INSTITUTO
UNIVERSITÁRIO
DE LISBOA

A book-oriented Chatbot

Nuno Alexandre Mestre Barradas

Master in Telecommunications and Computer Engineering

Supervisor:

Doctor Ricardo Daniel Santos Faro Marques Ribeiro, Associate Professor,
Iscte – Instituto Universitário de Lisboa

Co-Supervisor:

Doctor Fernando Manuel Marques Batista, Associate Professor,
Iscte – Instituto Universitário de Lisboa

October, 2020

iscte

TECNOLOGIAS
E ARQUITETURA

Department of Information Science and Technology

A book-oriented Chatbot

Nuno Alexandre Mestre Barradas

Master in Telecommunications and Computer Engineering

Supervisor:

Doctor Ricardo Daniel Santos Faro Marques Ribeiro, Associate Professor,
Iscte – Instituto Universitário de Lisboa

Co-Supervisor:

Doctor Fernando Manuel Marques Batista, Associate Professor,
Iscte – Instituto Universitário de Lisboa

October, 2020

Resumo

A resposta automática a perguntas em língua natural é um tema estudado há largos anos. Tendo por base os sistemas existentes de resposta a perguntas, quando comparamos a percentagem de respostas correctas sobre um conjunto de perguntas, geradas a partir de um conjunto de dados, conseguimos ver que o desempenho está ainda longe de 100%, que muitas vezes é o valor alcançado quando as perguntas são testadas por humanos.

Este trabalho aborda a ideia de um agente conversacional orientado para livros, mais propriamente um sistema de resposta a perguntas direccionado para responder a perguntas cujo conjunto de dados seja um ou mais livros. Deste modo, pretendemos adoptar um novo sistema, incorporando dois projectos existentes, o OpenBookQA e o Question-Generation. Utilizámos dois conjuntos de dados de domínio específico, sem terem sido ainda estudados nos dois projectos, que foram o QA4MRE e o RACE. A estes aplicámos a abordagem principal: enriquecê-los com perguntas geradas automaticamente. Corremos uma série de experiências, treinando modelos de redes neuronais. Deste modo, pretendemos estudar o impacto das perguntas geradas e obter bons resultados de precisão de respostas correctas para os dois conjuntos de dados.

Os resultados obtidos sugerem que ter uma quantidade significativa de perguntas geradas num conjunto de dados, conduz a maior precisão de respostas correctas. Tornando claro que, enriquecer um dataset, sobre um livro, com perguntas geradas sobre esse mesmo livro, é dar ao dataset o conteúdo do livro.

Esta dissertação apresenta resultados promissores, a partir de conjuntos de dados com perguntas geradas automaticamente.

Palavras chave

resposta a perguntas

geração de perguntas

aprendizagem automática

agente conversacional

Abstract

The automatic answer to questions in natural language is an area that has been studied for many years. However, based on the existing question answering systems, the percentage of correct answers over a set of questions, generated from a dataset, we can see that the performance it is still far away from to 100%, which is many times the value achieved when the questions are tested by humans.

This work addresses the idea of a book-oriented Chatbot, more precisely a question answering system directed to answer to questions in which the dataset is one or more books. This way, we intend to adopt a new system, incorporating two existent projects, the OpenBookQA and the Question-Generation. We have used two Domain Specific Datasets that were not studied in both project, that were the QA4MRE and RACE. To these we have applied the main approach: enrich them with automatic generated questions. We have run many experiments, training neural network models. This way, we intended to study the impact of those questions and obtain good accuracy results for both datasets.

The obtained results suggest that having a significant representation of generated questions in a dataset, leads to a higher test accuracy results of correct answers. Becoming clear that, enrich a dataset, based on a book, with generated questions about that book, is giving to the dataset the content of the book.

This dissertation presents promising results, through the datasets with automatic generated questions.

Keywords

question answering

question generation

machine learning

chatbot

Acknowledgements

This Master thesis was a huge proof of resilience to me, because I have delayed my application in this Master's several times and due to some happenings in my personal life being too present during the development of it.

I want to thank my both Supervisors, Doctor Ricardo Ribeiro and Doctor Fernando Batista, for the constant support and guidance in the development of this thesis. They were tireless in the accompaniment during this work, telling me the truth, even in the first moment that I have approached them regarding the theme that I wanted to choose, and they told me that we should readjust it, until the last few months that we have doing almost always weekly meetings during this atypically year of 2020.

I want to thank my family, but specially to my Mother, she always encouraged me to study more, always saying the indisputable truth that there is always more space for knowledge. This work is specially dedicated to her. This thesis was not also possible without the support of my companion, Isabel, for always helping me in the moments of uncertainty. I also want to thank my Sister, Father, Parents-in-law, Aunt, Uncle, Grandmother and remaining family for the support. I also want to thank my friend Pedro for being always available and concerned about the development of this thesis.

This thesis was not possible without the help of INESC-ID on allowing us to do the experiments in their machines, a special acknowledgment to my Supervisors again for making it possible.

The last but not the least important, I want to thank ISCTE-IUL for giving me the opportunity to take the MSc in Telecommunications and Computer Engineering in this Institution.

Lisboa, 30 de Outubro de 2020
Nuno Barradas

Contents

List of Figures

List of Tables

1 Introduction

This chapter provides a brief introduction to this thesis topic, Question Answering (QA). It starts by presenting the motivation and then addresses the context of this work, research questions, goals, and research method.

1.1 Motivation

The work performed in the scope of this thesis consists in the idea of a book-oriented Chatbot, that aims to test new approaches over Domain Specific Question Answering Systems, increasing the size of the used datasets with generated questions, in order to obtain a good evaluation. Most of the work applies Natural Language Processing (NLP) approaches, an interdisciplinary field of computer science and linguistics, with focus on the ability for computers to understand human language [1].

One of the research areas of NLP is Question Answering (QA), which is the capability of the computers to be able to answer a human question (given one or more datasets). Those questions and answers can be spoken or written. We will focus on the written ones, more precisely, on textual language processing, question answering.

Over the last decades, the amount of digital information got overwhelming bigger, and since it comes very often to people in real-time, the necessity to obtain answers in the shortest time possible, became more indispensable.

In general, a Chatbot is a program capable to handle a written conversation with a human. When it receives a question, the system processes the question, probably simplifying and preparing it to be used in the system. Then, comes a search for a possible answer, based on the available knowledge in the database, that is often composed by hundreds of documents. The search can then be heavy due to filtering techniques, such as Information Retrieval-based or Knowledge-based, over those huge quantities of information. This description is very simple, so normally a system like this is much more complex. We want to explore a Question Answering System capable to understand a book, generating questions about it, and allowing the Chatbot to be trained based in those questions.

In this work we will use a well known Question Answering System, the OpenBookQA [2].

1.2 Context

A QA System can be considered an information retrieval system in the sense that it answers to queries or questions in natural language, giving as output an answer in natural language or the valid choice over several multiple choices [3].

Our research will be essentially based in QA, more specifically on what have been done more recently regarding this area. In general, there are two major paradigms in QA, information-retrieval-based question answering and knowledge-based question answering [4]. As mentioned in Section 1.1, we are going to use the OpenBookQA project, which uses Deep Learning for trainable neural models. Upon several neural baseline models, one of the most successful, *Question Match*, tries to predict the choice that best matches a question, without relying on external knowledge [2]. We will search for multiple-choice Domain Specific QA datasets to train, and for a question generator project, with the goal of generate questions through the source of the questions in the datasets. This way we can feed the datasets with generated questions, and analyze the impact on accuracy results of the trained neural network models in OpenBookQA.

Since this work will be focused in books, we want to know which advantages QA can bring over them. The most obvious way to access a book content or search for a specific answer on it, is through searches, however this way of operating has a lot of limitations. A Question Answering System would come to minimize this problem.

Below are some general advantages that can be achieved with a QA System:

- Close to real human conversation;
- Answers retrieval over questions in a very short time;
- Companies cost saving by the replacement of human attendance.

