bonaventura Coppola, Achille Fokoue-Nkoutche, Lei Zhang, et al. Structured data and inference in deepqa. *IBM Journal of Research and Development*, 56(3.4):10–1, 2012.

[67] John F Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41, 1984.

[68] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[69] Eliyahu Kiperwasser and Miguel Ballesteros. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240, 2018.

[70] Julia Kiseleva, Kyle Williams, Jiepu Jiang, Ahmed Hassan Awadallah, Aidan C Crook, Imed Zitouni, and Tasos Anastasakos. Understanding user satisfaction with intelligent assistants. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, pages 121–130. ACM, 2016.

[71] Dan Klein and Christopher D Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, pages 3–10, 2003.

[72] Klaus Krippendorff. Computing krippendorff's alpha-reliability. 2011.

[73] Jonathan K Kummerfeld, Sai R Gouravajhala, Joseph J Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros C Polymenakos, and Walter Lasecki. A large-scale corpus for conversation disentanglement. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3846–3856, 2019.

[74] Adam Lally, Sugato Bagchi, Michael A Barborak, David W Buchanan, Jennifer Chu-Carroll, David A Ferrucci, Michael R Glass, Aditya Kalyanpur, Erik T Mueller, J William Murdock, et al. Watsonpaths: scenario-based question answering and inference over unstructured information. *AI Magazine*, 38(2):59–76, 2017.

[75] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436, 2015.

[76] Sungjin Lee and Maxine Eskenazi. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *Proceedings of the SIGDIAL 2013 Conference*, pages 414–422, 2013.

[77] Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, volume 3, pages 448–455, 2003.

[78] Xin Li, Lidong Bing, Wai Lam, and Bei Shi. Transformation networks for target-oriented sentiment classification. *arXiv preprint arXiv:1805.01086*, 2018.

[79] Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*, 2016.

[80] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

[81] Irene Lopatovska and Harriet Williams. Personification of the amazon alexa: Bff or a mindless companion. In *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval*, pages 265–268. ACM, 2018.

[82] Vanessa Lopez, Michele Pasin, and Enrico Motta. Aqualog: An ontology-portable question answering system for the semantic web. In *European Semantic Web Conference*, pages 546–562. Springer, 2005.

[83] Vanessa Lopez, Victoria Uren, Marta Sabou, and Enrico Motta. Is question answering fit for the semantic web?: a survey. *Semantic Web*, 2(2):125–155, 2011.

[84] Vanessa Lopez, Miriam Fernandez, Enrico Motta, and Nico Stieler. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic Web*, 3 (3):249–265, 2012.

[85] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, 2015.

[86] Zhengdong Lu and Hang Li. A deep architecture for matching short texts. In *Advances in neural information processing systems*, pages 1367–1375, 2013.

[87] Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

[88] Melvin Earl Maron and John Larry Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)*, 7(3):216–244, 1960.

[89] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[90] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[91] Bhaskar Mitra and Nick Craswell. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509*, 2017.

[92] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137*, 2016.

[93] Bhaskar Mitra, Nick Craswell, et al. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1):1–126, 2018.

[94] Robert J. Moore. Repeat repair & disengagement with voice-enabled conversational agents. In *CHI 18: Workshop on Voice-based Conversational UX Studies and Design*. ACM, 2018.

[95] Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, 2017.

[96] J William Murdock, Aditya Kalyanpur, Chris Welty, James Fan, David A Ferrucci, DC Gondek, Lei Zhang, and Hiroshi Kanayama. Typing candidate answers using type coercion. *IBM Journal of Research and Development*, 56(3.4): 7–1, 2012.

[97] Pio Nardiello, Fabrizio Sebastiani, and Alessandro Sperduti. Discretizing continuous attributes in adaboost for text categorization. In *European Conference on Information Retrieval*, pages 320–334. Springer, 2003.

[98] Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 647–656. ACM, 2018.

[99] Shiyan Ou, Viktor Pekar, Constantin Orasan, Christian Spurk, and Matteo Negri. Development and alignment of a domain-specific ontology for question answering. In *LREC*, 2008.

[100] Anastasia Pentina, Viktoriia Sharmanska, and Christoph H Lampert. Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5492–5500, 2015.

[101] Chen Qu, Liu Yang, W Bruce Croft, Johanne R Trippas, Yongfeng Zhang, and Minghui Qiu. Analyzing and characterizing user intent in information-seeking conversations. *arXiv preprint arXiv:1804.08759*, 2018.

