Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Andrés Javier Estévez Fernández

Combining question answering models with transformer-based generative conversational agents

Master's thesis in Informatics
Supervisor: Jon Atle Gulla
Co-supervisor: Yujie Xing and Benjamin Kille
July 2022

NTNU
Norwegian University of
Science and Technology

Andrés Javier Estévez Fernández

Combining question answering models with transformer-based generative conversational agents

Master's thesis in Informatics
Supervisor: Jon Atle Gulla
Co-supervisor: Yujie Xing and Benjamin Kille
July 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Question answering (QA) is the task within the field of natural language processing (NLP) that intends the implementation of systems (i.e., artificial intelligence models) skilled to answer questions formulated by humans using a natural language. On the other hand, transformer-based generative conversational agents are trained to generate text on their own rather than drawing it out. While the former system aims to answer factoid questions (questions whose response may be found already written in a resource), the latter attempts to engage in dialogues that are more human-like.

It is a common issue for a user to encounter discrepancies when asking questions that may have different intentions and expect certain type of response from the conversational agent at hand. Since both type of systems provide a different conversational experience, this thesis proposes a way to combine a QA model such as BERT (that has been fine-tuned), with a transformer-based generative conversational agent as it is DialoGPT, by introducing a query classifier in that can forward the question to the model that is best suited to provide an answer.

The main results show that when combining the two models the overall F1 score gets increased by 4.93% in comparison with the best score achieved when only using the models separately to answer all types of questions. In order to associate the two approaches, three different query classifier were tested out: Naive Bayes, Support Vector Machine, and Random Forest. The Support Vector Machine algorithm showed to perform the best by attaining an F1 score of 20.79%, in contrast with 15.86% and 11.08% from the case in which only BERT and DialoGPT are employed, respectively.

Sammendrag

Spørsmålssvar (QA) er oppgaven innenfor feltet naturlig språkbehandling (NLP) som har til hensikt å implementere systemer (dvs. kunstig intelligensmodeller) som er dyktige til å svare på spørsmål formulert av mennesker ved hjelp av et naturlig språk. På den annen side er transformatorbaserte generative samtaleagenter opplært til å generere tekst på egen hånd i stedet for å tegne den ut. Mens det førstnevnte systemet tar sikte på å svare på faktaspørsmål (spørsmål hvis svar kan finnes allerede skrevet i en ressurs), prøver sistnevnte å delta i dialoger som er mer menneskelignende.

Det er et vanlig problem for en bruker å støte på avvik når de stiller spørsmål som kan ha forskjellige intensjoner og forvente en viss type respons fra samtaleagenten. Siden begge typer systemer gir en annen samtaleopplevelse, foreslår denne oppgaven en måte å kombinere en QA-modell som BERT (som har blitt finjustert), med en transformatorbasert generativ samtaleagent som den er DialoGPT, ved å introdusere en spørring klassifiserer i som kan videresende spørsmålet til den modellen som er best egnet til å gi et svar.

Hovedresultatene viser at når man kombinerer de to modellene, økes den samlede F1-poengsummen med 4,93% sammenlignet med den beste poengsummen oppnådd når man kun bruker modellene separat for å svare på alle typer spørsmål. For å assosiere de to tilnærmingene ble tre forskjellige spørringsklassifiserere testet ut: Naive Bayes, Support Vector Machine og Random Forest. Support Vector Machine-algoritmen viste seg å yte best ved å oppnå en F1-score på 20,79%, i motsetning til 15,86% og 11,08% fra tilfellet der kun henholdsvis BERT og DialoGPT er ansatt.

Preface

This thesis has been submitted to the Norwegian University of Science and Technology (NTNU), Department of Computer Science (IDI) as part of the course IT3920 - Master Thesis for MSIT, which belongs to the study program Master of Science in Informatics: Artificial Intelligence Specialization.

For their valuable guidance and feedback at every level of the project, my co-supervisors Yujie Xing, Benjamin Kille, and my supervisor Jon Atle Gulla are acknowledged. I also want to express my gratitude to my friends and family for their continuous support throughout the entirety of my academic career, especially those who were closer to me during my stayed in Trondheim.

Contents

Figures

Tables

Chapter 1

Introduction

This chapter provides a summary of the primary arguments that are presented in this thesis. The reasons for addressing the problem that is presented, as well as the objectives and research questions, are presented, and a brief discussion of the methodology utilized and the findings that were obtained follows. After the results section, there is an outline for the remaining chapters of the thesis.

1.1 Motivation

An open-domain conversational agent should succeed in language understanding, language creation, emotional resonance, memory, vocabulary knowledge, and conversational strategy, amongst other desired traits, in order to create a believable experience for users. In scientific work, the effective method of pushing growth along specific skills and constructing an assemblage of subsystems has been adopted. Each of these subsystems specializes at a solitary attribute while typically bypassing others. For example, enabling user activity in open-domain interactions is an incredibly difficult task since it demands comprehension of the many ways in which a user might show initiative and the capacity to reply to each of them in a manner that is distinctive. As a result presented by this difficulty, w hen it concerns to in-depth chats, many of the prior dialogue systems relied heavily on bot-initiative to guide users down carefully prepared pathways. Contrastingly , dialogues are likely to be more superficial in systems that aim to increase the user's level of initiative through the use of non-scripted alternatives. Therefore, there is a significant amount of potential for both development and innovation in the pursuit of concurrently achieving two or more traits that are complimentary to one another.

Widely known conversational systems such as Cortana, Alexa, and Siri have experienced rapid adoption as a mean of assisting users in accomplishing certain objectives or in satisfying the users' need for communication. The majority of conversational interfaces have as their primary goal the communication of information to human beings through the provision of replies that are question-relevant, natural, and instructive. Nevertheless, preceding dialog systems that

were assembled by end-to-end training of neural networks have a tendency to undergo the commonly named "safe response" issue [1]. This problem refers to the fact that these systems frequently produce non-informative replies, such as "I am not sure" and "I do not know what you mean"

In addition, human interactions are often more casual and chaotic, and when they take the form of written communication, they frequently involve informal acronyms as well as grammatical and lexical mistakes. The majority of generation systems that fall into the category of open-domain models have flaws with inconsistencies in their content or approach [2], an absence of long-term contextual knowledge, and flatness. Even though a model strategy for increasing information content can help ease these problems, when it comes to dealing with these challenges, a transformer-based topology that often employs multiple layers of self-attentive technique, which allows for cross-attention in a mathematically effective way, appears like it is a logical fit for a more comprehensive solution. Transformer architectures, for instance, make it possible for long-term dependence information to be properly retained over time [3], which improves content coherence.

1.2 Goals and research questions

The goal of this thesis is to explain what are the main state-of-the-art question answering (QA) models as well as the latest advances in regards to transformer-based generative conversational agents. Furthermore, efforts are focused on investigating what are the drawbacks that these generative agents may present, and finally, an approach to combine QA and transformer-based models is suggested with the purpose of enhancing the user experience and while providing meaningful answers to the later.

The research questions that this thesis pursue are the following:

RQ1 What is the state-of-the-art in QA models and generative models?

RQ2 What are the drawbacks of transformer-based generative conversational agents and how these can be addressed by combining such agents with QA models?

RQ3 How can knowledge be incorporated into transformer-based generative models?

1.3 Approach

The objective of this thesis is to evaluate how question answering (QA) models can be combined with transformer-based generative conversational agents. With this intention, the experiments include fine-tuning a model that maps an input

sequence that can vary in length to a fixed-length vector (also known as encoder) such as BERT, so that it is trained specifically for the task of answering questions. Testing a well-known open-domain question answering model such as DialoGPT is also part of the experiments, and finally, three different query classifiers are proposed in order to combine these two models (BERT and DialoGPT) and forward the inquiry to the appropriate system according to the type of question. The three approaches are Naive Bayes (NB), Support Vector Machine (SVM), and Random Forest (RF).

Several publicly available datasets were used to run the experiments and the results were compared to the case in which only one of the models was used to answer the questions (instead of being combined).

1.4 Results

Table 1.1 illustrates a summary of the main results that were obtained. Findings show that when providing the fine-tuned BERT model with factoid questions (i.e., queries that require extracting an answer from certain passage rather than producing it), an F1 score of 31.03% was achieved. On the other hand, when feeding generative conversational agent DialoGPT with open-domain questions (i.e., questions that demand text generation from the model) a score of 12.20% was attained. Lastly, when associating both models under the hybrid setup by introducing a query classifier in between (NB, SVM, and RF), a score of 20.79% was reached when employing the Support Vector Machine algorithm.

Results reflects that the overall performance of a conversational system may be enhanced by allocating a query classifier that can categorize the type of question and forward it to the model that is best suited to create a response.

Table 1.1: F1 score obtained from the fine-tuned BERT model and DialoGPT separately (tested on the type of questions that the model has being trained to respond), as well as when combined with the query classifiers NB, SVM, and RF, where both types of queries were fed into the system.

	Factoid questions	**Open-domain questions**	**Factoid + Open-domain questions**		
BERT	31.03%	—	15.86%		
DialoGPT	—	12.20%	11.08%		
Hybrid	—	—	NB	SVM	RF
			19.39%	20.79%	20.69%

1.5 Thesis outline

The following is the order in which the remaining portions of the thesis are presented: The information offered in Chapter 2 aims to provide the necessary theoretical

framework in order to better comprehend the work that is presented in subsequent chapters. In Chapter 3, we take a high-level look at some of the relevant work that has been done in the fields of question-answering models and transformer-based generative conversational agents. In Chapter 4, the data that was utilized for running the experiments and analyzing the models is introduced. In Chapter 5, a comprehensive discussion of the tests and their procedures is provided. In Chapter 6, the outcomes of the experiments are debated. Finally, t he thesis is concluded in Chapter 7, which also includes any pertinent future work.

Chapter 2

Background Theory

2.1 Question Answering

Question answering (QA) is a sub-field of natural language processing (NLP). A QA task studies the implementation of systems (i.e., artificial intelligence models) that are capable to answer questions proposed by human in natural language [4]. Besides answering questions, these type of tasks also require the system to understand what is being asked. The capacity of a program to convert utterances into an internal representation (i.e., the data structure that a computer uses as internal knowledge) so that it can provide legitimate responses to queries posed by a user is known as computer comprehension of natural language (or natural language understanding).

The lexical gap between the question at hand and the history of previous inquiries is probably the most significant challenge. It may be seen in the contrast between inquiries presented in natural language and the knowledge base content that is semantically organized [5]. This lexical gap may be noticed as the difference between user queries that ask for the same item but use different terms, as well as the difference between response and query, which might differ significantly from a lexical viewpoint.

Even though it may seem straight forward for humans to distinguish the difference between a question such as “What is the owner of Facebook?” and “What is 13 multiplied by 21?”, QA systems will most likely find the answer to the former on a collection of resources while struggle with additional processing to answer the latter one. There are two main types of QA approaches: extractive QA, which aims to find the answer that is already known and written down on a resource (e.g., documents); and abstractive QA, wherein the system needs to redact a response that has not been extracted textually from a passage (thus involving language generation).

There are several types of questions that a QA system can be asked to solve, varying from simple one-word-answer questions to queries that expect a list of responses or interactive answers [6]. Factoid questions represent the simplest form of query in which the user is asking for a fact and the answer can be expressed in

a few words (e.g., “Who is the president of the United States?”). How (and why) questions can also be considered within this group regardless needing a phrase or two (e.g., “How did Albert Einstein die?”).

Furthermore, list questions are the ones that require a set of factoid answers as a response (e.g., “What movies does Brad Pitt appear in?”). In this case, the answer may be assembled from a set of documents and the system will have a set of candidate answers when it reaches the latter stages of processing, and while just a single answer is necessary for factoid questions, the system must return as many as are believed to be right for this type of inquiries. The system must not only provide numerical scores to candidate answers for the purpose of ranking them, but it must also have certain confidence threshold in order to determine when to truncate the list as a result of this need.

Definition questions such as “What does a CPU do?” might need a set of properties or facts for a valid answer. Besides, relationship questions ask whether a relationship between two entities exists, the evidence that there is for one of them, or how one entity influences the other (e.g., “What was the influence of Socrates in the life of Plato?”). Finally, interactive questions include an optional interactive stage in which the system might ask for clarifications or guidance as to which query terms are more important. In this type of question, the system must keep record of previous query-answer tuples.

There are two major paradigms in QA: information-retrieval (IR) based QA, and knowledge-based QA [7].

2.1.1 Information Retrieval Techniques

A term refers to a single word in a collection, but it may also refer to phrases within the collection. The IR task consists of returning a group of documents from a collection as a result of a query posed by the user to a retrieval system (a subset of information access systems). In the literature, the task that considers the query term importance and thus performs relevance matching, is also known as ad-hoc retrieval [8]. Documents refer to any unit of text that the system indexes and retrieves (e.g., scientific papers, web pages, news articles, short paragraphs). A collection refers to a group of documents that are used to respond to user term queries. Finally, a query is a representation of a user’s information requirement expressed as a collection of terms.

As an example, a simple information retrieval architecture might map queries and documents to vectors based on unigram word counts, and then utilize the cosine similarity between the vectors to rank potentially relevant items. When evaluating the match between a document and a query, the term weight can be utilized to grade the match.

TF-IDF

TF-IDF is a statistical metric that assesses how important a word is to a document inside a set of documents. It is the product of two terms: the term frequency (tf) and the inverse document frequency (idf). The term frequency describes how frequently a word appears in a text; words that appear more frequently in a document d are more likely to include information about the document's contents. Rather than using the raw count of words, it is common to utilize the log10 of the word frequency. The idea is that a term occurring 100 times in a text does not increase the likelihood that the word is significant to the content of the document by a factor of a hundred. Because it is not possible to compute the log of zero, 1 is generally added to the count, as follows [7]:

$$tf_{t,d} = \log(count(t,d) + 1) \tag{2.1}$$

The number of documents in which a term t appears is represented by the document frequency df_t. Terms that only appear in a few documents are valuable for distinguishing those documents from the rest of the collection; words that appear throughout the whole collection are not as effective for this purpose. The inverse document frequency, often known as the idf term weight, is defined as follows [7]:

$$idf_t = \log\left(\frac{N}{df_t}\right) \tag{2.2}$$

where N is the total number of documents in the collection, and df_t is the number of documents in which term t occurs. The fewer documents in which a term occurs, the higher this weight; the lowest weight of 0 is assigned to terms that occur in every document. Finally, the $TD\text{-}IDF$ value for term t in document d is the product of the term frequency ($tf_{t,d}$) and the inverse document frequency (idf_t):

$$TF\text{-}IDF_{t,d} = tf_{t,d} \times idf_t \tag{2.3}$$

Document scoring

The objective is to get the documents ranked to keep only the most promising ones. This score is obtained from the cosine similarity between the document's vector d and the query vector q [7].

$$\text{score}(q,d) = \cos q, d = \frac{q \cdot d}{|q||d|} \tag{2.4}$$

By utilizing the $TD\text{-}IDF$F values and writing out the dot product as a sum of products, we may express the prior equation in the following way:

$$score(q,d) = \sum_{t \in q} \frac{TD\text{-}IDF(t,q)}{\sqrt{\sum_{q_i \in q} TD\text{-}IDF^2(q_i,q)}} \cdot \frac{TD\text{-}IDF(t,d)}{\sqrt{\sum_{d_i \in d} TD\text{-}IDF^2(d_i,d)}} \tag{2.5}$$

As a result, it is standard practice to approximate Equation 2.5 by making the query processing more straightforward. Because queries are often relatively brief, each query term is most likely to have a count of one. Furthermore, the cosine normalization for the query will be the same for all documents, therefore the ranking between any two documents will not be affected by this factor. As a result, we often apply the following simple score for a document d when presented with a question q:

$$score(q,d) = \frac{TD\text{-}IDF(t,d)}{|d|} = \frac{TD\text{-}IDF(t,d)}{\sqrt{\sum_{d_i \in d} TD\text{-}IDF^2(d_i,d)}} \tag{2.6}$$

The practice of removing high-frequency terms from both the query and document before representing them was widespread in previous years. A stop list is a list of high-frequency terms that should be eliminated from the vocabulary. The assumption is that high-frequency words (such as the, a, and to) have minimal semantic significance and will not aid in retrieval. The disadvantage of employing a stop list is that it makes it more difficult to search for sentences that contain terms from the list itself. Examples of frequent stop lists may include reducing phrases such as "the movie is not good" to the sentence "movie good", changing completely the sentiment of the text. Today's IR systems employ stop lists far less frequently, partially because of an increased efficiency, and partly because much of its functionality is already handled by IDF weighting, which devalues function words that appear in every document. Nonetheless, stop word removal is still be beneficial in a variety of NLP tasks [7].