1.3 Research Questions

- Are there effective QA Systems over books?
  - If there are, which is their level of effectiveness?
- Basing on literature, what are the most effective approaches to answer questions over books?
- What strategies can be used to optimize Domain Specific QA System results?
  - Strengthen the Train sets with generated questions?
    - Can these robust Train sets improve the accuracy results?
    - What will be the results with different Embedding Systems?
    - Will the combination of these Train sets with External Knowledge provide better results?

1.4 Goals

The main goal is to build a new system, incorporating OpenBookQA [2] and Question-Generation[1] projects, able to train neural network models, with Train sets consolidated by generated questions, study the impact of these questions and increase the accuracy results.

The first step is to setup OpenBookQA successfully. After that, we will need to carefully choose the datasets to be used and in parallel adapt their structure, to be used in our system.

After that, we will need to find a Question Generation (QG) System that is able to generate questions given a specific input, that will come from the datasets mentioned before. Once we have the QG System, we will need to adapt it to the OpenBookQA dataset structure and populate the datasets with different proportions of questions per set, in order to explore the impact on the accuracy results after a model is trained.

After we have that, we will define the different experiment setups that we are going to use, run them, and, finally, analyze the results for each setup in order to understand the impacts of each setup.

Once we have the results and analysis made, it will be time to find conclusions about the accuracy obtained and the income of having generated questions appended to existent datasets.

1.5 Research Method

We will use the research method Design Science Research (DSR), a very well-known method, since we are developing a new artifact that intends to improve results in Domain Specific Question Answering Systems, making DSR the adequate methodology.

The DSR is motivated by the possibility of improving or giving new results to the science community, introducing new artifacts and processes for building those artifacts [5].

Like the DSR method describes, there are six main activities to guide the work, that will be described below, according to the work developed.

- Problem Identification
  - The problem is related with the uncertainty of the impact that generated questions, over domain specific datasets, can have in the accuracy results of a QA System. Specially when the datasets used are totally agnostics to the QA System in cause.
- Define Objectives for a Solution
  - The main objective is to improve the score of correct answers of the used QA System with new approaches.

[1]Question-Generation: https://github.com/KristiyanVachev/Question-Generation [Accessed: August 2020].

- Design and Development
  - As mentioned before, we will use the OpenBookQA project as base system;
  - For question's generation we will use the Question-Generation project[2];
  - To adapt certain files and datasets we will use Python and Shell Scripting.
- Demonstration
  - Do the setup of OpenBookQA project and make it able to run experiments;
  - Do the setup of Question-Generation project and make it able to generate questions with the correct format;
  - Generate questions for both datasets that we are going to use and populate them;
  - Run several experiments for both datasets using different setups.
- Evaluation
  - Analyze the results obtained in the different experiments and draw conclusions about them.
- Communication
  - A possible solution for this problem could be important for the advance of Artificial Intelligence, more precisely for the Natural Language Processing, Question Answering;
  - The difficulties inherent to searches over books could be partially suppressed. Its use could be very wide all over the professional areas, such as for research purposes, since it would decrease the time spent in the search for an answer, or to help a company to improve its online Chatbot.

1.6 Document Structure

In this section we provide a summary of the thesis structure from now on, explaining the main sections of each chapter.

Chapter 2 is dedicated to the Background, where some theoretical definitions are deepened. We start with an introduction to a more broad concept, Natural Language Processing, and after it we have two main sections. First we have Document Processing, where we describe more general topics like Tokenization, TF-IDF, Cosine Similarity, and Embeddings. Then, we address Question Answering, where the two major paradigms to deal with factoid question systems are briefly described to better introduce this thesis theory.

Chapter 3, named Related Work, is intended to give a better idea of what has been done until now regarding Question Answering, over Generic Knowledge and Domain Specific QA

[2]Question-Generation: https://github.com/KristiyanVachev/Question-Generation [Accessed: August 2020].

Systems. Since we have focused a lot of our work in the generation of questions, we have also tried to briefly introduce this theme.

Chapter 4 explains our methodology. Since we have used two projects to help us developing this thesis, we described their usage, importance and configuration, divided in two sections, respectively for those two projects, OpenBookQA and Question-Generation. We have then a final section dedicated to explain our adopted approach using both projects.

Chapter 5 presents the experiments and results obtained along this thesis, where we will give an explanation and some more details about the datasets used for the experiments. Then we will have a subsection to describe the *setups* to be the used along the experiments, followed by the Results subsection where we have the results of the experiments and an analysis of them.

Chapter 6 presents the Conclusions and Future Work, where we sum up our apprenticeships and what we can conclude with the work done.

Right after the Bibliography we will have the Appendices containing the most important JSON configuration files to be used in the projects described in the Chapter 4.

2 Background

Natural Language Processing (NLP) is an interdisciplinary scientific area that brings together Artificial Intelligence and Linguistics.

To understand what Natural Language Processing is, it is important to explain its meaning [6]:

- Natural Language is a way to communicate, either in written or spoken language, that has been evolving through the times, structurally and naturally;
- Processing means treating, analyzing, filtering and transforming input data with computers.

NLP intends to facilitate the communication between humans an machines, in the sense that it aims to teach machines how to process and comprehend human communication [6].

According to [6], and among others, we can emphasize three main topics regarding NLP:

- Speech Processing - The capability of the machine to convert spoken natural language into textual form. We have a clear example of that with some business voicemail systems and with Google Speech API, for instance;
- Natural Language Understanding - The capability of the machine to understand the human language. An example of this, is for instance the intelligent assistant Siri, from Apple, that is able to process and understand human communication;
- Natural Language Generation - The capability of the machines to generate a response in natural language, having into account a dialogue context. Siri is again a possible example.

Figure 2.1 shows some application examples of NLP according to the authors of [6].

2.1 Document Processing

In order for components to process documents, it is necessary to represent them in an appropriate manner. The next sections address aspects related to this work.

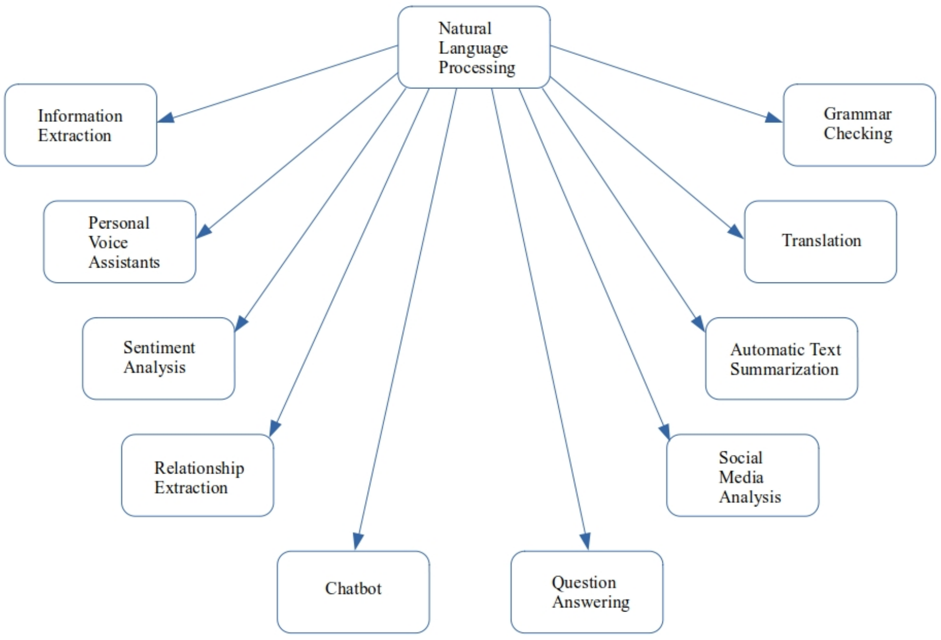

Figure 2.1: Application Examples of Natural Language Processing.

2.1.1 Tokenization

This is a well known process in document processing. It is responsible to break down a text into individual tokens, becoming the text more adequate to manipulate and analyze by a computer.

A token is considered a part of something, of course adapted to the context of text processing, so, like a word is a token in a sentence, a sentence is a token in a paragraph [6]. Therefore, it is possible to find two approaches for this process, either Sentence or Word Tokenization [7].