[102] Chen Qu, Liu Yang, W Bruce Croft, Yongfeng Zhang, Johanne R Trippas, and Minghui Qiu. User intent prediction in information-seeking conversations. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, pages 25–33. ACM, 2019.

[103] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI, 2018.

[104] Filip Radlinski and Nick Craswell. A theoretical framework for conversational search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, pages 117–126. ACM, 2017.

[105] Jinfeng Rao, Hua He, and Jimmy Lin. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM, 2016.

[106] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*, 2018.

[107] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[108] Stephen E Robertson. The probability ranking principle in ir. *Journal of documentation*, 33(4):294–304, 1977.

[109] Stephen E Robertson. The methodology of information retrieval experiment. *Information retrieval experiment*, 1:9–31, 1981.

[110] Joseph John Rocchio. Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing*, pages 313–323, 1971.

[111] Bernardino Romera-Paredes, Andreas Argyriou, Nadia Berthouze, and Massimiliano Pontil. Exploiting unrelated tasks in multi-task learning. In *International conference on artificial intelligence and statistics*, pages 951–959, 2012.

[112] Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. Interpretation of natural language rules in conversational machine reading. In *EMNLP*, pages 2087–2097, 2018.

[113] G Salton. The smart retrieval system: Experiments in automatic document processing. 1971.

[114] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[115] Mark Sanderson et al. Test collection based evaluation of information retrieval systems. *Foundations and Trends® in Information Retrieval*, 4(4):247–375, 2010.

[116] Robert E Schapire, Yoram Singer, and Amit Singhal. Boosting and rocchio applied to text filtering. In *SIGIR*, volume 98, pages 215–223, 1998.

[117] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. Introduction to information retrieval. In *Proceedings of the international communication of association for computing machinery conference*, page 260, 2008.

[118] Alex Sciuto, Arnita Saini, Jodi Forlizzi, and Jason I Hong. Hey alexa, what's up?: A mixed-methods studies of in-home conversational agent usage. In *Proceedings of the 2018 Designing Interactive Systems Conference*, pages 857–868. ACM, 2018.

[119] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[120] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1577–1586, 2015.

[121] Tao Shen, Xiubo Geng, Tao Qin, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. Multi-task learning for conversational question answering over a large-scale knowledge base. *arXiv preprint arXiv:1910.05069*, 2019.

[122] Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*, 2016.

[123] K Spark-Jones. Report on the need for and provision of an'ideal'information retrieval test collection. *Computer Laboratory*, 1975.

[124] Amanda Stent, Rashmi Prasad, and Marilyn Walker. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd annual meeting on association for computational linguistics*, page 79. Association for Computational Linguistics, 2004.

[125] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[126] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[127] Aixin Sun, Ee-Peng Lim, and Ying Liu. On strategies for imbalanced text classification using svm: A comparative study. *Decision Support Systems*, 48(1): 191–201, 2009.

[128] Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. Learning to rank answers on large online qa collections. In *Proceedings of ACL-08: HLT*, pages 719–727, 2008.

[129] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[130] Krysta M Svore and Christopher JC Burges. A machine learning approach for improved bm25 retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1811–1814. ACM, 2009.

[131] Simone Teufel. An overview of evaluation methods in trec ad hoc information retrieval and trec question answering. In *Evaluation of text and speech systems*, pages 163–186. Springer, 2007.

[132] Paul Thomas, Daniel McDuff, Mary Czerwinski, and Nick Craswell. Misc: A data set of information-seeking conversations. In *SIGIR 1st International Workshop on Conversational Approaches to Information Retrieval (CAIR'17)*, volume 5, 2017.

[133] Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. How to make context more useful? an empirical study on context-aware neural conversational models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–236, 2017.

[134] Johanne R Trippas, Damiano Spina, Lawrence Cavedon, and Mark Sanderson. How do people interact in conversational speech-only search tasks: A preliminary analysis. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, pages 325–328. ACM, 2017.

[135] Johanne R Trippas, Damiano Spina, Lawrence Cavedon, Hideo Joho, and Mark Sanderson. Informing the design of spoken conversational search: perspective paper. In *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval*, pages 32–41. ACM, 2018.

[136] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010. Curran Associates Inc., 2017.

[137] Ellen M Voorhees. The philosophy of information retrieval evaluation. In *Workshop of the cross-language evaluation forum for european languages*, pages 355–370. Springer, 2001.

[138] Ellen M Voorhees et al. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82. Citeseer, 1999.