Evaluating IR performance

Precision and Recall are often used as metrics to evaluate the results of an IR system. These metrics are mainly based on the following assumption [9]: every document within the collection and returned by the system can be catalogue as "relevant" or "not relevant" for the purpose of the query in dispute. Precision refers to the proportion of relevant documents among the retrieved ones, whereas recall is the fraction of relevant documents that has been retrieved.

$$\text{Precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \tag{2.7}$$

$$\text{Recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} \tag{2.8}$$

Several insights may be established with such definitions. Firstly, while precision can be seen as a measure of retrieval performance in excluding non-relevant documents from the retrieved set, recall, on the other hand, can be interpreted as a measure of effectiveness in including relevant documents in the retrieved set. Furthermore, even though 100% recall can always be achieved by examining all

the collection, this is something that is avoided as it goes against the principles of any information retrieval system. In fact, high recall may not always be needed since the system could have limitations in regards to the number of relevant documents that it requires for further processing.

These indicators themselves are insufficient for evaluating the performance of a system that rates the documents it provides to the user. For the purpose of evaluating the performance of two ranked IR systems, it is required a metric that favors the system that ranks relevant documents higher in the results list. Then, a modification is applied to precision and recall in order to describe how effectively a system performs when it comes to selecting relevant documents up the ranking. As we need to combine the precision and recall values from all queries to be able to compare two retrieval systems, an average precision (AP) plot is made at 11 fixed recall values (varying from 0 to 1.0 in steps of 0.1). To perform this operation interpolated precision values are used for each of the 11 recall levels; this is, taking the maximum precision value reached over all recall levels greater or equal than the one we are currently computing. While this interpolation approach allows us to average performance over several queries, it also aids in smoothing out erratic precision values in the original data. By awarding the highest accuracy value obtained at greater levels of recall than the one being assessed, it is intended to provide systems with a benefit of the doubt in their performance [7].

Another metric commonly used to compare system is mean average precision (mAP). In this approach we only consider the precision at those points where a relevant document has been found (from the set of returned documents). Hence, we take the average precision (AP) of these individual precision measurements throughout the whole result set for a single query as [7]:

$$AP = \frac{1}{GTP} \sum_{d \in R_r} Precision_r(d) \tag{2.9}$$

where GPT is the total number of ground truth positive (i.e., relevant documents in the collection), R_r is the set of relevant documents returned by the system, and $Precision_r(d)$ is the precision of document d considering it has been placed at a rank r. Finally, for a set of queries Q it is possible to obtain the mean average precision over all of the *APs* (one for each query) as:

$$mAP = \frac{1}{|Q|} \sum_{q \in Q} AP(q) \tag{2.10}$$

IR-based QA paradigm

The purpose of IR-based QA is to answer a user's inquiry by locating short text segments on the internet or in any other collection of documents that answer the issue.

The *retrieve and read* approach, as seen in Figure 2.1, is the most common paradigm for IR-based QA. This strategy includes two stages. The first stage in-

volves retrieving relevant sections from a text collection, which is often accomplished through the use of a search engine. In the second step, a neural reading comprehension algorithm goes over each passage and looks for spans of text that are likely to contain the answer to the query [10].

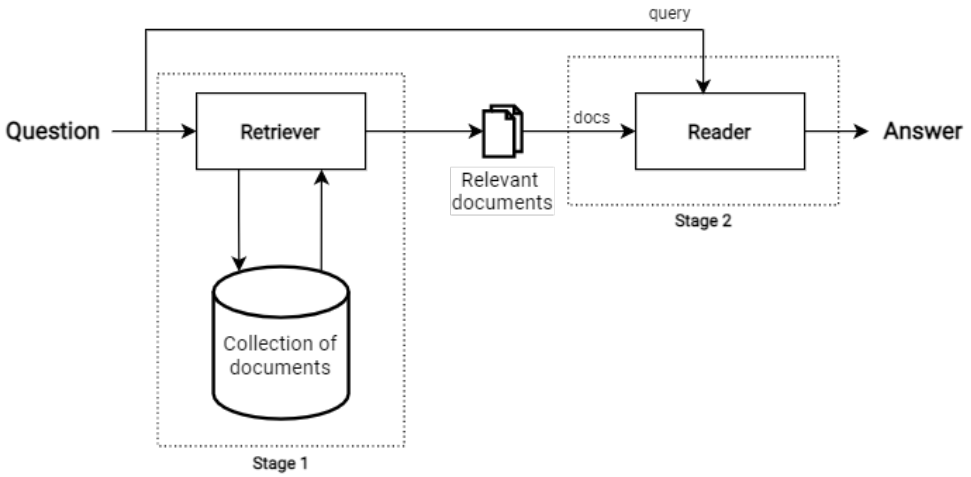


Figure 2.1: Basic 2-stages architecture of an IR system [own work].

An answer to a query and the section in which the answer should be discovered are provided to a system in the reading comprehension task. On the other hand, in the complete QA task (both stages, retriever and reader) systems are not provided with the passage as input, but are instead asked to get this information from a collection of documents on their own. In this two-stage QA task, a triple such as (question, passage, answer) is employed to instruct the reader during the training process. However, when it comes to the inference process (i.e., process in which the system is tested out after being trained), the passages are eliminated and the system is given merely the query, as well as access to the corresponding corpus. The system must then perform IR in order to locate a collection of pages, which it must subsequently read and extract and answer from.

The span labeling task is widely used to approach the answer extraction problem; this is, recognizing in a passage a span that represents an answer. The neural span algorithms for reading comprehension are given a query q of n tokens $q_1, ..., q_n$ and a passage p of m tokens $p_1, ..., p_m$ and are asked to solve the problem. Ultimately, the goal is to determine the conditional probability $P(a|q,p)$ that each potential span a is the answer to the question.

The following assumption is made to make things easier: assuming each span begins with the position a_s and ends with a_e, we may estimate the likelihood of this happening as $P(a|q,p) = P_{start}(a_s|q,p) \times P_{end}(a_e|q,p)$. As a result, for each token p_i in the passage, we will compute two probabilities: $p_{start}(i)$, which indicates that p_i is the beginning of the response span, and $p_{end}(i)$, which indicates that p_i is the end of the answer span [7].

A common baseline for the reading comprehension task is to pass the query and the passage to an encoder such as BERT [11]. It is possible to represent the question as the first sentence of the model, followed by the corresponding [SEP] separator, and then the passage as its second sentence. In addition, a softmax linear layer would need to be added to compute the probability of each token being the beginning and the end of the potential answer. Two special vectors would need to be incorporated: a span-start embedding S and a span-end embedding E, that would be learned during fine-tuning. The span-start probability for each token p'_i can be calculated as the dot product between S and p'_i and then getting a probability distribution applying softmax [11]:

$$P_{start_i} = \frac{e^{(S \cdot p'_i)}}{\sum_k e^{(S \cdot p'_k)}} \tag{2.11}$$

While P_{end_i} can be extracted with the analogous formula, candidate spans from index i to index j within the passage are ordered according to their score (i.e., $S \cdot p'_i + E \cdot p'_j$), and the one maximizes its value whilst $j \geq i$ is met is chosen as the prediction (output) of the model.

Evaluation of answers

Factoid answers are often evaluated using mean reciprocal rank (MRR) [12]. MRR is intended for systems that produce a short ranked list of answers or passages for each test set question, which can then be compared to the (human-labeled) right answer. Each question in the test set is rated with the complementary of the first correct answer; for instance, if the system returned six answers to a query but the first two are incorrect (i.e., the highest-ranked correct answer is placed third) the reciprocal rank would get a value of $\frac{1}{3}$. In this way, if we consider Q as the set of test questions, and a ranked list of answers returned by the system in which the score for a question that returns no correct answer takes a value of 0, the MMR of the system is the average of the scores for each of the questions [7]:

$$MMR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \tag{2.12}$$

Two other metrics used in the literature when evaluating the performance of a QA system are the exact match and score. The former simply assesses whether or not the answer proposed by the system match exactly (all terms) the ground truth answer, and the evaluation of the system takes form as the proportion of predicted answers that meet this exact match. The latter (F_1) is widely used for classification problems but also in the field of QA. More precisely, it is determined by comparing each individual word in the prediction with each word in the ground truth answer. When comparing a prediction and its truth, the F_1 score is calculated as follows: precision corresponds to the ratio of the number of shared words in

the prediction to the total number of words in the ground truth, and recall is the ratio of the number of shared words to the total number of words in the ground truth. It is calculated as follows:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.13}$$

2.1.2 Knowledge-based techniques

There is a wealth of knowledge in the vast volume of text on the internet, but there is also information available in more organized formats. Knowledge-based question answering is the idea of responding to a natural language inquiry by translating it to a query over a structured database [7].

When it comes to knowledge-based question answering, there are two main paradigms. Graph-based QA represents the knowledge base as a graph, with entities acting as nodes and relations or propositions acting as edges between nodes. The other approach is QA by semantic parsing, which makes use of semantic parsing techniques.

Named Entity Linking

A named entity is a real-world item that can include people, places, organizations, and other things. The aim of named entity recognition (NER) is to identify and categorize occurrences of named entities in text into pre-defined categories based on their appearance. This task of assigning tags to each word in a phrase is represented by the NER task. Nevertheless, when recognizing a certain entity in a text the system does not know which entity it corresponds to in the knowledge base, and there is where entity linking plays an important role. Named Entity linking (NEL), also known as entity linking, is the task that links entity mentions in text with their corresponding entities in a knowledge base [13].

The task of NEL is not as straight forward as one could think. When analyzing the challenges that may arise in the process, the variation problem appears when a same entity within the knowledge base is referred to as different mentions in the text. An example of this can easily occur with names, wherein the entity Michael Jeffrey Jordan can also be expressed as Jordan, MJ, or Michael Jordan, for instance. Furthermore, the ambiguity problem is related that a mention may refer to distinct entities depending on the context. For example, the name Washington may be referring to the first president of the United States, George Washington, or to the capital of the same country [14].

Knowledge-Based QA from RDF triple stores

A semantic triple, often known as an RDF triple or just a triple, is a data object in the Resource Description Framework (RDF) data model that represents

an atomic data entity. As its name implies, a triple is a group of three things that together encode a statement about semantic data in the form of subject-predicate-object phrases. Triples are commonly used in computer science and mathematics in the form subject–predicate/relation–object (e.g., "Bob is 35", or "Bob knows John").

After entity linking has been completed, a basic pipeline for a basic knowledge-based QA system can be described. Starting with the most basic example of graph-based QA, in which the dataset is a collection of factoids represented as RDF triples and the goal is to provide answers to inquiries regarding one of the missing parameters. Consider an RDF triple that states the following:

Table 2.1: Example of RDF triple.

subject	**predicate/relation**	**object**
Rafael Nadal	birth-country	Spain

Such a triple may be employed to get answer to queries like “Who was born in Spain?” or “Where was Rafael Nadal born?”.

Assume that we have previously completed the entity linking stage that was discussed previously. As a result, a mapping between a textual reference of Rafael Nadal and the canonical entity ID in the knowledge base has already been created. When addressing a basic triple relation query, the next step is to discover which relationship is being asked about by mapping from a string such as "In which country was... born" to canonical relations in the knowledge base such as birth-country. The following is a rough outline of the combined task:

Table 2.2: Example of mapping from query to canonical relation.

“In which country was Rafael Nadal born?”	→	birth-country (Rafael Nadal, ?x)
“What is the currency used in Norway?”	→	currency-used (?x, Norway)

The next phase is the relation discovery and linking. Detection and linking of simple questions, where we assume that the question has only one possible relationship, can be accomplished in a similar manner to that of neural entity linking models, this is, by computing similarity (generally by dot product) between the encoding of the question text and an encoding for each possible relation [7].

QA by semantic parsing

In the second form of knowledge-based QA a semantic parser is used to convert a query to a structured program, which then generates a response. These logical forms can be represented through a variant of a predicate calculus, a query language such as SQL or SPARQL, or any combination of these [15]. It is possible to fully supervise semantic parsing algorithms with questions that are paired with a hand-built logical form, or to weakly supervise semantic parsing algorithms with

questions that are paired with an answer, in which case the logical form is represented only as a latent variable [7].

Table 2.3: Examples of representation of queries in logical form (λ-calculus)

Question	Logical form
What countries border Spain?	$\lambda x.country(x) \wedge borders(x, Spain)$
What is the tallest building?	$argmax(\lambda x.building(x), \lambda x.height(x))$

An example of the representation of a question in its logical form is provided in Table 2.3. The next step is to take those pairs of training tuples and build a system that maps new questions to their equivalents in a logical fashion. A basic sequence-to-sequence model, for example, utilizing BERT to represent question tokens and feeding them to a biLSTM encoder-decoder, is a frequent baseline approach [7].

2.2 Conversational agents

Dialogue systems, also known as conversational agents, may be divided into two categories: task-oriented dialogue agents and open-domain conversational agents (also known as chatbots). The former engages in communication with customers to assist them in completing tasks (such as order food, send a text to someone, make a call, navigate on the map, etc.). Contrary to this, chatbots are systems developed with the goal of having extended conversations, arranged to emulate the unstructured conversations or "chats" distinctive of human-human interaction, primarily for entertainment purposes, but also for practical purposes such as attempting to make task-oriented agents more natural in their interactions with users.

Turns, speech actions, grounding, dialogue structure, initiative, and implicature are just a few of the complex qualities of human talks (along with others) that make it challenging to develop dialogue systems that can carry on realistic conversations with people. Many of these issues are now being researched in the field of conversation systems.

2.2.1 Rule-based conversational agents

This type of chatbot knows how to have simple talks based on if-then reasoning. As these systems are incapable of understanding context or purpose, it becomes impossible for them to respond to any inquiries that do not fall inside the defined rules. They do not learn from interactions and only execute and work within the context of the scenarios for which they have been specifically trained for [7].

2.2.2 Generative conversational agents

When compared to the closed domain design, which concentrates on answer selection from a collection of already established replies, the open domain architecture allows us to undertake unlimited text production. Closed domain systems make use of purpose categorization, entity identification, and response selection to accomplish their goals. However, with a generative conversational agent (also known as open domain chatbot), intent categorization is more difficult, and an enormous number of intents is likely to be encountered. A generative model, rather than picking whole replies, creates responses word by word, allowing for the creation of novel combinations of language.

When constructing a sentence it is crucial to be aware of the context of the conversation. The formation of better sentences in natural language dialogues between people is enhanced when the meaning of words is considered in relation to their context. As a result, different words may have multiple meanings in different situations, which makes it more difficult to extract the key semantic logic flow of each sentence in several paragraphs.

Humans do not start at the beginning of every second when they think. For instance, as you read this text you make sense of each word depending on the comprehension we get from the words that came before it. This avoids dumping everything and start over from the beginning with our thoughts to get the meaning of the next word.