The following example demonstrates a simple Word Tokenization applied in a simple sentence:

Hi! This thesis treats Natural Language Processing.
Hi | ! | This | thesis | treats | Natural | Language | Processing | .

2.1.2 Term Frequency and Inverse Document Frequency (TF-IDF)

The TF-IDF are two well-known techniques in Text Mining and Machine Learning areas. They are both statistical measurements to evaluate how relevant a word is in a determined context.

- Term Frequency (TF) - Consists in the frequency of appearances that a specific word has in a document. There are several approaches to calculate it, we are going to describe one of them, which is the term frequency adjusted for document length. The process starts with the calculation of a term frequency - count of the times each term is written in a text - followed by the calculation of the normalized term frequency, that, simplifying is the division between the frequency and the total number of terms in that specific text. A simple formula to obtain the TF for a specific term in a document is:

$$TF(word) = \frac{Number\,of\,appearances\,of\,the\,term\,word\,in\,document}{Total\,number\,of\,terms\,in\,document}$$

- Inverse Document Frequency (IDF) - After the Term Frequency has been calculated, it can be noticed that all words are treated equally. It means that a word like "for", which is often a stop word, has the same importance as a word like "cat", and the words do not have all the same importance. The idea is to value the terms that occur less often, because normally they have more relevance, and in other way to devalue the terms that occur more often. A simple formula to obtain the IDF of a word in a group of documents is:

$$IDF(word) = log(\frac{Total\,number\,of\,documents}{Number\,of\,documents\,containing\,the\,word})$$

Finally comes the TF-IDF calculation, which is simply the multiplication between both values. A higher TF-IDF value is obtained when there is a higher term frequency value in a specific document, and a lower document frequency of the term in the set of documents. For practical usage, it is normal to obtain the TF-IDF factor for each term in a document, in order to reach each term weight [8].

2.1.3 Cosine Similarity

The cosine similarity is another measurement metric used to determine how similar the documents can be, despite of their size [9].

It measures the angle cosine between two vectors in a multi-dimensional space. A vector is the representation of the words of a document in a multi-dimensional space. It is commonly connected to TD-IDF, because the words in a document are converted to numbers, using those techniques, so that they can be represented as a vector.

A simple formula to obtain the cosine similarity between two angles is:

$$\cos(\theta) = \frac{A \prod B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

```
,    -0.082752 0.67204 -0.14987 -0.064983 0.056491 ...
.     0.012001 0.20751 -0.12578 -0.59325 0.12525 ...
the   0.27204 -0.06203 -0.1884 0.023225 -0.018158 ...
and  -0.18567 0.066008 -0.25209 -0.11725 0.26513
to    0.31924 0.06316 -0.27858 0.2612 0.079248 -0.21462
```

Figure 2.2: Excerpt from GloVe.

2.1.4 Embeddings

Embeddings are high-dimensional representations of the meaning of words, based of the different contexts where the words can appear [4].

Embeddings allows us to compare the similarity of words and then provide useful information to Natural Language Processing Models.

The connection between the similarity, in the way the words are distributed, and the similarity between their meaning is called distributional hypothesis [10]. It is based in the assumption that words are similar if they have similar contexts.

Although we can not assume that the meaning of one word is preserved when replaced by another one of the same context, some NLP applications require a demanding criterion when it comes to similarity, to guarantee that some words can be replaced accurately, when it is needed. For that we can have the help of several lexical relations, such as synonymy, hypernonymy, hyponymy, antonymy, and meronymy [11].

In this Project we will use two different Embedding Systems, GloVe and ELMo.

GloVe is an abbreviation for Global Vectors [6], and it is a global log-bilinear regression model for word representations [12]. Another well known Embedding System is Word2Vec, it only uses a small window around each word of each text, GloVe, like it can be understood by the name, looks into all the words presented in a text.

GloVe uses a co-occurrence count matrix to build the embeddings. With this matrix it is possible to study the relationship between two words, analyzing the ratio of the co-occurence probability [12]. A dimensional reduction is applied to this matrix to create the resulting embedding matrix, in each row will be an embedding vector per word. The Figure 2.2 shows an excerpt of the GloVe Embeddings.

The ELMo (Embeddings from Language Models) is a contextualized word representation that models syntax and semantics characteristics of words and how these uses change across different contexts, like model polysemy. The word vectors are learned functions, that are built using a deep bidirectional model (biLM), that is pre-trained with a large text corpus.

2.2 Question Answering (QA)

A large part of Question Answering Systems covers only the factoid questions.

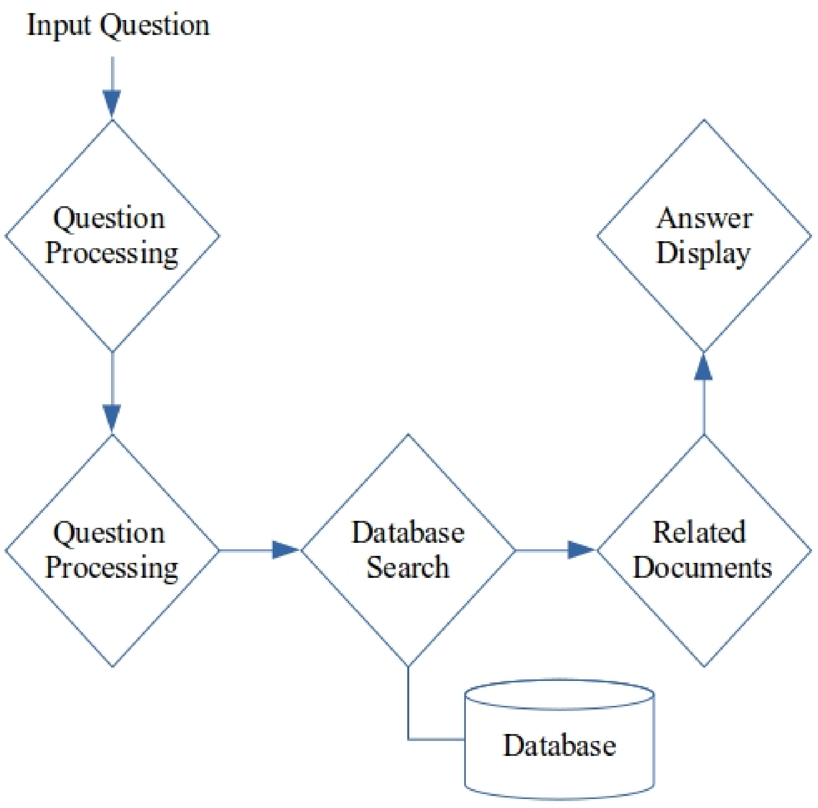

Figure 2.3: Question Answering Simple System Architecture.

According to [4], the two major paradigms that are used to deal with them are the Information-Retrieval (IR) and the Knowledge-Based (KB).

The human being has a natural will to ask for knowledge, that is why we have tried, since the decade of 1960, to add the feature of "ask information" to a computer. Given a specific dataset or group of documents, QA attempts to find the correct answer for a specific question in natural language [13].

Most of the Question Answering Systems consists in three sections: question processing, document processing and answer processing. They take precedence, in the sense that if one fails it will compromise the following ones.

Please see the Figure 2.3 to a better clarification over a simple Question Answering System, based in [13].

2.2.1 Information Retrieval-based QA

Information Retrieval-based QA approach rely on the textual information existent in the Web or in collections, such as PubMed or Wikipedia.

The way this paradigm works starts with an analysis on the question, and with that, a search for relevant documents or text passages is made. After this, comes a part where the systems resort to reading comprehension algorithms to build the answer directly from parts of the text [4].

The following example illustrates a question-answer pair that can be used in a IR System:

Q: Which is the capital of Portugal?
A: Lisbon

There are three phases for the factoid IR processing:

- Question Processing;
- Passage Retrieval and ranking;
- Answer Extraction.

The first phase, Question Processing, is responsible for collecting keywords and sometimes further information, such as: answer type, focus, and question type.