[139] Chong Wang, Miao Xiong, Qi Zhou, and Yong Yu. Panto: A portable natural language interface to ontologies. In *European Semantic Web Conference*, pages 473–487. Springer, 2007.

[140] Zhuoran Wang and Oliver Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432, 2013.

[141] Steven Wartik. Information retrieval. chapter Boolean Operations, pages 264–292. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992. ISBN 0-13-463837-9. URL `http://dl.acm.org/citation.cfm?id=129687.129699`.

[142] Zhongyu Wei, Qianlong Liu, Baolin Peng, Huaixiao Tou, Ting Chen, Xuanjing Huang, Kam-Fai Wong, and Xiangying Dai. Task-oriented dialogue system for automatic diagnosis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–207, 2018.

[143] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, 2013.

[144] W.A. Woods. Lunar rocks in natural English: Explorations in natural language question answering. In A. Zampolli, editor, *Linguistic Structures Processing*, pages 521–569. North-Holland, Amsterdam, 1977.

[145] Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505, 2017.

[146] Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. Building task-oriented dialogue systems for online shopping. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[147] L. Yang, M. Qiu, C. Qu, J. Guo, Y. Zhang, W. B. Croft, J. Huang, and H. Chen. Response ranking with deep matching networks and external knowledge in information-seeking conversation systems. In *SIGIR '18*, 2018.

[148] Liu Yang, Minghui Qiu, Chen Qu, Jiafeng Guo, Yongfeng Zhang, W Bruce Croft, Jun Huang, and Haiqing Chen. Response ranking with deep matching networks and external knowledge in information-seeking conversation systems.

In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 245–254. ACM, 2018.

[149] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[150] Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. Investigating capsule networks with dynamic routing for text classification. *arXiv preprint arXiv:1804.00538*, 2018.

Appendix A

Glossary

A.1 Hyper-parameter tuning

An overview of the tested parameters that were used for various experiments throughout our research in displayed in Table A.1.

Model	Parameter	Tested value
AdaBoost	$n_{estimators}$	[50, 100, 200, 250, 300, 500]
	$learning\_rate$	[0.1, 0..2, 0.3, 0.5, 0.7, 0.9]
	max_depth	[1, 3, 5, 7]
Gradient Boosting	$n_{estimators}$	[50, 100, 200, 250, 300, 500]
	$learning\_rate$	[0.1, 0..2, 0.3, 0.5, 0.7, 0.9]
	max_depth	[1, 3, 5, 7]
BM25	k_1	[0.5, 0.7, 0.9, 1.1, 1.3, 1.5, 1.7, 1.9, 2.1]
	b	[0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]
DMN	learning rate	[0.001, 0.0001, 0.00001]
	$n_{iterations}$	[500, 700, 1000, 1200, 1300]
BERT	learning rate	$[1e^{-5}, 2e^{-5}]$
	$max\_sequence\_length$	[128, 200]

Table A.1: Overview of parameters that were used throughout the experiments.

A.2 Vader Score threshold

One of the conditions for a conversation be accepted in our dataset was that, if the latest utterance originated from an information seeker, it should express a positive feedback. Otherwise, it is considered that the conversation has not ended. To be able to separate positive feedbacks from other types of intents, we have used the Vader score [62] to obtain the polarity of a utterance. More specifically, we have sampled 150 conversations from each domain and, without looking at the obtained score of the last utterance, we have labelled it as Positive Feedback / Not Positive Feedback. Then, using a decision stump, we have identified the optimal threshold for each domain that separates the positive feedback from the rest. All the thresholds can be observed in Table A.2.

Domain	Vader score threshold
apple	0.3
askubuntu	0.37
dba	0.35
diy	0.1
electronics	0.06
english	0.24
gaming	0.1
gis	0.35
physics	0.54
scifi	0.12
security	0.28
stats	0.48
travel	0.17
worldbuilding	0.58

Table A.2: Vader score threshold that was used to separate positive feedback from other types of intents. Values are calculated per domain

A.3 Intent annotation process

The annotation process was performed using a custom interface. In the top part of the interface, the task description and an explanation of all types of intents was supplied. The middle part contained a randomly sampled conversation, which the annotator would have to read to familiarize with the context. The bottom part contained the conversation broken down into utterances, each with a list of possible intents. For each utterance, the annotator would have to choose the appropriate intents from a list. After each utterance was labelled, submitting the action was necessary. The annotator is also able to receive another conversation if the difficulty proved to be too high. The interface is shown in Figure A.1.

Figure A.1: The interface that the annotators were required to use in order to label utterance intents.