A first approach for this issue is artificial neural networks (NNs). However, this is something that traditional neural networks are unable to perform, and this appears to be a significant deficiency. Consider the following scenario: you want to categorize the sort of event that is taking place at each time in a song you are listening to. Under this scenario, the exact way in which a typical neural network may utilize its reasoning about prior events in the song to inform later ones is not clear. This problem is addressed by recurrent neural networks (RNNs).

Recurrent Neural Networks (RNNs) and their long-term dependency problem

RNNs are networks having loops in them, which allow information to survive in the network over time. When it comes to classification, recurrent neural networks are dynamic systems, with an internal state at each time step of the process. This is attributed to the existence of circular connections between distinct neurons, as well as the presence of optional self-feedback links. Because of these feedback links, RNNs are able to transfer data from prior events to the present processing stages [16]. As a result, RNNs accumulate a recollection of time series events.

In Figure 2.2, a chunk of NN takes some input X_t and outputs a value Y_t. These loops allow information to be passed from one step t of the network to the next one $(t+1)$.

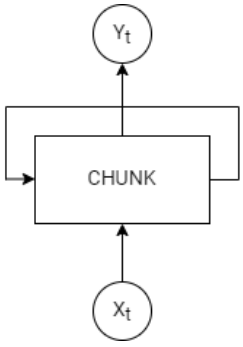


Figure 2.2: Basic RNN cell [own work].

In particular, the idea that recurrent neural networks might be able to connect previous information with the current task is appealing. For instance, the understanding of a previous movie scene might inform one's understanding of the current scene using the previous ones. If RNNs were capable of doing this, they would be immensely beneficial, although their performance against prior experiences depends on how much of context is needed.

Sometimes all we need to do is refer to recent facts in order to complete the current work. Consider the case of a language model that attempts to anticipate the next word based on the words that have come before it. If we are attempting to predict the final word in "the boats navigate in the <...>," we do not require any additional context as it is fairly obvious that the next word will be "sea" in this sentence. In such circumstances, where the distance between the relevant information and the location where it is required is short, RNNs can learn to make use of the knowledge from the past.

Nevertheless, there are some instances in which we require additional context. Consider attempting to guess the final word in the text, "I grew up in Norway... I speak fluent Norwegian". In this case, recent information within the sentences it appears that the next word is most likely to be the name of a language; however, in order to specify which language it is, we must look at the context of France from the beginning of the sentence (or even paragraph). This means that it is absolutely conceivable for a significant distance to exist between relevant information and the stage at which it is required.

When the mentioned gap widens, the performance of RNNs decreases as they are not able to connect the pieces of information. Bengio et al. [17] present theoretical and experimental data demonstrating that gradient descent of an error criterion (criterions compute a gradient according to a given loss function, given an input and a target) may be insufficient for training them for tasks with long-term dependence on the environment, in which gradient descent becomes increasingly inefficient when the temporal span of the dependencies increases.

Long Sort-Term Memory (LSTM) Recurring Networks

LSTMs are a particular type of RNNs that are designed to address the long-term dependency problem. LSTMs have a chain-like structure as well (loops); however, the repeating module has a different structure. There are four neural network layers (rather than single ones), each of which interacts in a unique manner [18].

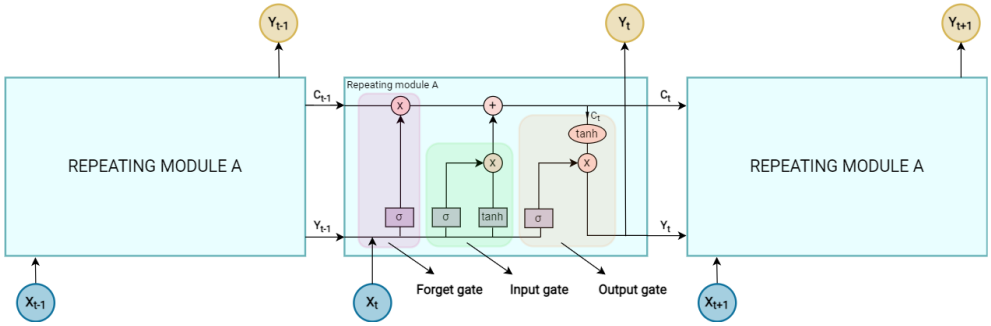


Figure 2.3: Interactive layers in LSTM repeating module [own work]

The cell state, represented by the horizontal line running through the top of Figure 2.3, is the main element of LSTMs. The cell state is similar to that of a conveyor belt. It proceeds in a straight line along the whole chain, with only a few linear interactions along the way. It is relatively easy for information to just travel through it in an unaltered state.

One of the features of the LSTM is that it has the capability of removing or adding information to the cell state. This ability is carefully controlled by structures called gates.

Gates are a method of allowing or disallowing information to pass through. They are made up of a sigmoid neural network layer (represented by the symbol sigma in the diagram) and a pointwise multiplication operation (x in the diagram). Sigmoid layers generate values between zero and one that describe how much of each component should be allowed to pass through the filter. Zero indicates that nothing should be let through, whereas one signifies that everything should be allowed through. A basic LSTM model has three of these gates, which are responsible for protecting and controlling the cell state [18].

If we were to describe what are the steps in which information flows through a repeating module in a LSTM, we first encounter the forget gate. This gate is responsible for noticing if there is a change in the context and then forget previous information if appropriate. It is formed by a simple neural network (sigmoid function) that looks at the previous state Y_{t-1} and the current input X_t and outputs a number between 0 (indicating to forget this value) and 1 (indicating to keep this value) for each number in the cell state C_{t-1}. For example, assuming that the vector Y_{t-1} and X_t represent sentences such as "Tennis and table tennis are sports in which a racket is used" and "Table tennis is also known as ping-pong and is played on a special table", respectively, the forget gate would notice that the next sentence is about table tennis and information regarding tennis may be omitted.

The input gate is responsible for deciding how much of the new information

is added to the cell state. It is made up of a sigmoid function, which determines which values are allowed to pass through with zeros and ones. The tanh function assigns a weight to the values that are supplied, with the weight ranging from -1 to 1 depending on the significance of the data. With the use of the input gate we make certain that the only information that is put to the cell state is vital information that does not include any redundancies. For instance, following the previous example stated in relation to the previous gate (forget gate), if the new information X_t were to be "A friend of mine told me yesterday that the space of the table in which table tennis is played is smaller than it is for a tennis court", the relevant data would be the fact that the available space in table tennis is smaller than in tennis, and the "a friend told me yesterday" may be catalogue as irrelevant (thus can be skipped).

Finally, the output gate controls which portions of the current cell are sent to the output. As in the other two gates, the sigmoid function is in charge of letting values through. The tanh function assigns a weight to the values that are passed from the cell state C_t, determining the importance of those values on a scale from -1 to 1, and then multiplies that value by the output of the sigmoid function.

Encoder-Decoder architecture

Deep Neural Networks (DNNs) are a strong type of machine learning model that, when applied to challenging problems like speech recognition and visual object detection, yield great results. However, they can only be utilized with huge amounts of data that have been labeled, and the inputs and targets must be able to be represented coherently using vectors of a certain dimension. They are not suitable for mapping one sequence onto another sequence (e.g., machine translation) [19]. Sequence-to-Sequence tasks, also known as Seq2Seq problems, belong to a subcategory of Sequence Modelling Problems that are unique in that both the input and the output of the problem are sequences. The Encoder-Decoder models were initially developed as a solution to this type of tasks.

An LSTM cell makes up the encoder portion of the circuit. It attempts to encapsulate all of its information and store it in its ultimate internal states, which are h_t (hidden state) and C_t (cell state). It gets fed in the input-sequence throughout time steps. The decoder component receives the information about the internal states (known as context vector), which it then puts to use in an effort to generate the target sequence. The outputs of the encoder component at each time step are discarded.

Once the encoder has finished reading the whole input-sequence, it then sends the internal states on to the decoder, and it is at this point that the prediction of the output-sequence starts. Similarly to the encoder block, the decoder block is an LSTM cell. An important point to note is that the initial states (h_0, C_0) of the decoder are set to the same values as the end states (h_t, C_t) of the encoder. The decoder is able to construct the necessary target-sequence with the assistance of these, which serve as the context vector.

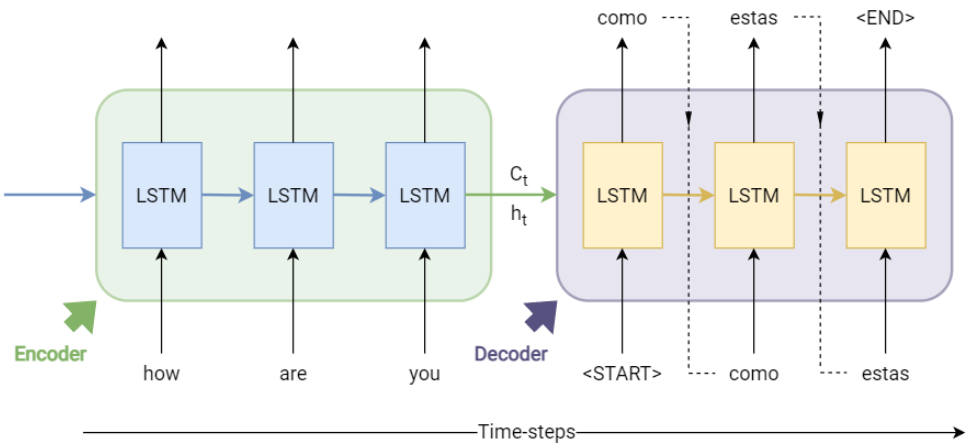


Figure 2.4: Encoder-Decoder model architecture performing machine translation from english to spanish [own work]

The way that the decoder operates is such that its output at any given time-step t is meant to represent the t^{th} word in the ground-truth sequence ("como estas" in the example from the Figure 2.4). Let us examine what takes place at each time step to better understand this concept.

At the beginning of the first time step, the input that is being sent to the decoder is the unique symbol $< START >$. This is how the beginning of the output series is shown by the system. Now, the decoder takes this input and utilizes it along with the internal states (h_t, C_t) to generate the output in the first time step. This output is expected to be the first word or token in the target sequence (i.e., "como").

The output of the first time step, which is denoted by "como", is used as an input for the second time step. It is anticipated that the word "estas", which comes second in the target sequence, will be the output of the second time step.

In a similar manner, the result of one time step is used as the input for the subsequent time step. This continues until we reach the $< END >$ token, which is another unique symbol that denotes the conclusion of the output-sequence. Once we reach this symbol, the process is complete. The decoder's ultimate internal states are discarded.

The output $y_{t_{pred}}$ of the decoder, at any given time-step t, represents the probability distribution over the entirety of the vocabulary contained in the output dataset. This distribution is produced by making use of a softmax activation function. It is decided that the token with the highest probability will serve as the token that will be anticipated.

It is important to keep in mind that $< START >$ and $< END >$ are being used as a representation and they are not the only possible special symbols for this purpose. The fact that these strings do not appear in our data corpus means that the model will not confuse them with any other words, thus being interchangeable.

The procedure described above is how a decoder would operate when it per-

forms inference over certain instances (testing). However, during the training stage, it is trained in a slightly different way. During training, the ground-truth output or token from the prior time step, rather than the output or token that was predicted, is used as input for the current time step. This technique has been introduced to rectify slow convergence and model instability problems that were encountered when using the predicted words as input for the different time steps [19].

Ultimately, the loss is computed based on the anticipated outputs from each time step, and mistakes are back-propagated over time in order to update the model's parameters. The loss function that is used is the categorical cross-entropy loss function, and it is applied between the target sequence and the sequence that was predicted.

Since one-hot-encoded vectors can be large, and the embedded vectors provide a much better representation of words, the input-sequence is sent through an embedding layer in both the decoder and the encoder. This reduces the dimensions of the input word vectors and is usually applied regardless the stage the model is being used for (either training or testing).

2.2.3 Corpus-based conversational agents

Corpus-based conversation agents, as opposed to utilizing predefined rules, mine conversations that have already taken place between humans. These systems have a very high demand for data and require millions, (or even billions) of words in order to be trained.

Movie dialogues, databases with talks about certain subjects, and other types of conversations are used to train these systems. Although these datasets are big, they do not approach the scale of billions of words. Because of this, many systems are first pretrained on massive datasets of pseudo-conversations gathered from social media sites such as Twitter or Reddit [20]. Another typical method is to extract plausible replies from knowledge sources, such as Wikipedia and news articles, so that an agent may tell stories or cite facts that it has gained in this manner.

After a conversational agent has been implemented, the turns that people use to answer to the conversational agent may be utilized as extra conversational data for the purpose of training or fine-tuning the agent. It is essential to have confidence metrics in this regard in order to guarantee that these turns result from dialogues that are proceeding favorably [21].

The majority of corpus-based chatbots produce their responses to a user's turn in context either by retrieval techniques (employing information retrieval to capture an answer from certain corpus that is relevant given the dialogue context) or generation methods, producing a response based on the conversation context with the usage of a language model or encoder-decoder to create the response given the dialogue context). In either scenario, systems will often only provide a single answer turn that is suitable given the entirety of the discussion thus far

(this applies to dialogues that are brief enough to be included inside the window of a single model). Because of this characteristic, they are also refered to as response generating systems. The algorithms that are used in corpus-based conversational agents are derived from the methods that are used in question-answering systems. These algorithms mainly concentrate on single replies while neglecting longer-term conversational goals [7].

Response by retrieval

The retrieval technique of replying involves treating the user's turn as a question q, and the task consists in selecting and repeating a suitable response r from a collection C of previous conversations. In most cases, the training set for the system is also designated by C. A score to each possible turn in C that may be a response to the context q is obtained and then we choose the turn with the highest score. The similarity between the two texts is used as the measure for scoring, and we select the r that is most comparable to q by referring to any of the IR approaches that were described in Section 2.1.1. This may be accomplished by employing traditional IR methods in order to generate $TF-IDF$ models for C and q, and then selecting the r that possesses the greatest $TF-IDF$ cosine value with q in the following way [7]:

$$response(q, C) = \arg\max_{r \in C} \frac{q \cdot r}{|q||r|} \tag{2.14}$$

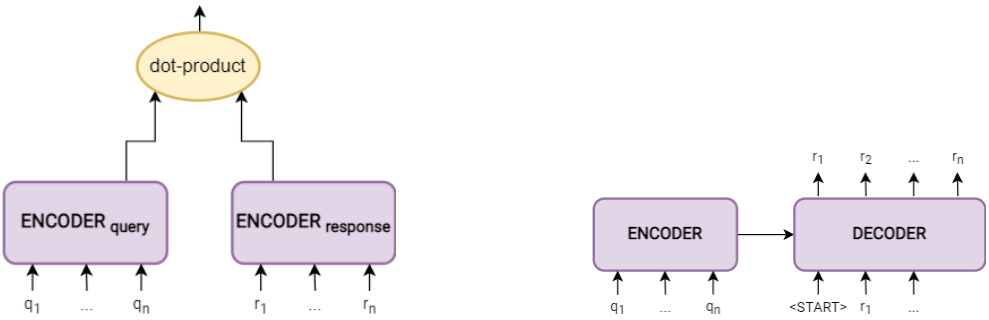
Alternately, a neural IR method may be employed. The simplest of these models is called a bi-encoder model, and it entails training two independent encoders: one to encode the user question, and another to encode the potential response. Then, the score is calculated by taking the dot product of these two vectors (Figure 2.5a).

Response by generation

One other method of using a corpus to produce conversation is to consider answer creation to be an encoder-decoder task. This involves the process of transitioning from the user's previous turn to the agent's turn. We may think of this as a form of ELIZA [22] that uses machine learning; the model learns from a corpus to convert a query into a response. The concept that response creation is a form of translation inspired the generalization of the encoder-decoder paradigm. One way to think of response generation is as a translation.