The query formulation is the task to create a reasonable query to send to an IR System to get documents that might have the desired answer. Commonly, are used techniques like TF-IDF and Cosine Similarity, described in Section 2.1, in IR Systems to index and search Documents. When it comes to search in Wikipedia, it helps to calculate TF-IDF over bigrams rather than unigrams [14]. In the web, a common query formulation rule applied is the query reformulation, which is basically transform the original query into a substring of it, removing non-relevant words.

The answer type is another common resource used in many systems, using answer type taxonomy to classify the question. Sometimes a single word can be enough to identify the answer type, it is called the *answer type word* or *question headword.* After this, it is possible to reduce the amount of documents to search.

The next step will be the Passage Retrieval, where basically with the IR query obtained by the question processing phase, an IR engine will search for relevant passages in the documents. This search can be overtaken to the next phase, answer extraction, or it can be done some filtering, seeking for an answer type.

Finally, we have Answer Extraction. In this task, the goal is to extract the answer from the passage. For that, it is used span labeling, where the span of text with the answer is extracted from the passage. It is commonly adopted a named entity tagger to verify if each passage has the answer.

There are two main types of Answer Extraction methodologies, Feature-based and Neural-based.

The Feature-based methods use several learning approaches to analyze which can be the correct answer for the inputted question, for instance: Answer Type, Pattern, Number of matched question keywords, Keyword distance, Novelty factor, Apposition features, Punctuation location and Sequences of question terms [4].

The Neural-based method tries to find the semantic similarity between the question and the answer, through reading comprehension algorithms. For a basic Neural Reading Comprehension algorithm, there are two arguments, the question and the passages of the documents. In a basic neural system like this, it is calculated an embedding for the question and an embedding for every token of each passage, and then are filtered passage excerpts based on the similarity between their embedding and the embedding of the question [4].

2.2.2 Knowledge-based QA

The Knowledge-based QA is based on a semantic representation of a given question. After that, a map to a logic query is done, to run on structured databases.

This methodology was introduced in 1961 with the System BASEBALL [15], where questions about baseball were answered based on a database of games and statistics.

So, normally the question is transformed into a logic form. This process is made by specific systems often named semantic-parsers. The logic form can be a predicate calculus or a query written in some language like SQL or SPARQL.

In order to facilitate the parsing from the original question to a logical representation of it, there are a lot of rule-based methods, supervised and semi-supervised methods. For instance, the word "When" is associated to a return type *Date*, and the word "Who" is associated to a return type *Person*, therefore it is possible to map *question types* to standard logical forms:

When was the Fall of the Berlin Wall? -> event-date (Fall of the Berlin Wall, ?x)

Who invented the lamp? -> relation (?x, lamp)

3 Related Work

This chapter analyzes some of work related to the topics of our project. For that, we will present some papers about Natural Language Understanding, Reading Comprehension and Domain Specific Question Answering Systems.

Our main focus is to work with multiple-choice questions datasets and systems capable to handle them and provide the best accuracy of correct answers possible.

3.1 Generic Knowledge QA

A big part of the existent Question Answering Systems uses only external knowledge as source, such as the Wikipedia.

The work "Reading Wikipedia to Answer Open-Domain Questions" [14] has the goal of answer factoid questions having only the Wikipedia as source. In order to answer each question, the proposed system uses Document Retrieval to find the relevant documents over more than five million articles, and then scans them to find the right answer for the questions. The authors named this as *machine reading at scale.*

The authors have developed a system to handle the purpose above, named DrQA, composed of two main modules, *Document Retriever* - a module using bigram hashing and TF-IDF techniques to filter potential documents - and *Document Reader* - a neural network machine comprehension model to find the correct answers in the filtered documents.

After the *Document Retriever* phase, the *Document Reader* System performs several important tasks to find a possible answer:

- Paragraph encoding;
- Question encoding;
- Prediction.

To train and test DrQA, the authors have used also the datasets SQuAD, CuratedTREC, WebQuestions, and WikiMovies.

The accuracy of the system was 70%. The authors reiterate that machine reading comprehension needs systems like DrQA to foster its success, and that this one should be improved.

Another work that we have analyzed, is a system composed by two components: probably one of the most known datasets in the world of the Natural Language Processing, is Stanford Question Answering Dataset (SQuAD) [16], and a logistic regression model to predict the correct answer for the questions of the dataset.

This system provides a huge dataset, made by crowd workers, with 107785 question-answer pairs, based in 536 articles from Wikipedia[1]. The dataset does not provide a list of answers for each question [16].

The dataset was built in three phases:

- Filter passages – Obtainment of the top 10,000 high-quality articles of Wikipedia and filtering them to only 536. From the filtered articles, there were obtained 23,215 paragraphs and divided in three datasets: training set (80%), development set (10%) and a test set (10%);
- Question-answer collection – Crowd workers from Amazon Mechanical Turk have written the questions, five for each worker, and the respective answers also, for human performance measurement;
- Additional answers collection – To help on getting a human performance indicator and to become the evaluation more robust, the crowd-workers have selected two more answers for each question of the development and test sets.

The authors of [16] have tested the dataset using the built logistic regression model, and compared its accuracy results with three baseline methods:

- Random Guess;
- Sliding Window [17];
- Sliding Window + Distance-Based extension [17].

A plenty of features were used such as, *Matching Word Frequencies*, *Matching Bigram Frequencies*, *Root Match*, *Lengths*, *Span Word Frequencies*, *Constituent Label*, *Span POS Tags*, *Lexicalized* and *Dependency Tree Paths*.

The highest score obtained was with the *Logistic Regression* method, achieving a *F1 Score* of 51%, while the human performance was around 86.8%. The *F1 Score* is a metric that calculates the average overlap between the prediction and the correct answer [16]. Although there was a considerable difference between the results, it was a big advance for Natural Language

[1]SQuAD: https://standford- qa.com [Accessed: March 2020].

Understanding too. The introduction of this dataset was a big step forward. It is nowadays a reference in all world to be used for researches since it is freely available.

The authors emphasize that should exist an effort to enrich the reading comprehension systems in order to improve the results obtained.

3.2 Domain Specific QA

Several researchers have been tackling the task of improving machine reading. One of the possible ways to improve results in this area is to provide more External Knowledge, and the careful selection of it, in order to do not provide unuseful knowledge to the system. We have analyzed some of their works. In "MCTest" [17], they have created a new dataset, with the help of Amazon Mechanical Turk workers, consisting in 500 stories and 2000 questions related to them. Each story has four questions related to it and each question has four multiple-choice answers, and no doubt that this is a way to get clear metrics, instead of answers in open text. The workers that wrote the stories and questions were asked to use a given vocabulary with 8,000 words, so that a seven year child could understand them, in order to facilitate the later machine reading process.

The stories were grammatically corrected by the research team and rated in age-appropriateness, grammatically, and clarity.

The baseline models used sliding window and word-distance based algorithms.

The sliding window approach choose a specific combination of words from the question to be the possible answer. On the other hand, the word-distance based algorithm analyzed the difference between words to find similarity between questions and answers.

The results for those baseline models reached 58% of correct answers, still far away from what the human judges got, nearly 90%.

3.2.1 OpenBookQA and Recent Improvements

Domain Specific Question Answering is one of the challenging tasks in Question Answering, because a system that implements it needs to gather the domain specific knowledge and perform multi-hop reasoning to choose the best answer to a question. We can compare the Domain Specific Question Answering task to the scenario of when a question about a book is asked to a human being and he can only consult that book [18].

A very interesting work regarding this matter is OpenBookQA [2]. It has a system that joins Domain Specific QA and Open-Domain QA, in the sense that it uses the domain specific source of knowledge but also external knowledge. This work is composed by a dataset and a system to answer the questions of the dataset. The dataset consists in 6,000 multiple-choice questions, each of those associated with one of 1,326 core facts and an auxiliary book of 6,000 secondary facts. Each of those core facts can trigger several questions. The authors report the use of a crowdfunding process to generate and compile those questions, resorting to workers with Master's Level to help them. The work was based on combining the core facts with the secondary ones and then to elect the multiple choice answers. The questions were analyzed and it was discovered that, 21% of the questions did not rely on the science facts (main ones). The additional facts have been classified in five categories: ISA, PROPERTY, DEFINITION, CAUSAL and BASIC. The majority of them have been categorized with PROPERTY.

The authors studied several experimental setups to seek the most effective one:

- No Training, External Knowledge only;
- No Training, Core Facts and External Knowledge;
- Trained Models and no External Knowledge;
- Trained Model with External Knowledge.

The *No Training, External Knowledge* is a setup in which there were used existing pre-trained systems relying specifically in their background knowledge, without taking into account the core facts of OpenBookQA.

Several succeeded solvers used to determine the right answer in scholar exams, did not have much success with OpenBookQA [2]. The authors have considered four such solvers, PMI, TableILP, TupleInference and DGEM. The first, PMI, uses pointwise mutual information to score every answer, based on their 280 GB of information (plain text). TableILP is an Integer Linear Programming (ILP), which is based on several relational database tables. Unfortunately, the small set of knowledge tables did not allow the system to answer 24% of the OpenBookQA questions. TupleInference, like TableILP, it is an ILP System, the system builds tuples composed by subject-verb-object, while it is getting information from their database based on the specific question. Finally, the DGEM, that is a neural model using an Open Information Extraction System.

Regarding *No Training, Core Facts and External Knowledge* setup, the authors have considered to provide the core facts to two existing solvers: the Information Retrieval Solver of [19] and with the TupleInference solver.

Regarding the *Trained Models and no External Knowledge*, they have experimented several neural baselines that were prepared for using the OpenBookQA dataset.

Some of those baselines were:

- Embeddings + Similarities as Features
  - The authors based themselves on the studies of [20]. Using centroid vectors built from the question, they have found the cosine similarities between the question and each possible answer;

- BiLSTM Max-Out Baselines
  - The authors have adopted a similar approach to the BiLSTM max-out model proposed by [20].

Finally, the *Trained Model with External Knowledge* setup helped to understand the importance of external knowledge in finding a correct answer. The model in this setup was a variant of the model described by [21], implemented using a *BiLSTM* baseline, in which for each question and possible answers, the external knowledge is analyzed independently [2]. There were some remarks achieved, namely that there is a clear need for external knowledge and deeper reasoning - there was 5% of improvement in the performance results when the science fact with the correct answer was provided into the knowledge-enhanced reader. For instance, [2] have reached the conclusion that the WordNet improved the score in 0.5%, while the ConceptNet reduced the score due to the distraction it caused in the system. The setup that has achieved the best results was the combined one, *Training Model with External Knowledge*.

Now that we know more about OpenBookQA, is time to analyze some more research regarding the users of OpenBookQA dataset.

3.2.2 Improving Results with External Knowledge

We have seen already that the external knowledge can improve significantly the correct answers accuracy.

The authors of [22] have used ARC-Easy, ARC-Challenge and OpenBookQA datasets to demonstrate it, and they have increased the accuracy results, in relation with the values obtained by [2], in 8.1, 13.0 and 12.8 p.p., respectively. They have incorporated two new sources of external knowledge: open-domain and in-domain. In relation with open-domain, they have enriched the corpus with *Wikipedia* relevant texts, while the in-domain refers to additional instances and reference corpora added for training. They also mentioned the challenge of incorporating unstructured and structured knowledge simultaneously as a future work.

In [23], the authors improved the state of the art achieved in [2] by 11.6 p.p., using careful selection of knowledge. They have shared two new approaches to extract knowledge over a knowledge base. One is based on language abduction to generate queries in order to retrieve missing knowledge, and the other has the goal of removing redundant knowledge, resorting to an Information Gain-based Re-ranking to reduce distractions.

3.3 Question Generation (QG)

Question generation has the goal of creating questions in natural language given an input text.

This area can have several applications but probably two of the most important are education [24] and chatbots.

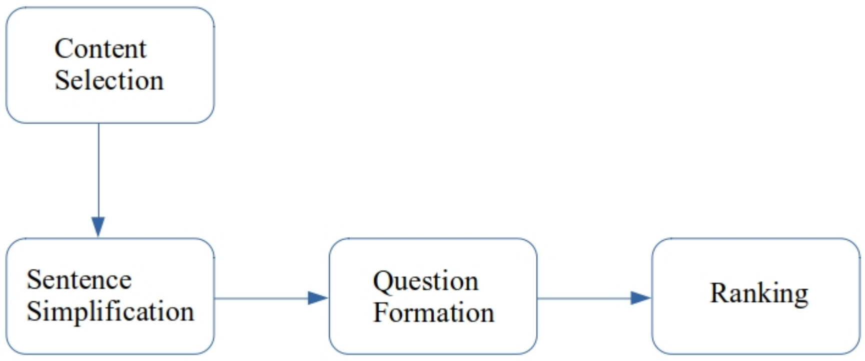

Figure 3.1: Stages of a Common QG System.

Most of the existent systems tackle the QG using a rule-based approach, having four abstract stages, like stated in the Figure 3.1. After Content Selection, there is a Sentence or Text Simplication, where each of the sentences is splitted into a set of derived declarative sentences, changing sometimes the lexical items and syntactic organization. The next module is Question Formation, where the system tries to generate interrogative sentences [25]. There can be some complex input texts, so this task it is not easy to complete, that is why that are usually a lot of small steps during the generation [26], such as:

- Elementary sentence construction - The system tries to simplify more complex texts, parsing them and analyzing a resultant syntactic tree representation;
- Sentence classification - This step is used for the most basic sentences. The parser obtains the normal sentence components, like subject, objects, preposition ans verb. After that, the system classifies each sentence in one of the five classifications: Human, Entity, Location, Time and Count.

The systems that have a QG module, often have also a module for ranking the questions for evaluation. The authors of [27] have it as the end stage of the process. As this stage they rank the generated questions and select the best ones.

They divide the process in two steps, the first will rank the questions by the depth of the predicate and guarantees that the questions from a main clause gets an higher rank that the ones that come from a secondary clause. The second step will order the questions with the same rank by the number of pronouns present in each question, and a lower score is attributed to the questions with more pronouns.

4 Methodology

To develop this work we have focused in two main projects: OpenBookQA and Question-Generation, both freely available on GitHub.

The OpenBookQA was the main project used, while the Question-Generation is a complement in the sense that we have made use of it to generate questions and answers to feed the datasets we chose, used in OpenBookQA project, to evaluate them.

This Chapter aims to describe each one of these projects and the importance of them for this work.

The final Section describes our adopted approach.

4.1 OpenBookQA Project

The OpenBookQA project is available on GitHub[1], supplied by Allen Institute for AI. Like we have previously seen in Section 3.2.1, this project is a new challenge for multihop reasoning with partial context, using both retrieval and reasoning mechanisms. The main goal is to, given a set of questions and multiple-choice answers, a set of scientific facts and sometimes external knowledge, achieve the highest score of correct answers, using different models, to get closer scores to the human performance. More details regarding this work are described in that Section.

We will describe the way how to configure it and how to do the setup of the project in order to run an experiment.

This project is mainly written in Python, using tools and frameworks such as PyTorch, scikit-learn, spaCy and CUDA. Besides Python there is a lot of Shell Scripting content to orchestrate and structure the project in order to organize it and make it runnable.

4.1.1 Setup

In Appendix A it is possible to consult the principal configurations, in the correct format, to be used in the Project.

[1]OpenBookQA: https://github.com/allenai/OpenBookQA [Accessed: March 2020].

Question:
A person wants to start saving money so that they can afford a nice vacation at the end of the year. After looking over their budget and expenses, they decide the best way to save money is to
A) make more phone calls
B) quit eating lunch out
C) buy less with monopoly money
D) have lunch with friends

Correct Answer:
B)

Science Fact:
Using less resources usually causes money to be saved.

Figure 4.1: OpenBookQA Dataset Excerpt.

Like in any system of this kind, it is necessary to provide a dataset to the system, in this case, splitted in three files: train, development and test. The configuration can be consulted in Appendix A.