Encoder-decoder models produce each token r_t of the answer by depending on the representation of the complete question q and the response collected until certain point $r_1 ... r_{t1}$ [7]:

The intuitive workings of the generator and retriever approaches for answer generation are illustrated in Figure 2.5. A more extensive context is often incorporated into the generator design, allowing the creation of inquiries based not



(a) Response by retrieval **(b)** Response by generation

Figure 2.5: Two different architectures that may be used to generate replies for a neural conversational agent. An answer is selected for response by retrieval (a) by locating the turn in the text (conversation) whose encoding has the highest dot-product with the user's turn. An encoder-decoder model may be used to produce the answer for the response by generation method (b) [own work]

just on the user's turn but also on the complete dialogue that has taken place up to this point. An example is illustrated in Figure 2.6.

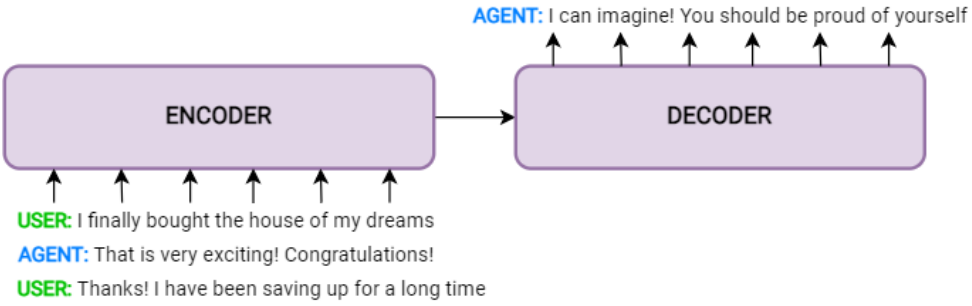


Figure 2.6: Example of encoder-decoder model in which the encoder uses the entire conversation context [own work]

In order to make the basic encoder-decoder model suitable for the activity of response creation, a number of adjustments need to be made to it. Encoder-decoder models, for instance, have a propensity to provide predictable but repeated and, as a result, uninteresting replies such as "OK" or "I don't know", which conclude the conversation. We may thus utilize diversity-enhanced variants of beam search or diversity-focused performance goals in place of greedily selecting the response that is most likely to occur (and hence the most expected) [23]. The most fundamental models have a tendency to generate statements that are too brief; thus, it is essential to include minimum length limitations [20].

The encoder-decoder design has an alternative, which is to fine-tune a big language model on a chat dataset and then utilize the model directly as an answer producer. For instance, the neural chat component of the Chirpy Cardinal system

(which is a social conversational agent) [24] generates replies using GPT-2, which are then fine-tuned using a different dataset.

Encoder-decoder answer generators concentrate on producing a single answer at a time; as a result, they do not often do a very good job at producing continuous replies that are consistent across a number of turns. This issue can be remedied by employing techniques such as adversarial networks and reinforcement learning in order to acquire the ability to select replies that contribute to the entire dialogue appearing more genuine [25].

Chapter 3

Related work

Over the past several years, there has been a significant growth in the quantity of work completed in both the field of question answering and generative transformer-based models. This chapter provides an introduction to some of the most notable articles and research projects in various fields that connect to the work in this thesis, and attempts to answer the first research question.

3.1 Language models

3.1.1 Question answering models

BERT

One of the most significant obstacles in natural language processing is the absence of sufficient training data. If we wish to generate datasets that are unique to a job, we need to divide the massive quantity of text data that is accessible into the very many different categories. Overall, there is a huge amount of text data that is available. When this is done, the number of human-labeled training samples that we are left with is anywhere between a few thousand and a few hundred thousand. Deep learning-based natural language processing models, however, require substantially bigger volumes of data in order to function well. These models demonstrate significant gains when trained on billions of annotated training instances. Researchers have created a variety of methods for training general purpose language representation models utilizing the massive amounts of unannotated text that are available on the internet in order to assist bridge this gap in data (known as pre-training). When working with challenges such as question answering and text classification, for example, these general purpose pre-trained models may later be fine-tuned on shorter datasets that are particular to the job at hand. When compared to training on the smaller datasets that were unique to the job from the beginning, this strategy yields significant gains in accuracy. Bidirectional Encoder Representations from Transformers (BERT) [26] is a relatively new addition to these methods for NLP pre-training. It generated a commotion in

the community of deep learning researchers since it produced results that were state-of-the-art in a broad variety of NLP tasks, including question answering.

In comparison with the research prior to BERT, it stands out for being bidirectional trained. In comparison to the language models that only go in one direction, this indicates that we can now have a more comprehensive understanding of the context and flow of language. BERT employs a revolutionary method known as Masked LM (MLM), which masks words in the phrase at random before attempting to predict which masked word will come next in the sequence. This is done in place of the traditional method of anticipating the next word in a sequence. In order to correctly forecast the masked word, the model must first perform the process of masking, which involves looking in both directions and taking into account the entire context of the phrase, including its left and right surroundings. It is distinct from the earlier language models in that it simultaneously considers both the tokens that came before and those that will come after them.

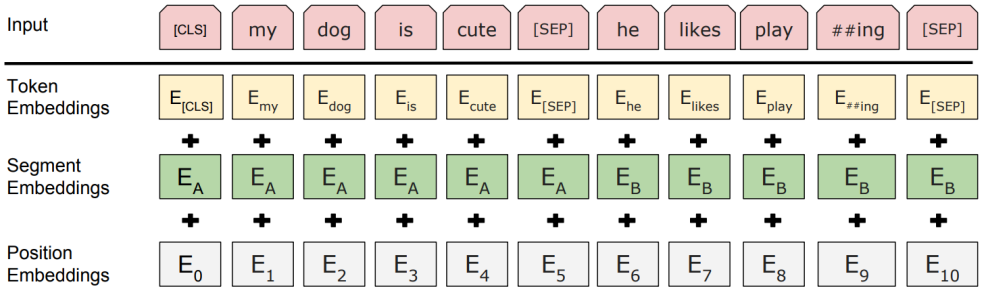


Figure 3.1: BERT embeddings [26]

A succession of tokens serves as the input for the encoder for BERT. These tokens are first transformed into vectors before being sent on to the neural network for processing. However, in order for the processing to begin, BERT requires the input to be modified with certain additional metadata. One of this extra data is the special tokens $[CLS]$, which is inserted at the beginning of the first sentence, and $[SEP]$, which indicates the separation between two sentences. Segment embeddings involve adding a marker to each token that indicates whether it belongs to one sentence or the other. Because of this, the encoder is able to differentiate between sentences. As the last supplementary data, a positional embedding is appended to each token in order to specify where in the phrase the token is located.

Masked tokens and next sentence prediction were the two methods that have been utilized during the training process. On one hand, randomly masking out 15 percent of the words in the input and replacing them with a $[MASK]$ token was carried out, running the entire sequence through the BERT attention based encoder, and then anticipating only the words that have been masked based on their context that is given by the rest of the words that appear in the sequence (the ones that were not masked).

On the other hand, BERT's training stage additionally makes use of next sen-

tence prediction in order to comprehend the connection that exists between the two phrases at hand. In order to be useful for activities such as question answering, a model should have been pre-trained with this sort of understanding. During training, the model is given sentences in pairs as input, and it learns to determine whether or not the second sentence is also the next statement in the original text.

RoBERTa

A couple of months after Google's work (BERT) was published, Meta AI Research (previously known as Facebook AI Research or FAIR) detected room for improvement in Google's model. A Robustly Optimized BERT Pretraining Approach (RoBERTa) [27] was then proposed to enhance what its authors suggested as a substantially under-trained model.

One of the important changes applied to BERT has to do with the way masking is done during training. During the data pre-processing stage, the masking is performed just once; as a result, there is only one static mask. Therefore, the model was always given the exact same set of input masks at each and every epoch. Meta utilized dynamic masking so that it would not repeatedly cover up the same word more than once. The training data was iterated ten times, and each time after that, the word that was covered up would be different. This means that the masked words would be distinct while the sentence remains the same.

Another area for improvement is the format of the next sentence prediction (NSP). The <segment-pair+NSP loss> format was first implemented as BERT's primary input format. Each input in this case is comprised of a pair of segments, each of which may include several phrases; nevertheless, the overall length of all of these sentences must be fewer than 512 tokens.

It has been observed that the use of individual utterances is detrimental to the performance of downstream tasks. The reason for this, as per the hypothesis, is that the model was unable to understand long-range correlations; consequently, the authors could conduct an experiment to determine the effect of removing or adding NSP loss on the capability of the model. As a first experiment and aiming to keep the NSP loss, they changed the input to be <sentence-pair+NSP loss>, containing then a pair of natural phrases (instead of segment), taken from a continuous section of a single text or from many documents separately. Contrarily, in an attempt to remove the NSP loss from the input format, Meta's researchers explore filling each input with complete phrases taken in sequence from a single text or across many documents (which they called <full-sentences>), with the goal of keeping the overall length to a maximum of 512 tokens. In a similar manner, inputs were constructed the same way with the only difference of not being allowed to exceed document boundaries (denominated <doc-sentences>). Because inputs sampled closer to the conclusion of a document could have fewer than 512 tokens, they dynamically raised the batch size when they encountered one of these scenarios so that the total amount of tokens could be reached.

In contrast to the original BERT, which included NSP loss, the performance is

slightly better when the NSP loss is removed. The sequences from a single document (<doc-sentences>) perform somewhat better than packing sequences from numerous documents (<full-sentences>) as indicated in Table 3.1.

Table 3.1: Performance (accuracy) of BERT_{BASE} (with NSP) vs. RoBERTa FULL-SENTENCES and DOC-SENTENCES (without NSP) [27]

Model	**SQuAD 1.1/2.0**
FULL-SENTENCES (without NSP)	90.4/79.1
DOC-SENTENCES (without NSP)	90.6/79.7
BERT_{BASE}	88.5/76.3

The last significant alteration has to do with some of the training parameters. When BERT was first trained, it was trained for 1 million steps, and each batch had 256 sequences. This indicates that there is room for development in the complexity of the Masked Language Modelling (MLM) aim.

It has been shown that enhancing the complexity of the MLM aim while simultaneously increasing the final accuracy may be achieved by training a model using large mini-batches. For instance, the computing cost of a batch size of 256 is comparable to that of a batch size of 8K, which contains 31K steps. Additionally, it is simpler to parallelize the training of large batches using distributed parallelism.

It was shown that training BERT on bigger datasets led to a significant improvement in its overall performance. As a result, the size of the uncompressed text training data was raised to 160 gigabytes.

ELECTRA

In order to be back in the discussion about the latest developments in the field of question answering, Google put efforts in conceiving Efficiently Learning an Encoder that Classifies Token Replacement Accurately (ELECTRA) [28]. ELECTRA makes use of a new pre-training task known as replacement token detection (RTD). This task trains a bidirectional model (similar to an MLM) while simultaneously learning from all input locations. ELECTRA is a machine learning framework that was developed with the help of generative adversarial networks (GANs). It trains models to differentiate between "genuine" and "fake" input data. Rather than corrupting the input by substituting tokens with "[MASK]," as BERT does, the method the authors created consists in corrupting the input by altering selected input tokens with fakes that are incorrect but still seem convincing. For instance, in the picture that can be found below, the word "cooked" is substituted with the word "ate." Even while this makes some logic, it doesn't really work when taken into consideration with the rest of the context.

This new strategy of replaced token detection involves training not one but two neural networks: a generator (G) and a discriminator (D). Each one is made

up largely of an encoder that converts a stream of input tokens into a stream of embeddings that are contextualized. The discriminator will next determine whether or not it is real based on an analysis of the distribution of its data.

The generator, trained to carry out masked language modelling or MLM, starts by picking at random a set of tokens to mask off the input. Tokens are removed from the spots that have been chosen and changed with a [MASK] token taken from the generator. After that, the generator acquires the knowledge necessary to optimize the probability of masked-out tokens. The discriminator is taught to identify tokens in the data against tokens that were substituted by generator examples.

Google's team achivied comparable results to the ones from RoBERTa while utilizing less than 25% of computing resources. Furthermore, its large version (with 1.75M parameters) reached a F1 score of 91.4% against 89.8% from RoBERTa-Large on the SQuAD 2.0 test set.

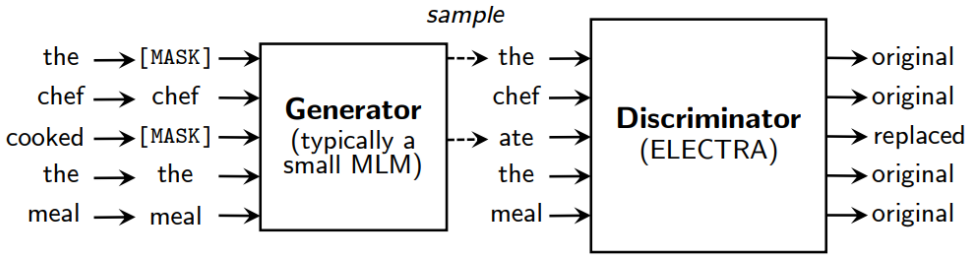


Figure 3.2: ELECTRA's replaced token detection architecture [28]

3.1.2 Generative transformer-based models

GPT-3

Standing for Pre-Trained Generative Transformer 3 (GPT-3) [29] is one of the most cutting-edge language models that has been developed. The outcome of GPT-3 is a statistical rational answer to the input that was provided to it, which is founded on data that was used to train the model. The GPT-3 algorithm computes the probability that a particular word will appear in the text due to the presence of other words (also known as conditional probability of words). As an example, considering the sentence "I heard the concert is getting sold out very quick, we should buy the ", there is a considerably greater possibility for the word "tickets" of being suggested in the text than for the word "pizzas".

While it processes lots of sample texts and does some type of data compression in the process, GPT-3 converts words into vectors, which are numerical representations of the words. After that, the language model will begin the process of decoding the compacted text into phrases that are understandable to humans. Therefore, compressing and decompressing text helps enhance the accuracy of the model whilst simultaneously computing the conditional probability of words.

GPT-3's outstanding performance in "few-shot" conditions allows it to reply in a way that is consistent with an particular text which has not been presented to it beforehand. Due to its extensive training with numerous text samples, it just requires a small number of instances to provide an appropriate response.

While being a state-of-the-art generative model, it also presents several limitations. GPT-3 has a number of structural and algorithmic flaws. An autoregressive language model was chosen for the study of in-context learning behavior because sampling and computing likelihoods are straightforward with a model of this type. As a direct consequence of this, none of the trials that were carried out make use of bidirectional designs or any other forms of training with goals such as grammar correction (denoising). This is a marked contrast to the majority of the research that has been published in recent years, the majority of which has proven an improvement in fine-tuning performance when utilizing these methodologies rather than typical language models [30]. Therefore, the judgment the authors made during development might result in a decrease in performance on tasks that scientific work has shown to be improved by bidirectionality.

Moreover, there are still challenges in terms of bias, fairness, and representation due to the fact that the model was fed with content that people created on the internet during training. As a result, GPT-3 has the potential to produce information that is biased or stereotypical.

DialoGPT

Researchers have been motivated to use pretraining approaches to conversational artificial intelligence as a result of the empirical success of these methods in other fields of natural language processing. On the other hand, these models are trained on traditional written language, which is not always indicative of how people communicate with one another. DialoGPT (Large-Scale Generative Pretraining for Conversational Response Generation) [31] pushes transformer-based pretraining a step beyond to exploit huge volumes of publicly available conversational text data. With this model, Microsoft's objective is to achieve the topic variety given by scale while also achieving a tone that is more conversationally engaged.

The vast textual datasets (such as wikis and news articles), which were used to train earlier versions of pretrained models, differ from the interactive data that was used to train DialoGPT. It is less formal, more engaging, often somewhat friendless, and generally with more noise than other settings. DialoGPT, in contrast to GPT-2, which trains on generic text data, relies on 147 million multi-turn conversations gathered from Reddit comments threads, thus adapting pretraining approaches to form a relevant answer utilizing an important amount of informal data.