Besides the dataset, it can be also provided an embedding model. We have used GloVe and ELMo. The configuration is available in Appendix A.

The configuration to run each experience is present in JSON files and it contains obligatorily the number of epochs, the configuration is available in Appendix A. An epoch is equivalent to a complete cycle over the full training dataset, and it is mandatory because we are using Neural Network Models. We have used the default configuration of 5 rounds of 40 epochs each.

Optionally, there are a lot of configuration parameters that can be added, such as the path for external knowledge, which was used for the experiences using the source from where the questions where based on and ConceptNet, the configuration is available in Appendix A.

There are two main ways to run an experience, with or without external knowledge. The main difference between both ways is that with knowledge, before we run an experience, it is necessary to rank the dataset knowledge for the given questions that will be evaluated in the experience, for this we can use the configuration detailed in Appendix A. For the both type of experiments, we can achieve new results by training a new model, giving the parameters already mentioned above, this way we can find new accuracy values to analyze.

4.1.2 Preliminary Validations

In order to validate the OpenBookQA setup was successfully concluded, we have performed some initial experiments with the datasets OpenBookQA and ARC, both mentioned in the work of [2].

The OpenBookQA dataset is composed by 5,957 multiple-choice questions, and each one associated with one fact from the 1,326 science facts that are used to answer them. The facts consist on a filtered subset of 2,287 WorldTree facts. The generation of the questions was

Question:
An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?
A) Planetary density will decrease.
B) Planetary years will become longer.
C) Planetary days will become shorter.
D) Planetary gravity will become stronger.

Correct Answer:
C)

Figure 4.2: ARC Dataset Excerpt.

done with a crowd-sourcing process, like described in [2]. Figure 4.1 shows an excerpt of this dataset, where we can see an example of a question and its answers, and the science fact that supports the example. The original dataset is in JSON format.

This AI2 Reasoning Challenge (ARC) dataset is composed by 7,787 multiple-choice questions. They were built based in a corpus of 14M science-related sentences from the Web, definitions from Wiktionary, and articles from Wikipedia. There are two subsets of questions in this dataset, one named Challenge Set and the other Easy Set, we made the experiment using the Challenge Set, composed by 2,590 questions [28].

Figure 4.2 illustrates an example of a question, and its answers of the ARC dataset. The original format is JSON.

As expected, we verified that the results obtained in our initial experiments were very similar to the ones described in [2]. We run two experiments, one with each dataset.

We have run the first experiment with OpenBookQA dataset plus GloVe only, and we have obtained a test accuracy of 0.510. Secondly, we have run the same experiment setup, but now with ARC dataset and we achieved a test accuracy of 0.336. The original results described in [28] reveal a value range between 0.493 and 0.511 for the first experiment, and an obtained value of 0.339 for the second.

Since that the first obtained accuracy feats in the original accuracy range and that the second has only a difference of 0.003 from the original, we had the direction that we were working properly with the system.

4.2 Question-Generation Project

Besides the OpenBookQA Project, we have also used another project, the Question-Generation[2]. Like the name suggests, it is a project able to generate questions for a specific input text. Additionally it is also able to generate a configurable number of multiple-choice answers for each question, indicating the correct one.

[2]Question-Generation: https://github.com/KristiyanVachev/Question-Generation [Accessed: August 2020].

Input text: “Electronic music intended primarily for dancing at nightclubs and raves.”

Question 1 Generated
"Electronic music _____ primarily for dancing at nightclubs and raves.”
A) “meant”
B) “designed”
C) “would”
D) “to”
E) “intended” (Correct answer)

Question 2 Generated
“Electronic music intended primarily for dancing at nightclubs and _____.”
A) “rave”
B) “plaudits”
C) “kudos”
D) “raves” (Correct answer)
E) “kudos”

Figure 4.3: Question-Generation Example.

The goal of generating questions is to have an easy way to understand a document or a set of documents. The generated questions can be used to train a system to answer questions about those documents.

This project is written in Python and it uses spaCy[3] and scikit-learn[4] as frameworks and GloVe as the Embedding System.

The *Gaussian Naive Bayes* algorithm, from scikit-learn, is used to classify every word, giving to each a probability to be the correct answer.

For each question generated there are also given four distractores, wrong answers. This is possible using GloVe and the cosine similarity, to help us on finding the most similar answers to the correct one.

Before we use the Question-Generation project, we can set the number of questions that we want to generate from the source text. Since the output was not compatible with the format that we use in OpenBookQA and we could not specify a file to where it should be written, we had to make some changes to meet these requirements.

When we run the Question-Generation project for a given text, the output is composed by the number of questions we have configured and five possible answers per each question.

Figure 4.3 shows an example of an input text and the generated questions and answers.

The project code to generate questions and a raw output is available in Appendix B.

[3]spaCy: https://spacy.io/ [Accessed: August 2020].
[4]scikit-learn: https://scikit-learn.org/stable/ [Accessed: August 2020].

4.3 Adopted Approach

Our adopted approach is then a symbiosis between the last two projects.

Figure 4.4 illustrates the general architecture of our project.

We have divided our project in three main phases:

1. Question Generation and Dataset Population

2. Adapt Dataset and Choose Experiment Setup

3. Run Experiment and Result Analysis

In the first phase we start with the generation of questions using the QUESTION-GENERATION project. As mentioned in the previous Section, to use the project is necessary to provide the facts or articles from the selected dataset to the project, and then choose the number of questions desired to be generated per each fact. We have generated from 50 to 250 questions per each input. Before we pass to the next phase, it is necessary to populate the dataset with the generated questions. This stage can be extensive due to the fact that it depends on the desired goals, in our case we have tried to build several dataset setups with the generated questions, but we have focused in two different paradigms:

- For each set of questions generated per fact, we built two dataset setups, oscilatting in the number of questions per the three sets: train, development and test;

- Maintain the same development and test sets for several dataset setups and increase the number of questions in the train set.

The second phase starts with the dataset adaptation to the OPENBOOKQA project. Like explained in the Section 4.1, it is very likely that a chosen dataset must be adapted to the JSON format used in OPENBOOKQA, because its structure needs to obey to certain rules. After that we need to choose which one of the three experiment setups we want to use, or even choose all, but not to run simultaneously, assuming that there is only one machine. The three setups are briefly described in the Section 5.2. We have made more experiments using the first, Final Dataset QA with Embeddings. This setup uses the adapted dataset and one of the two Embedding Systems we have used, GLOVE or ELMO. The middle setup, Final Dataset QA with Embeddings and Facts, will use also the facts that support the dataset questions, as knowledge. Finally, the Final Dataset QA with Embeddings, Facts and External Knowledge, will also use External Knowledge, which is ConceptNet 5.

In the last phase we run the experiment and analyze the results obtained. If we chose the experimental setup without knowledge we only need to train the model configured, otherwise we first need to rank the knowledge for the dataset questions.

Finally, we obtain results and it is possible to analyze the results, principally the test accuracy, which was our main evaluation metric.