The model achieved state-of-the-art results in what the authors called automatic evaluation, specifically in DSTC-7 Challenge response generation task [32], with a Bilingual Evaluation Understudy B-2 (BLEU B-2) of 19.18% compared to

14.35% of the system in second place. The end-to-end conversational modeling challenge that is presented in DSTC7 has the objective of generating conversational replies that go beyond meaningless small talk by introducing meaningful responses that are founded in external knowledge. In contrast to what is conventionally understood to be goal-oriented or task-oriented dialog, wherein there is neither a predetermined nor an explicitly stated aim (for example, when setting up an alarm or adding a new event to your calendar). Instead, it focuses on interactions that are analogous to those between humans, in which the ultimate purpose is frequently unclear or not known exactly. Many dynamic contexts (as it could be planning meetings) have these types of interactions going on all the time.

3.2 QA and transformer-based models combination

Long et al. [33] proposed Search Engine Enhanced Response Generator (SEARG), and detailed in the corresponding paper their attempts at creating natural and instructive replies for customer service driven interaction that incorporates external knowledge. Using a search engine, the system compiles all of the external knowledge relevant to a particular conversation. Afterwards, a knowledge-enhanced sequence-to-sequence framework is built to simulate multi-turn conversations based on the conditional application of external knowledge. On the basis of a list of preset customer service accounts, the entire system was trained and assessed using the DSCT6 dataset [34] received from Twitter.

When it comes to creating responses, the system includes a query encoder and decoder as well as a knowledge encoder and decoder for the extraction of knowledge. To collect dialog-related information from the internet, the knowledge extractor uses a provided dialog history, including the most recent inquiry. After that, a knowledge encoder based on a Convolutional Neural Network (CNN) is introduced to extract features from the gathered information. Finally, using the Long Short-Term Memory (LSTM) as encoder to reflect the conversation history and user input into a legitimate vector, the decoder then produces a response grounded on the dialogue history vector and information attributes in the knowledge enhanced sequence-to-sequence model.

Lowe et al. [35] present a method for finding meaningful replies based on external knowledge sources. This method moves closer to unstructured domains, although it is still within the context of task-driven dialogues. The model obtains pertinent information from the Ubuntu man pages, then employs that information to get a pertinent context-answer pair that is consistent with the information that has been gleaned.

In [36], the possibility of including external sources in conversational systems as a framework to augment the conventional context encoding and to promote the generation in order to be more precise with an accelerated learning time is explored. In addition, particularly in the case of chit-chat systems, scholars assert that information may be acquired from a variety of subjects (e.g., fashion,

cuisine, travel). Sequence-to-Sequence (Seq2Seq) agents trained with the Reddit News dataset are the primary focus of this work. Additionally, they investigate the possibility of combining external information from Wikipedia summaries [37] as well as from the NELL knowledge base [38]).

3.3 Datasets

While the primary datasets employed in this thesis will be covered in Chapter 4, there are additional datasets that are relevant in the field of question answering and text generation, and for which much research has been done.

The Conversational Question Answering Challenge (CoQA) [39] aims to evaluate how well machines can comprehend a piece of written content and provide responses to a sequence of linked questions that are posed in the context of a conversation. It contains more than 127,000 questions with answers gathered from more than 8,000 dialogues. Question Answering in Context, often known as QuAC [40], is a dataset that encourages to model, comprehend, and take part in information seeking dialogue. In this dataset, instances take the form of an ongoing dialogue among two crowd workers, one of whom is a student who asks a series of freeform questions in an effort to acquire more knowledge about a concealed Wikipedia text, and the other of whom is a teacher who responds to the questions by providing brief snippets from the text.

Unlike other QA datasets in which the answers are extractive (can be obtained from a text span), TWEETQA [41] enables answers to be abstractive. Presented by the University of California and the International Business Machines Corporation (IBM), TWEETQA is a question-and-answer dataset with a primary focus on social media. It is the first large-scale dataset for quality assurance over social media data, and it was generated by researchers at both institutions. The dataset now contains 13,757 pairs of crowdsourced questions and answers in addition to 10,898 articles and 17,794 tweets.

In the search for more natural utterances, Natural Questions (NQ) [42] is a massively-scaled question answering collection that can be used to train and evaluate open-domain QA systems. This dataset, which was presented by Google, is the first of its kind since it replicates the whole process by which users obtain answers to their inquiries. It includes 300,000 naturally occurring queries together with human-annotated answers derived from Wikipedia pages, and it is intended for use in the process of training QA systems. In addition, the researchers included 16,000 instances where responses (to certain questions) are supplied by five distinct annotators. These examples will be helpful for evaluating the effectiveness of the trained QA systems since they provide a variety of perspectives on the same responses.

Chapter 4

Data

The data for this thesis is derived from three separate datasets, each of which is analyzed and discussed in this chapter. While the second and third datasets are utilized to conduct a qualitative analysis of the experiments, the first dataset is brought to use in the process of fine-tuning the model that is in charge of performing extractive QA.

With the goal of fine-tuning the QA model used in this project (BERT), the SQuAD dataset was employed. When working with the generative conversational model (DialoGPT), a combination of the MS MARCO and PERSONA-CHAT dataset was used. In the mixture created, there are a total of 27,782 instances of which 50% are PERSONA-CHAT type of question (with more natural and fluid language) wherein only the question is provided to the model, and the other 50% comes from MS MARCO where both the query and the passage containing the answer are given to the model. Details about the models that were used and with what purpose are presented in Chapter 5

4.1 Stanford Question Answering Dataset (SQuAD)

The Stanford Question Answering Dataset, also known as SQuAD [43], is a reading comprehension dataset that is comprised of more than 100,000 questions that were presented by crowdworkers on a collection of Wikipedia entries. The answer to each query is a section of text, also known as a span, from the correlating reading, or the question may not have an answer. This dataset was used to fine-tune BERT, which acts as the QA model in the experiments performed in this thesis.

4.1.1 Assemblage of the dataset

The information for the dataset was gathered in three stages: first, texts used as passages were curated, then questions about those sections were crowdsourced for replies, and lastly, more answers were obtained. When it comes to passage curation and in order to retrieve high-quality entries, the internal PageRanks of

Project Nayuki's Wikipedia were utilized in order to extract the top 10,000 publications of the English Wikipedia. From this pool of articles, the creators of the dataset sampled 536 articles in a manner that was homogeneously random. They took individual paragraphs from these articles and discarded any paragraphs that were less than 500 characters long. Images, figures, and tables were also removed from the texts. The end product was a total of 23,215 paragraphs spread among 536 pages spanning a broad variety of subject matter, ranging from well-known musicians to esoteric ideas. The articles were then arbitrarily divided into three groups: a training set consisting of eighty percent of the total, a development set consisting of ten percent, and a test set (with the other ten percent).

Next, the researchers used crowdworkers to generate questions as part of the stage where they collected responses to those questions. It was needed of crowdworkers that they have a HIT (completion of tasks) acceptance rate of at least 97%, that they had completed a least 1000 HITs, and that they reside in either the US or Canada. The workers were instructed to spend four minutes on each paragraph and were given the duty of posing and providing answers to as many as five questions related to the content of each paragraph. It was required that the questions be put into a text box, and that the answers be highlighted in the corresponding paragraph. In addition, those working the crowd were strongly urged to ask questions using their own language, rather than lifting phrases or words directly from the text (to ensure this the copy-paste feature on the passages was disabled). Finally, they acquired at least two more responses for every question in both the development set and the test set. This allowed them to gain a better sense of how humans perform on SQuAD and made their evaluation more reliable.

4.1.2 Analyzing the data

Understanding the qualities of a dataset is an essential component of any reliable collection of data. To achieve this, the authors looked at the following three areas:

Various classifications of replies. All responses were sorted into the corresponding group (e.g., verb sentence, place, numeric, noun sentence, numeric, date, among a few others). The responses are broken down as follows: 19.8 percent are comprised of dates and numbers, 32.6 percent are comprised of nouns, 31.8 percent are comprised of noun-phrases, and the remaining 15.8 percent are comprised of miscellaneous categories.

The use of reasoning is essential. The developers took a sampling of queries from the development set and then manually categorized them into the various types of reasoning that were required to solve them. For instance, the category known as "syntactic variation" indicates that the question has effectively been rephrased and that a reorganization of words is required to determine the correct response.

There is a discrepancy in the syntax. In order to determine how challenging a question is, the developers assessed the degree of syntactic divergence that existed between the question being asked and the phrase that contained the response.

They did this by developing a measure that counts the amount of changes that need to be made in order to turn a question into the phrase containing the answer. This helps to ensure that the dataset has a wide variety of syntactic structures.

4.1.3 Data format

Each instance of the dataset comes with the features presented in Table 4.1.

Table 4.1: SQuAD dataset format (fields)

Field	Description
ID	Unique identifier.
Title	Relevant title that relates with the passage and the question.
Context	Span of text (passage) that may or not contain the answer to the question.
Question	Inquiry for which an answer must be found.
Answers	Dictionary containing the answer and the character index of the start of the answer within the passage.

4.2 Microsoft Machine Reading Comprehension (MS MARCO)

The development of intelligent agents that are capable of open-domain question answering (QA) or machine reading comprehension (MRC) utilizing data taken from the actual world is an important objective of artificial intelligence research. The MS MARCO dataset [44] contains 1,010,916 questions that have been anonymized and were sampled from the search query logs of Bing. Each of these questions has an answer that was generated by a human, and the dataset also contains 182,669 answers that after being produced were completely rewritten by humans as well. In addition, the dataset includes 3,563,535 online documents that were obtained by Bing and contain a total of 8,841,823 passages. These passages give the information that is essential for constructing the natural language answer.

4.2.1 Advantages of the dataset

When analized and compared to other NLP and MRC datasets there are a few particularities that stand out. As an illustration, the discussion in CoQA is centered on text fragments originating from seven different domains. In addition, HotpotQA [45] possesses the following important characteristics: (1) finding and analyzing a variety of supporting papers is required to answer the questions; (2) there are

no preexisting information sources or knowledge structures that confine the inquiry.; (3) it provides sentence-level factual support for argumentation, enabling QA systems to deliberate with robust monitoring and justify their predictions.

In addition to this, CoQA and QuAC are members of the MRC in the context of dialogue (consecutive group of question-answer pairs that is is comparable to a conversation). And in contrast to students participating in the CoQA challenge, those in QuAC do not know the answers to their inquiries before they ask them. This means that string matching and straightforward paraphrasing play a less significant part in providing responses to the students' inquiries.

Overall, when compared with the other datasets found in the MRC task, MS MARCO highlights several advatanges:

- Actual questions. Each question is based on a sample taken from genuine Bing inquiries that have been anonymised.
- Real sources. The vast majority of the URLs that were used to source the sections include the whole online documents. These might be used to improve systems by providing additional contextual knowledge, or they could be utilized to engage in the authors' expert task.
- Answers that were generated by humans. Each question has a response that was composed by a human being (not only from a passage span). If the judge read any paragraphs that did not contain a response, they will write "No Answer Present" next to the appropriate text.
- Human generated well-formed answers. Certain inquiries present further human examination in order to provide well-formed responses that are suitable for use by intelligent agents (e.g., Siri, Google Assistant, Cortana, Alexa). This feature is present on the version 2 of the dataset.
- Size of the dataset. With over one million questions, it is considerable enough to train even the most sophisticated systems, and it also allows users to access the data for a variety of applications.

4.2.2 Generation of MS MARCO Q&A challenge

All of the questions that are utilized in MS MARCO are derived from actual, anonymised user queries submitted to Bing. This helps to ground the dataset in a real-world scenario and gives researchers access to real-world settings in which their models may be used. Using the most recent iteration of the Bing search engine, the context sections from which the responses in the dataset are obtained are taken from actual online documents and retrieved using the aforementioned search engine. The responses to the questions were created by a human being.

The MS MARCO collection is produced through a streamlined process that has been fine-tuned to provide the best possible quality instances. The steps that take place are as described below.

1. The Bing logs are randomly picked, filtered, and anonymized to ensure that the inquiries gathered are relevant to the research community while also being sensitive to Bing users.

2. Using the aggregated and concealed information from the sampled queries The top 10 most pertinent articles are generated by Bing based on the query.
3. Judges who have received extensive training examine the question and the texts that are connected to it (retrieved passages). If there is an answer, the texts that support it are marked, and a response written in natural language is produced.
4. A lesser subset of questions (about 17 percent of the whole dataset) are then forwarded on to a subsequent group of judges, who are tasked with determining whether or not the response is accurate and rewriting (if necessary) the inquiry for the answer to be well formed. These responses are intended to be comprehended even without the presence of a complete context, and they were developed with virtual assistants and voice assistants in mind.

4.2.3 Data format

To facilitate investigation, debugging, and loading, the data is released as a json file. Each instance of the dataset contains the fields described in Table 4.2.

4.3 PERSONA-CHAT

The PERSONA-CHAT collection [46] is a crowd-sourced set of data that was gathered using Amazon Mechanical Turk. In this dataset, each one of the pair of participants defines their discussion based on a predefined profile, which is given. The purpose of this research is to make casual conversation more interesting and on a more intimate level for speakers.

The dataset includes conversations with several turns that are determined by personas. It comprises a total of 968 full conversations for testing, 1000 dialogues for validation, and 8939 dialogues for training. In each interaction, two crowdsource employees assumed fictitious personas (designated by short biography description such as "I like to surf," "I am a soccer player," or "I love pancakes for breakfast") and engaged in a conversation. There are 955 training personas, 100 personas available for validation, and 100 testing personas. A rephrased, generalized, or specialized version of the original persona descriptions is also offered.

4.3.1 Collection of the dataset

The gathering of the data is broken up into three steps:

Personas. The authors solicited assistance from 1155 individuals through crowdsourcing to develop a character (persona) profile using five sentences. The goal was to develop biographies that are not only natural and descriptive but also include common themes of general interest that the participant is able to cite during the conversation. This was accomplished without the inclusion of any personally identifiable information. The workers were given the instruction to keep each statement as brief as possible, with 15 words at most allowed per sentence. This

Table 4.2: MS MARCO dataset format (fields)

Field	Description
Query ID	Each question has its own unique identifier, which is utilized during assessment.
Query	A unique inquiry based on Bing use history.
Passages	On average, a set of 10 passages for each question that may or not include the answer to the question is provided. These extracts were taken from related online pages by Bing's cutting-edge passage retrieval algorithm. The editors are to annotate the flag 'is selected' as 1 or 0 for each passage depending on whether they utilized the text to build the final response.
Query type	Each inquiry is additionally automatically tagged with one of the five segmented labels employing a trained classifier: NUMERIC, ENTITY, LOCATION, PERSON, or DESCRIPTION.
Answers	The majority of the replies given by human judges consist of a single response, whereas less than one percent comprise several responses. These responses were created by actual individuals in their own words, as opposed to being pulled from a predetermined text. Similar or identical terminology may be used in their response to any of the passages. The editors were encouraged to read and comprehend the questions, examine the recovered sections, and then generate a natural language response with the proper information collected just from the offered passages.
Well formed answers	For certain question-answer pairs, the dataset additionally includes one or more responses created by a second judge and rewriting procedure. This procedure includes a different editor reading and revising the submitted response (and re-writing it if necessary). This field was not used as it is primarily meant for the Natural Language Generation (NLGEN) task.

is beneficial for both humans and robots since, if they are too extensive, crowdsourcing employees are prone to losing interest in the activity, the task might become harder for machines.

Personas with revisions. One of the problems associated with conditioning on literary personas is that there is a risk that individuals, even when explicitly instructed not to do so, may unintentionally duplicate profile details either identically or with considerable word overlap . This is the case even if they are told not to. Further machine learning problems may get simpler (not challengin), and the solution methods may not be applicable to increasingly demanding tasks.

The initial original personas were shown to a new group of crowdworkers, and they were asked to reinterpret the phrases so that the new description becomes includes related attributes that the same person may have. Because the revisions could be phrases with different words but same general meaning, generalizations, or specifications, this problem was solved by presenting the original personas to the a new set of crowdworkers.

Persona chat. It was missing to gather the conversations themselves, based on the personalities that were collected. For each conversation, two crowdworkers were selected at random and paired together. They were then given the directive to strike up a conversation with another worker while assuming the role of a particular persona that had been selected for them from the collection of personas that had been gathered earlier. Additional rules included limiting each message to no more than 15 words and assuming responsibility for both posing questions to and providing responses to those made by the chat partner.

4.3.2 Data format

The dataset is presented as Table 4.3 illustrates.

Table 4.3: PERSONA-CHAT dataset format (fields)

Field	Description
Conversation ID	Unique identifier for each dialogue.
Utterance index	The index corresponding to each turn between the speakers.
Profile	Sentences that form the description in the speaker's profile.
History	The utterances that have occurred so far in the conversation (alternating turns between speakers).
Candidate replies	List of distracting utterances that may be chosen as response with the ground truth reply at the end.

Chapter 5

Methods

This chapter provides a high-level summary of the procedures that were utilized to carry out the experiments and the analysis for this study. The chapter is broken up into two primary parts or sections. The first presents an overview of the tools that were used for the implementation, while the second aims to answer the third research question on incorporating knowledge into later agents by combining them with QA models. The drawbacks of transformer-based generative conversational agents (third research question) are addressed in Chapter 6.

5.1 Libraries

During the experiments, Python has been the programming language used due to its versatility when it comes to object-oriented code and all the resources it can be integrated with for AI projects. The experiments were developed using a conda[1] environment with the libraries highlighted in Table 5.1

[1]https://docs.conda.io/en/latest/
[2]https://huggingface.co/docs/datasets/index
[3]https://www.nltk.org/
[4]https://numpy.org/
[5]https://pandas.pydata.org/
[6]https://pytorch.org/
[7]https://regexr.com/
[8]https://scikit-learn.org/stable/
[9]https://scipy.org/
[10]https://huggingface.co/docs/tokenizers/index
[11]https://huggingface.co/docs/transformers/index
[12]https://huggingface.co/docs/accelerate/index

Table 5.1: Libraries used in the experiments

Library	Version	Description
Huggingface Datasets[2]	1.13.3	The library provides assessment measures for NLP tasks, computer vision, and audio activities, as well as easy access to datasets.
Nltk[3]	3.7	It includes packages for categorization, tokenization, lemmatization, tagging, parsing, and semantic reasoning that are simple to use.
Numpy[4]	1.21.5	The numpy provides fast and versatile vectorization as well as vector operation. It simplifies working with data structures and offers efficient mathematical operators.
Pandas[5]	1.4.1	Pandas is a library that is widely used because its advantages in data analysis that is quick, robust, customizable, and simple to use.
Pytorch[6]	1.10.2	Pytorch is a widely known deep learning library that simplifies the use GPU for tensor operations.
Regex[7]	2021.8.3	Regex is a package that lets you define patterns to help you match, find, and handle text. It is highly useful in text processing stages.
Scikit-learn[8]	0.23.2	Scikit-learn is a powerful library that offers performant machine learning algorithms that may be used during training or evaluation when experimenting with data..
Scipy[9]	1.7.3	Library dedicated to mathematical operations and data analysis.
Huggingface Tokenizers[10]	0.10.3	Library that is in charge of converting text to data that can be processed by machines. It represents one of the main elements of every NLP pipeline.
Huggingface Transformers[11]	4.11.3	The Huggingface Transformers library provides several application programming interfaces (API) to easily download and implement pre-trained models.
Huggingface Accelerate[12]	0.5.1	This library provides an API for easy use of distributed computation (GPU, TPU, etc.) when writing custom training loops with mixed precision.

5.2 Combining QA models and transformer-based generative conversational agents

5.2.1 Building up the QA model - BERT

BERT (in its base-cased version) has been chosen as the QA model to be used to answer factoid questions (e.g., What is the capital of Norway?). Because BERT was pre-trained with the objectives of Masked Language Modelling (MLM) and Next Sentence Prediction (NSP), an internal representation of the language (english in this case) is learned by the model, and this representation may be used to extract characteristics helpful for subsequent tasks. Then, the model was fine-tuned for the specific task of question answering using the SQuAD dataset.

Preparing the data

In order to begin the process of running these experiments, the text data had to first be converted into the format specified by the model for the purposes of extracting features and fine-tuning. Huggingface Transformer models implies the input to be structured in a very particular manner, and as a result, they provide tokenizers that are unique to each model. These tokenizers accept as input either a sequence of text or a batch of these and construct the input IDs needed by the model. The first thing that has to be done in order to commence this procedure is to load the dataset into an object of type Huggingface Datasets. By passing both the question and the context to the tokenizer, the later combines both sequences and insert special tokens to produce an output of the form [CLS] question [SEP] context [SEP].

Every model has a maximum sequence length, which defines the maximum amount of data the model can take in and analyse at once. This constraint is important to consider when dealing with long contexts that might get truncated if no correction is applied. When sequences are excessively long, they are shortened (by being truncated), and when their lengths vary, padding is practiced to make them all the same length. The tokenizer additionally generates a binary attention mask, which is formed by zeroes in the padding values and ones for the actual input. This is done to ensure that padding tokens are not considered into the attention computation. The tokenizer produces a list including positional information which is referred to as an overflow to sample mapping. If an input sequence is too extensive, every feature vector that corresponds to that sequence or input will have the same value in the overflow to sample mapping list (for example 0 if they all come from the context with index 0 in the original dataset before being splitted).

The pre-trained bert-cased model that is being utilized for this project can manage contexts that are no more than 384 tokens long (words). The creation of many training features from a single instance of the dataset, with a sliding window of 128 tokens in between each of them, is the method that is being used to

handle lengthy contexts (i.e., passages). As a result of this, when the vectors have been tokenized an overlapping may occur between them. In a similar manner, it generates certain training cases in which the response is either omitted from the context altogether or, alternatively, where the answer has merely been truncated (meaning that only the beginning or ending of the answer is in that feature vector).

Following an analog procedure when it comes to training objective, just like this model was pre-trained with the MLM and NSP goals, in this case we need to add some extra fields to the data before we can train the model. This new information is the starting and ending index of the answer within the context, so that the output of each feature vector is the probability of every token being the start or the end of the answer. In the cases mentioned above in which the answer gets trucated or is not in the passage (as a result of splitting long passages into several feature vectors with overlapping tokens), both of these labels (starting and ending) are set to zero.

It is feasible to determine the end character in the context by adding the length of the response to the information that the SQuAD dataset already offers regarding the start character of the answer in the context. After that, it is necessary to determine if the portion of the context in a particular feature vector begins after the response or comes to an end before the answer does. If this is the case, the label is (0, 0).

Fine-tuning the model

There are several pre-trained language models available today. Scientists can utilize these models once they've been trained on massive quantities of data by the authors who created them. The amount of time and processing power required to pre-train these models is extremely vast for anybody other than major corporations and research institutes; however, fine-tuning these models on a smaller dataset is a task that is far more doable. Either the models may be used straight as they are or they can be fine-tuned so that they can adjust to the domain-specific knowledge and job that they are going to be used for.

It's possible that the metric computation function was the most challenging aspect of the tuning to get right. The post-processing of the predicted results into extracts of text in the original samples is the portion that is the most difficult. As a result of the fact that the model produces logits for the starting and ending locality of the answer in the input IDs vector, it is essential to mask away the later logits that correspond to tokens that are not relevant to the context (outside the context). Because just the answer that is predicted has to be computed, the step of transforming the start and end logits into probabilities by using a softmax layer has been omitted. This is because it is not necessary to compute real scores. Furthermore , in order to speed up the process, not all of the (start token, end token) sets are scored; rather, only the ones that match up to the highest 20 scores are considered. Positions that produce (i) an answer that is not within the context,

(ii) a response that has negative length, or (iii) a response that is too long (maximum 30 words) are disqualified from consideration. Since the softmax is being skipped, the scores that are being collected are logit scores. These logit scores are then processed by calculating the sum of the start and end logits (applying $\log(ab) = \log(a) + \log(b)$).

During the training phase, Huggingface Accelerate and the default data collator that is supplied by Transformers were utilized. Additionally, the training set underwent additional shuffling. Half precision, often known as FP16, was used to cut down on the amount of time needed for training, and the custom loop is made up of three core parts. The training procedure begins with the forward pass, followed by the backward pass, and finally the optimizer step. The optimizer that was utilized was AdamW, which offered a better weight decay in comparison to Adam, and a learning rate of 2e-5 was used as it was suggested by the BERT authors that this would perform well with the majority of individual tasks throughout the process of fine-tuning [26]. The second stage is the evaluation, during which all of the values for start logits and end logits are collected and then turn them into NumPy arrays. After this step, we go on to the third stage. After the evaluation has been completed, all of the results are appended together. Note that truncation is necessary to be performed since Accelerator may have introduced several samples at the end in order to guarantee that we have the same amount of instances in each process. At last, the weights as well as the tokenizer are both stored and then uploaded to the Huggingface Hub so that they are available to be accessed at a later time. This is accomplished in an asynchronous manner to guarantee that regular training proceeds while these instructions are carried out in the background. This training process took place for three epochs.

Evaluating the fine-tuned model

The performance of the fine-tuned model and most of the work of this thesis is measured by using the F1 score metric. Specifically, the already built-in F1 computation from Huggingface Datasets adopted by the authors of the SQuAD paper [43] was used on the validation set during training (at the end of each epoch). Both the prediction and the ground truth answer are treated as bags of tokens and their F1 values are computed. The maximum F1 score over all ground truth responses for a particular question is taken, and then average it across all questions.

5.2.2 Setting up the generative conversational model - DialoGPT

Nowadays, there are plenty of conversational models trained on a large amount of data that is extracted from news resources, social media, discussion forums, and other source of interactions where humans can dialogue and exchange utterances. An already trained state-of-the art model such as DialoGPT has been employed to act as the transformer-based generative conversational agent. No fine-tuning

on the model was carried out as it has already learned linguistic and language features from millions of conversations posted on Reddit.

To make inference with this model, it is necessary to load both the tokenizer and the model itself (weights, hyperparameters, etc.) from the Huggingface Transformers library. One can easily test it out by looping the times of speeches desired to happen in the conversation, where the the input is encoded with the tokenizer and the EOS special token is appended at then end before getting a tensor in Pytorch. The EOS token is used by decoders to explicitly indicate the end of a sentence or sequence. Afterwards, the resulting tokens may be added to the dialogue history and a response is generated, which needs to be decoded using the same tokenizer.

5.2.3 Bringing in a query classifier

It is a common issue that users may experience deficiency when chatting with a chatbot that has pre-defined rules on how to answer (e.g., social media conversational robots, the chat assistant in most of bank websites). In an equivalent way, conversational agents that are developed to maintain a chit-chat on a human-like manner with users often expose certain shortage depending on the type of question the user makes. In order to incorporate knowledge into the transformer-based agent and combine both models, a query classifier is introduced in the middle of both models so that it can, depending on the intention and type of question, forward the inquiry to the appropriate model that will most likely answer best. Three different algorithms have been evaluated: Naive Bayes, Support Vector Machine, and Random Forest.

Training the classifiers

Naive Bayes, often known as NB, is a classification method that is founded on Bayes' Theorem and operates on the premise that all of the variables that predict the target value are unrelated of one another. After determining the probabilities associated with each class, it selects the category that has the best overall odds. It has proven effective for a wide range of applications, but NLP challenges stand out as notably suitable to its use. A Support Vector Machine (SVM), on the other hand, is a discriminative classifier that is essentially characterized by a separating hyperplane. As a result of using supervised learning (labeled training data), the method generates an optimum hyperplane that correctly classifies incoming samples. The term "hyperplane" refers to a line in a two-dimensional space that divides a plane into two sections, one for each category.

The SVM method was developed in such a manner that it searches the graph for points that are placed closest to the borderline and those that are directly next to it. The names given to these points are "support vectors." The next step in the process involves the algorithm determining the distance that separates the reference vectors and the splitting plane, which is referred to as the gap. The primary

objective of the algorithm is to achieve the greatest possible increase in the clearing distance. Hence, the ideal hyperplane is one in which the gap is maximized to the greatest extent feasible.

A Random Forest (RF) classifier provided by Sklearn is the subject of the investigation for the third technique. The Random Forest algorithm is a form of supervised ML algorithm that is mostly reliant on ensemble learning, which is the type of l earning where numerous instances of the same algorithm or other kinds of algorithms are combined to produce a more robust prediction model. It is called Random Forest as it combines several decision trees. Both regression and categorization problems are amenable to the use of the random forest technique.

All three models have been trained using the PERSONA-CHAT and MS MARCO mixture as they provide both factoid questions in which extractive QA is required, and open domain question with a more natural language (abstractive QA). The data must first be preprocessed, which is an essential step in any data analysis. Essentially, this entails converting raw data into a format that can be understood by NLP models, as data collected from real world scenarios sometimes lack specific tendencies, shows inconsistency with one another, or is incomplete. It has been demonstrated that pre-processing of data can address problems of this kind. Because of this, classifying algorithms will produce more accurate outcomes.

Several techniques as tokenization and lemmatization have been applied to the original data. The former involves separating a string of text into individual tokens that may be words, sentences, symbols, or other significant components, and that as a list of tokens it becomes available as input for processing that comes next . Handy functions as *word_tokenize* and *sent_tokenize* are found in the NLTK library to make it simple to parse a string of text into a list of individual words or sentences. The later (lemmatization), which is closely connected to the process of stemming, attempts to reduce all of a word's derivational forms to a single root. The distinction is that a stemmer works on a single word without knowing the context, and so cannot distinguish between words having varying meanings relying on parts of speech (i.e., verb, noun, adjective, etc.). Stemmers, as a counterpart, are often simpler to design and perform faster, and the decreased accuracy might not matter as much for some tasks.

Besides tokenization and lemmatization, the data has been also cleaned by removing blank rows, imposing lower case in all the questions, tokenizing the words, removing stop words such as "the", "is", "and", and removing text with non-alphanumeric characters. A label encoder was also utilized to encode the target data which in this case is the type of question (personachat and non-personachat).

TF-IDF has been utilized for the process of word vectorization, which is also known as words embeddings. This is a broad term that refers to the transformation of a corpus of textual data into numeric feature vectors. The TF-IDF model that is offered by the library Sklearn is fit on the entire corpus with a parameter of maximum 5000 unique words or features. Once the TF-IDF model has been trained and features have been extracted, the training set and test set are transformed to be vectorized using the recently trained TF-IDF. This ensures that tokens are

expressed as unique numbers that are representative of the significance that is associated with such a token.

Running the experiments

Once the QA model (BERT) is fune-tuned on the question answering task, the three query classifiers have been trained and the generative model (DialoGPT) is up and running, several test are run to evaluate the effectiveness of combining the two main models by introducing a broker that classifies the type of question. With this in mind, both the QA and generative model on their own (without any query classifier) were tested on the entire mixture of PERSONA-CHAT and MS MARCO so that performance could be measured in comparison with when these two models operate together with a classifier in between. More details about every experiment and the results obtained may be found in Chapter 6.