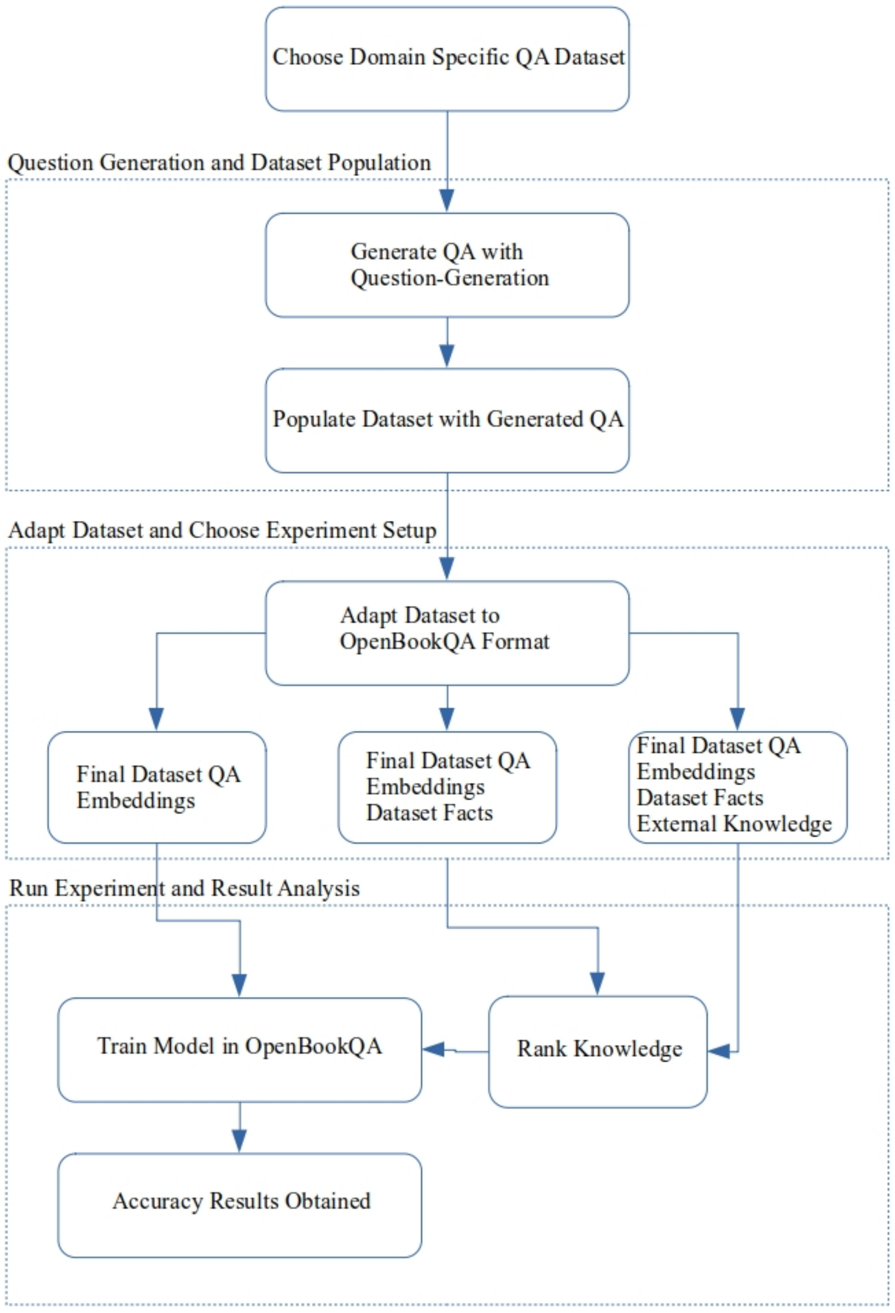
Choose Domain Specific QA Dataset
Question Generation and Dataset Population
Generate QA with Question-Generation
Populate Dataset with Generated QA
Adapt Dataset and Choose Experiment Setup
Adapt Dataset to OpenBookQA Format
Final Dataset QA Embeddings
Final Dataset QA Embeddings Dataset Facts
Final Dataset QA Embeddings Dataset Facts External Knowledge
Run Experiment and Result Analysis
Train Model in OpenBookQA
Rank Knowledge
Accuracy Results Obtained

Figure 4.4: Adopted System Architecture.

Experiments and Results 5

This chapter describes the experimental part of this work, their details, configurations and which datasets were used. Then, we will show the results of the experiments, discuss them and present the drawn conclusions. At the end, we summarize the results.

There was a sharp delay to obtain the first solid results due to several failed experiments, caused by low hardware capabilities to respond our requirements.

The first equipment we have used had 8 GB of RAM. Due to memory constraints, we explored Google Colab, an online platform that provides a Virtual Machine with almost 13 GB of RAM. The platform was not able to save the runtime environment, this suggested us to create a script that was responsible to build the project. Although it had better hardware capabilities than the other machine, it was not enough to run some of the experiments.

Finally, we have used a machine, from INESC-ID, with 250 GB of RAM, in which we were able to run all the experiments.

5.1 Data

We have used four datasets for the experimental part of the work, the Table 5.1 describes them: it shows the number of questions per dataset, the number of questions per each set of dataset, the total number of questions, the average number of words per question and per answer, and the minimum and maximum number of words that the questions of each dataset have. In the column *Datasets* we have several labels that describe the name of the Dataset, but with have some cases of dataset transformations, namely for QA4MRE (Section 5.1.1), and RACE (Section 5.1.2).

The label QA4MRE+50 means that, besides the QA4MRE base questions, there were generated 50 new questions, with 5 answers each, per each fact, so 800 new questions, that were distributed for the 3 sets. The same pattern was applied to QA4MRE+100, meaning that 100 questions were generated per each fact, giving the total of 1,600 questions.

The labels QA4MRE+150A and QA4MRE+150B have the same explanation than the ones before, for the part of the number of generated questions, that were, for this case, 150 new questions per fact. The capital letters A and B stands for different question distributions for the three sets. This allowed us to experiments with the same dataset but with different question

Datasets	Train	Dev	Test	Facts	Questions	Words per Question			Avg Choices
						Avg	Min	Max	
OpenBookQA	4957	500	500	1326	5957	10.643	1	68	4
ARC	1119	299	1172	0	2590	22.311	20	128	4
QA4MRE	95	95	94	16	284	10.611	3	31	5
QA4MRE+50	684	200	200	16	1084	23.502	1	89	5
QA4MRE+100	1084	400	400	16	1884	25.267	1	89	5
QA4MRE+150A	1484	600	600	16	2684	25.784	1	89	5
QA4MRE+150B	2284	200	200	16	2684	25.783	1	89	5
QA4MRE+150C	2684	200	200	16	3084	27.432	1	89	5
QA4MRE+150D	4095	200	200	16	4495	26.303	1	89	5
QA4MRE+200A	2684	400	400	16	3484	26.401	1	89	5
QA4MRE+200B	3084	200	200	16	3484	26.401	1	89	5
RACE	62445	3451	3498	20795	69394	10.411	1	62	4
RACE_SMALL	2053	500	500	890	3053	10.534	3	35	4

Table 5.1: Description of the Datasets.

proportions per each set of questions. So, we have configured two different setups, with the 2400 (150*16) questions. The QA4MRE+150C and QA4MRE+150D are two particular cases that have the same development and test sets of QA4MRE+150B, but having larger train sets. These train sets were populated with more generated questions, through the 16 available articles. Like this we can analyze the test accuracy with the same test set but with a different sized train.

The label RACE_SMALL refers to a reduced version of the RACE dataset, since we could not reproduce all the experiments with the original RACE due to technical constraints.

The following subsections contain more details about each dataset and more information about the transformations that we had to perform in the datasets content in order to use them.

5.1.1 QA4MRE Dataset

This dataset is composed by 284 multiple-choice question-answers and we have obtained it through the work done by [29].

These questions and answers were produced by the participants of a task released by the Conference and Labs of the Evaluation Forum (CLEF), where the focus was to build multiple-choice questions and answers based on 16 scientific articles.

The number of choices per question has an average of five, as it can be consulted in the Table 5.1.

In the Figure 5.1 we can see an excerpt of the QA4MRE dataset in the original format, XML.

Since that, originally, the dataset had only a test set, we had to split it in three, for us to have a train, development and a test sets. Besides that we had to transpose the XML structure to our JSON one in order to be used in our project.

```
<test-set>
  <topic t_id="1" t_name="Alzheimer">
    <reading-test r_id="1">
      <doc d_id="1"> Of mice and men: an Alzheimer's cure for our murine
          brethren. Alzheimer's Disease Alzheimer's disease is the most
         common form of dementia...
      </doc>
      <q  q_id="1" >
        <q_str>For what purpose were some mice injected with human genes
            that cause Alzheimer's?
        </q_str>
        <answer a_id='1'>cell generation</answer>
        <answer a_id='2'>sun exposure</answer>
        <answer a_id='3'>cell degeneration</answer>
        <answer a_id='4'>higher life expectancy</answer>
        <answer a_id='5'>None of the above</answer>
      </q>
    </reading-test>
  </topic>
</test-set>
```

Figure 5.1: QA4MRE Dataset Excerpt

5.1.2 RACE Dataset

The RACE dataset [30] it is composed by nearly 100,000 multiple-choice questions and answers, retrieved from English exams of Chinese middle and high schools, aggregating students in a range between 12 and 18 years old.