Chapter 6

Results and discussion

This chapter contains the findings obtained from the experiments and examinations that were carried out as part of this thesis, as outlined in chapter 5. The structure of this chapter is quite similar to that of the preceding chapter; it is divided into two primary parts, the first of which focuses on answering the second research question on the drawbacks of generative transformer-based conversational agents, while analyzing the results of the fine-tuned version of BERT and DialoGPT separately. The second part provides the findings obtained when combining the mentioned two models by introducing the query classifier, thus contributing to what was highlighted in the previous chapter about the third research question. Each of these sections is further broken down into subtopics for each trial that detail the results that were achieved, as well as a discussion of the experiments and references to pertinent prior research.

6.1 Evaluation of the QA model (fine-tuned BERT)

Training the model for it to be fine-tuned was done through 4 epochs while getting the best F1 score on the validation set in the third epoch. Figure 6.1 presents the scores obtained across training.

Table 6.1: F1 scores on SQuAD - Fine-tuned BERT vs. results presented in BERT's article

Fine-tuned BERT	**BERT paper**
88.5	88.5

Table 6.1 reports that the F1 score achieved by the fine-tuned BERT model reaches the same score as from the one in the original paper. An interesting fact regarding these results is that the fine-tuned model's performance improved slightly when deploying a more powerful hardware to be trained on. At first, the model was fine-tuned using a personal computer with a relatively low computing ca-

pacity compared to the hardware provided by the IDUN cluster [47] from High Performance Computing Group at the Norwegian University of Science and Technology (NTNU). Some of the distinctions between the two hardware architectures are shown in Table 6.2.

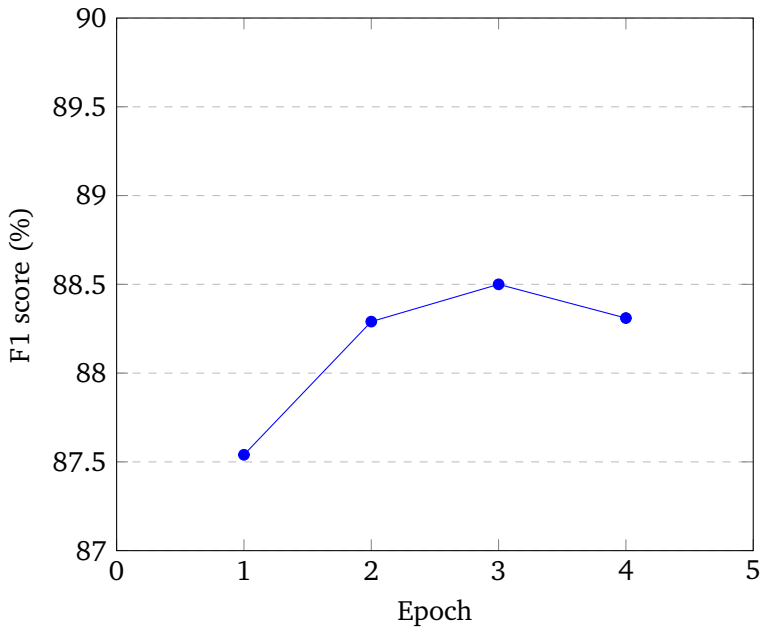


Figure 6.1: Fine-tuned BERT - F1 score on validation set across epochs during training

Differences in processors and their Instructions Set Architecture (ISA) may yield in smaall floating point rounding errors that can get added through training. An ISA is a component of a computer's abstract model that specifies the manner in which software exerts control over the central processing unit (CPU). Both what the processor is competent of doing and how it is accomplished are specified by the Instruction Set Architecture, which functions as a bridge between the hardware and the software. Additionally, training on single or multi-GPU may provoke slight discrepancies as well as the batch size [48], which varied across the two computer architectures due to different memory capacity. Altogether, the fine-tuned version of BERT seems to perform fine on the SQuAD validation set.

Once BERT was fine-tuned on the QA task, it was used to make inference over the whole dataset (mixture of MS MARCO and PERSONA-CHAT) so that later comparison with the generative model and the combination of both can be established. The model reported an F1 score of 15.86% over the 27,782 examples. Table 6.3 also reflects the score of 31.03& obtained when testing the model only on the portion of the custom dataset that contains non-personachat questions (i.e., instances from MS MARCO).

Upon obtaining 88.5% on the SQuAD validation set, one could think that the

Table 6.2: F1 scores on SQuAD of fine-tuned BERT - Personal PC vs. IDUN cluster

	Processor	**RAM**	**GPU**	**Rack Server**	**F1 score**
Personal computer	Intel Core i7-10750H	16GB	NVIDA GeForce RTX 2060	None	88.04
IDUN cluster	Intel Xeon E5-2650V4	128GB	NVIDIA Tesla P100	Dell PowerEdge R730	88.50

Table 6.3: F1 scores on MS MARCO and PERSONA-CHAT of fine-tuned BERT

MS MARCO	**MS MARCO + PERSONA-CHAT**
31.03%	15.86%

31.03% on part of MS MARCO is relatively low. One possible approach to explain this may have to do with the way MS MARCO was created and its bias. By recalling some of the notes described in Chapter 4 and presented in [44], it is possible to see that for each question that users made to Bing search engine, relevant documents were retrieved using its large web index, automatically selecting pertinent spans of texts (passages) from those documents, followed by human judges marking if these passage include applicable information to answer the question in hand; just these three steps could result in adding certain bias to the dataset, which is something exposed in applicable literature.

The authors in [49] refer to this dataset presenting what is known as survivorship bias. Survivorship bias alludes to the propensity to focus on the favorable outcomes of a screening process and miss the findings that cause bad outcomes. While some of the questions were not suitable (either because they would fall out of the set of requirements or simply because they were unsolvable), around 40% of the samples were discarded due to the editors not being able to find appropriate answers to those questions, something that could have been avoided if the corpus had been put together using more modern document ranking methods.

6.2 Evaluation of the generative conversational model (DialoGPT)

The DialoGPT model that was loaded from Huggingface Transformers library performed somewhat poorly. Table 6.4 shows the F1 score obtained when testing out the model on the entire custom dataset (PERSONA-CHAT + MS MARCO), as well on only the type of questions it is expected to perform the best (PERSONA-CHAT questions).

The end task of developing a model able to engage with a human in a conversation in a smooth and natural way as humans would do with each other is still an important challenge within the NLP community. Dialogue between a person and a

Table 6.4: F1 scores on MS MARCO and PERSONA-CHAT of DialoGPT

PERSONA-CHAT	MS MARCO + PERSONA-CHAT
12.20%	11.08%

computer is still in its early phases, despite the significant progress that has been made in NLP and conversation research in recent years. It was not until not long time ago that models have adequate capability and access to enough vast datasets to give the impression that they can provide meaningful replies in a situation similar to a casual conversation. However, engaging in even a brief conversation with such conversational models can rapidly reveals their shortcomings.

Generative conversational models frequently present drawbacks such as the following: an absence of a coherent personality since they are usually trained over several dialogues, each of which has a different speaker; a dearth of a long-term memory because they are generally trained to generate speeches given just the latest conversation history; and a propensity to yield non-specific responses such as "I do not know" (instead of expanding the answer with specific information expressed on the question). A combination of already these three issues results in an general experience that is unpleasant for a person to have while interacting with it.

Natural conversation (also known as chit-chat) is frequently disregarded as a potential end-application not only because the existing conversational agents are of poor quality but also because it is challenging to evaluate these models. Rather, the scientific community has been emphasizing on task-oriented interaction, such as making reservations at a restaurant, booking a flight, or even single-turn response searching (question answering) [50].

The challenge of comprehending the natural language, sometimes referred to as the Natural Language Understanding (NLU) task, is, as of the current day, the single most important factor in determining whether or not the text can be further comprehended and processed. Figuring out the definition of a word or the sense in which a word is used, assessing the possibilities of quantifiers, locating the connotations of anaphoras, and establishing the relevance the certain modifiers may have on nouns are some of the issues that have not yet been resolved.

It is challenging to both represent and deduce global information, and more specifically general knowledge. A single sentence can be used to enlighten, to confuse in regards to a fact or the belief that the speaker can have about it, to seek attention, to demand something, to remind something, and so on. This is a difficulty for pragmatics. It would appear that the practical interpretation is open-ended, which makes it challenging for systems to understand.

Another important aspect to be considered is the emotions and intention of human utterances. It is possible that two speakers will employ a different style of communication to describe the same topic depending on who they are as people, what they're trying to accomplish, and how they are feeling; for example, with

irony or sarcasm a speaker may communicate a message that is diametrically opposed to the one that is literally intended. It is fairly obvious to state that a significant portion of human conversation is focused on socializing, individual interests, and idle chitchat. For instance, fewer than five percent of postings on Twitter are queries, but over eighty percent of tweets are about the authors' own private emotional states or opinions [51]. Although there has been significant development in the field of sentiment analysis in the past few years, there is still more work to be done in order to correctly interpret the pragmatic competence of the text.

Last but not least, because natural language processing is a data-driven endeavor, there are also data-related issues that impact its performance. When considering the sort of data and the quantity of it that is truly best to collect, the answer is not as straight forward as one could think, as the usefulness of NLP tools is sometimes hindered by the presence of data that is insufficient, uneven, or even too diverse. In some cases , gathering additional data is either not possible (such as acquiring more resources for limited languages) or would result in increased unpredictability. In order to correctly describe a task even if the essential data is available, one is required to construct datasets and establish evaluation techniques that are adequate to monitor progress toward defined goals. Additionally, if we are to posses large datasets with a vast amount of documents, then supervision becomes increasingly difficult and costly to acquire. (i.e., collecting enoguh editors or crowdsource workers, provide training, avoid the introduction of bias into the data, etc.).

It is a well-known problem that, while there is a wealth of information available for widely spoken languages (e.g., Chinese or English), there are many more languages being spoken by a very small population and, as a result, receive a much smaller amount of focus and attention. For instance, just in Africa there are around 2,000 different languages, yet there is very little information available about them. Furthermore, the transfer of tasks requiring true natural language comprehension from high-resource dialects to low-resource ones remains extremely difficult.

6.3 Combining fine-tuned BERT and DialoGPT with a query classifier

6.3.1 Evaluating the question classifiers

Table 6.5 reflects the accuracy achieved by the three classifiers on the test set (30% of the entire mixture between PERSONA-CHAT and MS MARCO) after they were trained. While Naive Bayes performs, as it names suggests, naive operations assuming that each component is independent of the others given that the evidence is taken into account, it performs admittedly well when it comes with small spans of text. However, what appears to be a benefit can also play as a disadvantage for the algorithm since, in the vast majority of instances, the assumption does not keep its relevance. The assumption of class-condition independence, which is very unlikely to hold true in the majority of applications that are used in the real world,

means that it is unfeasible for the system to learn how features interact with one another, leading to an overall reduction in accuracy compared with the other two approaches.

Table 6.5: Accuracy of the three classifiers after training: Navie Bayes (NB) vs. Support Vector Machine (SVM) vs. Random Forest (RF)

NB	SVM	RF
89.75%	92.75%	95.55%

Support Vector Machine and Random Forest seem to perform the best. Since the data has fairly few irrelevant features because of the pre-processing steps that were taken with it, and because SVM has the capacity to plug a variety of kernels (including a linear kernel), it is able to effectively operate on such predictors and filter out the independent term to greatest distance, something which NB fails to achieve based on the independence assumption. The versatility of Random Forest makes it achieve the greatest accuracy,

6.3.2 Evaluation of the association between fine-tuned BERT and DialoGPT using a query classifier

Contrary to the results of the classifiers on their own wherein RF reached the highest accuracy after being trained, it is not the same case when using the classifiers to combine BERT and DialoGPT to answer the questions. When looking at Table 6.6 it can be seen that RF performs slightly worse than SVM (20.69% against 20.79%, respectively).

Table 6.6: F1 score achieved on the entire dataset when combining the fine-tuned BERT model with DialoGPT by introducing the query classifiers NB, SVM, and RF

NB	SVM	RF
19.39%	20.79%	20.69%

Table 6.7 illustrates a summary of the F1 scores obtained from both models separately (fine-tuned BERT and DialoGPT) and the score attained when combining the two models with the query classifiers. A noticeable improvement can be observed when comparing the Hybrid approach with the models being deployed entirely for the whole corpus (representing a baseline). The accuracy obtained during inference by the three classifiers is 89%, 93.94%, and 93.56%, respectively.

While the results suggest that knowledge may be incorporated into generative conversational agents through a query classifier that forwards the inquiries to the appropriate model, it can also be said that the conversational model (DialoGPT) is acting as a bottleneck in this case. If one takes the average F1 score of the the

best performance of both models (i.e., fine-tuned BERT making inference only on the MS MARCO portion of the dataset and DialoGPT only on the PERSONA-CHAT split), which is 21.62%, it is seen that it approximates to what is obtained from the combination of both models (20.79% using SVM).

Table 6.7: Summary of F1 score obtained from the fine-tuned BERT model and DialoGPT separately, as well as when combined with the query classifiers NB, SVM, and RF

	MS MARCO	**PERSONA-CHAT**	**MS MARCO + PERSONA-CHAT**		
BERT	31.03%	—	15.86%		
DialoGPT	—	12.20%	11.08%		
Hybrid	—	—	NB	SVM	RF
			19.39%	20.79%	20.69%

Chapter 7

Conclusion

This section brings to an end the work that has been accomplished throughout this thesis by providing some concluding thoughts on the research questions and more work that is required to expand the research on transformer-based generative conversational agents and combining such agents with question answering models.

7.1 Discussion of the research questions

RQ1 *What is the state-of-the-art in QA models and generative models?*

This thesis has presented several state-of-the art QA models developed by Google and Meta such as BERT, RoBERTa and ELECTRA. While being of the first models in incorporating attentions masks, BERT brings closer the research to the question answering tasks because of the way it has been trained. Next Sentence Prediction allows the model to gain understanding on the relation between two phrases, thus acquiring valuable knowledge of the language in hand.

Meta's work, RoBERTa, gives a twist to what BERT was proposing by changing the input format of NSP. Removing the NSP loss from the original <segment-pair+NSP loss> that the original paper suggested yielded in an improvement on the SQuAD dataset. The best results were obtained when the authors removed the NSP loss and by taking complete phrases from single documents (rather than grouping sentences from multiple documents). The results reported were in turn comparable to the ones from ELECTRA, which uses less than 25% of the computing resources employed by RoBERTa. The keys ELECTRA's performance lies in the introduction of two neural networks encoders such as a generator and a discriminator.

On a different note, great advances have also been done within the generative conversational agents field. GPT-3 and DialoGPT are just some of the several models that have shown brilliant performance. Researchers think of GPT-3 as a model that can undertake reading comprehension and writing tasks at a relatively close

human level, with the exception that it has observed more text than what any individual could ever see in their entire life. This is one of the main reasons why GPT-3 is so effective.

Microsoft's attempt to retain a point in which the model is more conversationally engaged than ever is reflected through DialoGPT. It achieved state-of-the art results in generating conversational answers that are more meaningful that a small talk by incorporating external knowledge in its replies.

RQ2 *What are the drawbacks of transformer-based generative conversational agents and how these can be addressed by combining such agents with QA models?*

This thesis has shown some of the downsides that transformer-based generative conversational agents posses. These include a lack of a long-term memory because they are typically trained to generate speeches using only the most recent conversation history, a tendency to produce non-specific responses like I do not know, and an absence of a consistent personality as they are typically trained across several interactions with each offering a distinct speaker. Efforts within the field are being put in combining these agents with QA models by integrating a knowledge base within the system with the aim of creating more natural responses.

RQ3 *How can knowledge be incorporated into transformer-based generative models?*

The work presented in this theses shows how the performance of transformer-based models (in this case, DialoGPT) can be enhanced by introducing a query classifier that forwards the questions to either the conversational agent or the questions answering model at hand (in this case, a fine-tuned of BERT). For this purpose, three different classifiers were test out during the experiments: Naive Bayes, Support Vector Machine, and Random Forest.

When using a custom dataset comprised by a mixture of MS MARCO and PERSONA-CHAT (factoid questions and natural conversational questions), results illustrated that the overall F1 score obtained increases from 15.86% (when only employing the fine-tuned BERT to answer all inquiries) to 20.79% that is achieved when introducing Support Vector Machine a the query classifier.

7.2 Further work

The results that are displayed in this thesis are encouraging, and there are multiple areas that can be explored for advancements and expansions that will support even more the efficiency of transformer-based generative conversational agents and their capacity to produce more meaningful answers, as well as the association

with QA models for the purpose of achieving this objective. A few of these are highlighted in this section.

7.2.1 Test other language models

Because the outcomes predicted by the models differ so widely, selecting a reliable model is of the utmost importance. More research has to be done to investigate the efficacy of more models and go further into which components of those models have the most significant influence on the outcomes. This thesis makes use of BERT but research has shown that, for instance, RoBERTa and ELECTRA can be perfect model candidates to try with. A few of the elements that are of concern include the learning goal, the pre-training dataset, and the architectural type. In regards to the generative conversational agent, both GPT-2 and GPT-3 may provide different results that expand the discoverments on combining these agents with QA models.

7.2.2 Apply different metrics

The level of success achieved on the job of providing answers to questions might vary substantially from one model to another, and the same is applicable for the metrics associated with evaluating those. Determining which tasks and datasets a particular measure is most appropriate for will be made easier by having a solid knowledge of the characteristics of what a good response is constituted of. Alternative measures like the gold response, the Recall Oriented Understudy for Gisting Evaluation (ROUGE), and the Bilingual Evaluation Understudy (BLEU) are some of the metrics that might be investigated.

7.2.3 Expand to other languages

When the scope of the analysis is broadened to include more languages, there is the potential to gain a significant amount of knowledge on the many ways in which QA modes might be combined with generative transformer-based conversational agents. Numerous language models are being trained on languages different than English, and models able to handle multiple languages are continually improving in terms of their overall performance. When it comes to these languages, the availability of question-answering datasets will open up a substantial new area for further research.

Bibliography

[1] J. Li, M. Galley, C. Brockett, J. Gao and B. Dolan, 'A diversity-promoting objective function for neural conversation models,' in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 110–119. DOI: 10.18653/v1/N16-1014. [Online]. Available: https://aclanthology.org/N16-1014.

[2] J. Li, M. Galley, C. Brockett, J. Gao and B. Dolan, 'A persona-based neural conversation model,' *CoRR*, vol. abs/1603.06155, 2016.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, 'Language models are unsupervised multitask learners,' 2019.

[4] Y. Ostapov, 'Question answering in a natural language understanding system based on object-oriented semantics,' *arXiv preprint arXiv:1111.4343*, 2011.

[5] Z. Ji, F. Xu, B. Wang and B. He, 'Question-answer topic model for question retrieval in community question answering,' in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 2471–2474.

[6] J. M. Prager, 'Open-domain question-answering.,' *Found. Trends Inf. Retr.*, vol. 1, no. 2, pp. 91–231, 2006.

[7] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st. USA: Prentice Hall PTR, 2000, ISBN: 0130950696.

[8] J. Guo, Y. Fan, Q. Ai and W. B. Croft, 'A deep relevance matching model for ad-hoc retrieval,' in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '16, Indianapolis, Indiana, USA: Association for Computing Machinery, 2016, pp. 55–64, ISBN: 9781450340731. DOI: 10.1145/2983323.2983769. [Online]. Available: https://doi.org/10.1145/2983323.2983769.

[9] M. Buckland and F. Gey, 'The relationship between recall and precision,' *Journal of the American society for information science*, vol. 45, no. 1, pp. 12–19, 1994.

[10] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li and J. Lin, 'End-to-end open-domain question answering with bertserini,' *arXiv preprint arXiv:1902.01718*, 2019.

[11] S. Ramnath, P. Nema, D. Sahni and M. M. Khapra, 'Towards interpreting bert for reading comprehension based qa,' *arXiv preprint arXiv:2010.08983*, 2020.

[12] E. M. Voorhees *et al.*, 'The trec-8 question answering track report,' in *Trec*, Citeseer, vol. 99, 1999, pp. 77–82.

[13] W. Shen, J. Wang and J. Han, 'Entity linking with a knowledge base: Issues, techniques, and solutions,' *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 443–460, 2015. DOI: 10.1109/TKDE.2014.2327028.

[14] X. Han and L. Sun, 'A generative entity-mention model for linking entities with knowledge base,' in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11, Portland, Oregon: Association for Computational Linguistics, 2011, pp. 945–954, ISBN: 9781932432879.

[15] R. J. Mooney, 'Learning for semantic parsing,' in *Computational Linguistics and Intelligent Text Processing*, A. Gelbukh, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 311–324, ISBN: 978-3-540-70939-8.

[16] R. C. Staudemeyer and E. R. Morris, *Understanding lstm – a tutorial into long short-term memory recurrent neural networks*, 2019. arXiv: 1909.09586 [cs.NE].

[17] Y. Bengio, P. Simard and P. Frasconi, 'Learning long-term dependencies with gradient descent is difficult,' *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994. DOI: 10.1109/72.279181.

[18] S. Hochreiter and J. Schmidhuber, 'Long short-term memory,' *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[19] I. Sutskever, O. Vinyals and Q. V. Le, *Sequence to sequence learning with neural networks*, 2014. DOI: 10.48550/ARXIV.1409.3215. [Online]. Available: https://arxiv.org/abs/1409.3215.

[20] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, E. M. Smith, Y.-L. Boureau and J. Weston, 'Recipes for building an open-domain chatbot,' in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Online: Association for Computational Linguistics, Apr. 2021, pp. 300–325. DOI: 10.18653/v1/2021.eacl-main.24. [Online]. Available: https://aclanthology.org/2021.eacl-main.24.

[21] B. Hancock, A. Bordes, P.-E. Mazare and J. Weston, 'Learning from dialogue after deployment: Feed yourself, chatbot!' In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3667–3684. DOI: 10.18653/v1/P19-1358. [Online]. Available: https://aclanthology.org/P19-1358.

[22] V. Sharma, M. Goyal and D. Malik, 'An intelligent behaviour shown by chatbot system,' *International Journal of New Technology and Research*, vol. 3, no. 4, Apr. 2017.

[23] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. J. Crandall and D. Batra, 'Diverse beam search: Decoding diverse solutions from neural sequence models,' *CoRR*, vol. abs/1610.02424, 2016. arXiv: 1610.02424. [Online]. Available: http://arxiv.org/abs/1610.02424.

[24] A. Paranjape, A. See, K. Kenealy, H. Li, A. Hardy, P. Qi, K. R. Sadagopan, N. M. Phu, D. Soylu and C. D. Manning, 'Neural generation meets real people: Towards emotionally engaging mixed-initiative conversations,' *CoRR*, vol. abs/2008.12348, 2020. arXiv: 2008.12348. [Online]. Available: https://arxiv.org/abs/2008.12348.

[25] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley and J. Gao, 'Deep reinforcement learning for dialogue generation,' in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1192–1202. DOI: 10.18653/v1/D16-1127. [Online]. Available: https://aclanthology.org/D16-1127.

[26] J. Devlin, M. Chang, K. Lee and K. Toutanova, 'BERT: pre-training of deep bidirectional transformers for language understanding,' *CoRR*, vol. abs/1810.04805, 2018.

[27] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, 'Roberta: A robustly optimized BERT pretraining approach,' *CoRR*, vol. abs/1907.11692, 2019.

[28] K. Clark, M. Luong, Q. V. Le and C. D. Manning, 'ELECTRA: pre-training text encoders as discriminators rather than generators,' *CoRR*, vol. abs/2003.10555, 2020.

[29] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, 'Language models are few-shot learners,' *CoRR*, vol. abs/2005.14165, 2020.

[30] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, 'Exploring the limits of transfer learning with a unified text-to-text transformer,' *CoRR*, vol. abs/1910.10683, 2019.

[31] Y. Zhang, S. Sun, M. Galley, Y. Chen, C. Brockett, X. Gao, J. Gao, J. Liu and B. Dolan, 'Dialogpt: Large-scale generative pre-training for conversational response generation,' *CoRR*, vol. abs/1911.00536, 2019.

[32] M. Galley, C. Brockett, X. Gao, B. Dolan and J. Gao, 'End-to-End conversation Modeling: DSTC7 Task 2 Description,' in *DSTC7 workshop (forthcoming)*.

[33] Y. Long, J. Wang, Z. Xu, Z. Wang, B. Wang and Z. Wang, 'A knowledge enhanced generative conversational service agent,' in *Proceedings of the 6th Dialog System Technology Challenges (DSTC6) Workshop*, 2017.

[34] C. Hori and T. Hori, 'End-to-end conversation modeling track in dstc6,' *arXiv:1706.07440*, 2017.

[35] R. Lowe, N. Pow, I. Serban, L. Charlin and J. Pineau, 'Incorporating unstructured textual knowledge sources into neural dialogue systems,' in *Neural information processing systems workshop on machine learning for spoken language understanding*, 2015.

[36] P. Parthasarathi and J. Pineau, 'Extending neural generative conversational model using external knowledge sources,' *CoRR*, vol. abs/1809.05524, 2018.

[37] T. Scheepers, 'Improving the compositionality of word embeddings,' Ph.D. dissertation, Master's thesis. Universiteit van Amsterdam, Science Park 904, Amsterdam . . ., 2017.

[38] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka and T. M. Mitchell, 'Toward an architecture for never-ending language learning,' in *Twenty-Fourth AAAI conference on artificial intelligence*, 2010.

[39] S. Reddy, D. Chen and C. D. Manning, 'Coqa: A conversational question answering challenge,' *CoRR*, vol. abs/1808.07042, 2018.

[40] E. Choi, H. He, M. Iyyer, M. Yatskar, W. Yih, Y. Choi, P. Liang and L. Zettlemoyer, 'Quac : Question answering in context,' *CoRR*, vol. abs/1808.07036, 2018.

[41] W. Xiong, J. Wu, H. Wang, V. Kulkarni, M. Yu, S. Chang, X. Guo and W. Y. Wang, 'TWEETQA: A social media focused question answering dataset,' in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5020–5031. DOI: 10.18653/v1/P19-1496. [Online]. Available: https://aclanthology.org/P19-1496.

[42] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le and S. Petrov, 'Natural questions: A benchmark for question answering research,' *Transactions of the Association of Computational Linguistics*, 2019.

[43] P. Rajpurkar, J. Zhang, K. Lopyrev and P. Liang, 'Squad: 100, 000+ questions for machine comprehension of text,' *CoRR*, vol. abs/1606.05250, 2016.

[44] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder and L. Deng, 'MS MARCO: A human generated machine reading comprehension dataset,' *CoRR*, vol. abs/1611.09268, 2016.

[45] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov and C. D. Manning, 'Hotpotqa: A dataset for diverse, explainable multi-hop question answering,' *CoRR*, vol. abs/1809.09600, 2018.

[46] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela and J. Weston, 'Personalizing dialogue agents: I have a dog, do you have pets too?' *CoRR*, vol. abs/1801.07243, 2018.

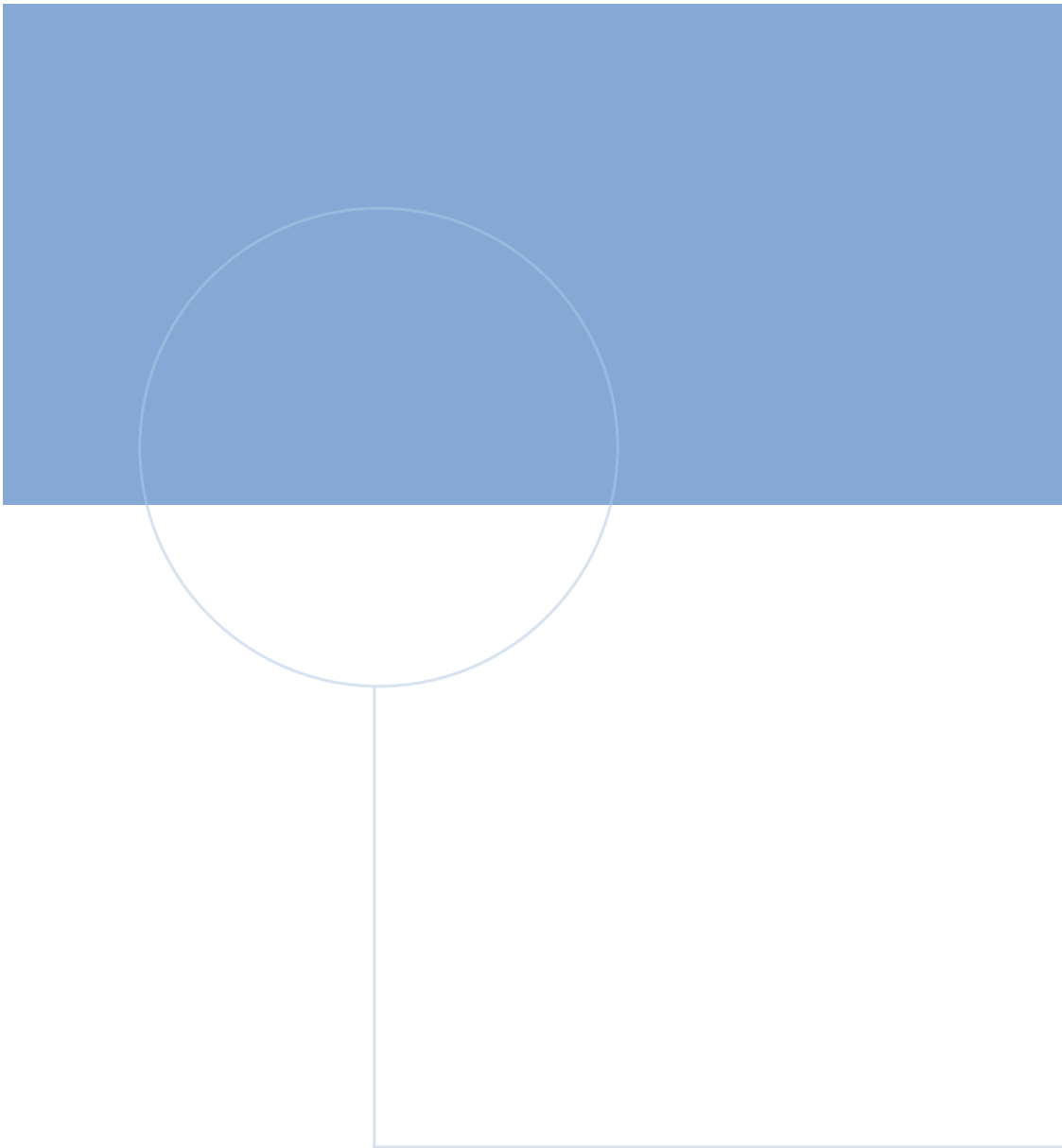
[47] M. Själander, M. Jahre, G. Tufte and N. Reissmann, *EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure*, 2019. arXiv: 1912.05848 [cs.DC].

[48] F. He, T. Liu and D. Tao, 'Control batch size and learning rate to generalize well: Theoretical and empirical evidence,' in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/dc6a70712a252123c40d2adba6a11d84-Paper.pdf.

[49] P. Gupta and S. MacAvaney, 'On survivorship bias in ms marco,' *arXiv preprint arXiv:2204.12852*, 2022.

[50] A. Bordes and J. Weston, 'Learning end-to-end goal-oriented dialog,' *CoRR*, vol. abs/1605.07683, 2016.

[51] M. Naaman, J. Boase and C.-H. Lai, 'Is it really about me? message content in social awareness streams,' Oct. 2010, pp. 189–192. DOI: 10.1145/1718918.1718953.



NTNU

Norwegian University of
Science and Technology