The questions are based in 28,000 passages and the questions were generated by humans, in this case by English instructors.

The number of choices per question varies between three and six, being the average four, as it can be seen in Table 5.1.

We needed to reduce the number of questions, answers and passages of the dataset in order to use it in our experiences, since we had a time estimation of 74 days for one of the experiments.

So we reduced the train set to 2,024 questions, the development set to 515 and the test set to 514. With this reduction we could run the experiences in a feasible time.

We needed to perform some modifications in the dataset structure also, to be able to be used in our project.

The original dataset was splitted in TXT files, each one with two or more questions, without letter assignment to the wrong answers and with a different structure.

In Figure 5.2 we can see an excerpt of the RACE dataset in the original format, JSON.

Therefore we needed to split the TXT files and then change the structure to be in the same JSON format that we have been using, namely to organize the JSON elements, change their name, give an identifier to each question and assign a letter to each wrong choice.

```
{
  "answers": [
    "C",
    "B",
    "C"
  ],
  "options": [
    [
      "Fucheng Garden Villas",
      "American Company--IDI",
      "Beijing Hongda Road Estate Co. Lid",
      "The Beijing Luftthansa"
    ],
    [
      "6.94",
      "More than 27.76",
      "27.76",
      "80%"
    ],
    [
      "From June 25 to July 31, 1997, you can buy or rent the Fucheng
         Garden Villas with a low price.",
      "You can't move into Fucheng Garden Villas before signing
         agreement",
      "If you haven't got enough money at the moment, you can't buy the
         Villas",
      "Not only equipment but also all materials of the construction and
          the decoration are made in America"
    ]
  ],
  "questions": [
    "The advertiser is   _  .",
    "How many hectares is Fucheng Garden Villas covered by trees and
       greens?",
    "According to the advertisement, which of the following statements
       is NOT true?"      ],
  "article": "Fucheng Garden Villas is situated along the North 4 th
     Ring Road , just 2 kilometres east away from the Asian Games
     Village with easy traffic connection. It is 5 kilometres from the
     Beijing Lufthansa Centre...",
  "id": "1"
}
```

Figure 5.2: RACE Dataset Excerpt.

5.1.3 External Knowledge

We have used ConceptNet 5[1], a semantic network representing words, sentences and relationships between them, to help machines understand the meaning of words used by humans. This network is fed by several sources of data, such as:

- Crowd-sources like Open Mind Common Sense[2];
- DBPedia[3], which is a project that extracts information boxes from Wikipedia articles;
- Wiktionary[4], a free multilingual dictionary.

5.2 Setup

The experiments were made using three kind of sources: the dataset, the embedding models and external knowledge.

We have tried to mix them in order to retrieve results for each possible combination.

Dataset+Embbedings

In this first setup we are going to perform experiments having as source the dataset and the Embbedings System only.

The Embbedings Systems, like previously mentioned, will be GloVe and ELMo.

Due to some system incompatibilities in OpenBookQA, we could only use GloVe or ELMo with this setup. The system is not prepared to use ELMo as Embbedings System for experiments with knowledge, which are the following 2 setups.

Dataset+Embbedings+Facts

In this setup, besides the dataset and Embbeding System, we add domain specific knowledge, namely the facts, also named as articles.

Dataset+Embbedings+Facts+EK

In this last setup, besides the dataset, the Embbeding System and the domain specific knowledge, we also are going to add external knowledge, ConceptNet.

[1]ConceptNet 5: https://github.com/commonsense/conceptnet5/wiki [Accessed: August 2020].
[2]Open Mind Common Sense: https://github.com/commonsense/omcs [Accessed: August 2020].
[3]DBPedia: https://wiki.dbpedia.org/ [Accessed: August 2020].
[4]Wiktionary: https://www.wiktionary.org/ [Accessed: August 2020].

Dataset Setup	**Accuracy**			**Time per Round**	**Total Time**
	Train	**Val.**	**Test**		
QA4MRE + GloVe	0.516	0.274	0.372	00:00:30	00:02:41
QA4MRE+50 + GloVe	0.894	0.215	0.278	00:05:00	00:23:08
QA4MRE+100 + GloVe	0.867	0.263	0.380	00:07:09	00:25:49
QA4MRE+150A + GloVe	0.776	0.342	0.417	00:04:53	00:28:12
QA4MRE+150B + GloVe	0.840	0.415	0.440	00:10:12	00:47:59
QA4MRE+150C + Glove	0.803	0.375	0.605	00:14:46	01:20:29
QA4MRE+150D + Glove	0.797	0.600	0.650	00:32:26	02:52:18
QA4MRE+200A + GloVe	0.849	0.370	0.450	00:14:16	01:02:38
QA4MRE+200B + GloVe	0.795	0.400	0.550	00:10:08	00:52:27
QA4MRE + ELMo	1.000	0.295	0.373	00:17:24	01:25:58
QA4MRE+50 + ELMo	0.997	0.185	0.304	01:40:10	06:51:46
QA4MRE + GloVe + Facts	0.863	0.326	0.351	02:21:40	06:56:45
QA4MRE+50 + GloVe + Facts	0.866	0.265	0.290	08:45:09	43:21:39
QA4MRE+100 + GloVe + Facts	0.867	0.303	0.390	18:54:53	59:47:48
QA4MRE + GloVe + Facts + EK	0.758	0.362	0.362	04:40:54	15:56:08
QA4MRE+50 + GloVe + Facts + EK	0.858	0.290	0.350	15:12:12	53:03:17
RACE + GloVe	0.553	0.431	0.397	04:45:05	23:48:44
RACE_SMALL + GloVe	0.907	0.320	0.309	01:24:04	03:54:42
RACE_SMALL + ELMo	0.999	0.377	0.352	06:16:05	38:03:26
RACE_SMALL + GloVe + Facts	0.838	0.330	0.334	16:39:12	24:10:08

Table 5.2: Experiment Results.

As mentioned before, we have used two Embedding Systems, GloVe and ELMo. However, due to configuration issues in OpenBookQA, we could only use ELMo in the experiments without domain specific or external knowledge.

We have used the default configuration to run the experiments, 5 rounds of 40 epochs each for all the experiments.

5.3 Results

The experiment results can be consulted in the Table 5.2. Like it was previously explained in the Section 5.2, we ran the experiments with three different base setups. It is possible to see some slice variations of them in Table 5.2, namely in the column *Dataset Setup*, where we have the following format: [DATASET NAME] + [Embbeding Name] + Facts + External Knowledge. The dataset name is explained in the Section 5.1, the Embeddings name refers to one of the two Embbedings Systems that we have used, GloVe and ELMo, the facts are the knowledge presented in the dataset and the External Knowledge is described in Section 5.1.3. The column *Accuracy* has three sub-columns that represent respectively the train, validation and test accuracies. The column *Time per Round* represents the time spent in the round with better results. Finally we have the Total Time, that represents the time spent in the experiment, that were 5 rounds of 40 epochs for each experiments, except for the RACE_SMALL + GloVe + Facts, which due to technical problems ran only 3 rounds.

We could not obtain results for an experiment that is not listed in Table 5.2 which was the RACE_SMALL + GloVe + Articles + EK. We have tried to run it several times, but due to technical problems, namely time constraints, it could never finish a round.

5.3.1 Execution Time

It is interesting to analyze the huge contrast between the time duration of the QA4MRE and RACE experiments. The total execution time of the experiments with QA4MRE was, for each experiment setup, much smaller than the respective RACE setup.

Throughout the experiments made with all kind of setups, we could observe that the total execution time grew up continuously.

If we first look closely to the Setup Dataset+Embbedings, the experiment QA4MRE + GloVe, that was the one with the smallest dataset, only 284 questions, took only 2 minutes and 41 seconds. Then, with QA4MRE+50 + GloVe, it already took sensitively 23 minutes. Each experiment took longer as the dataset was increasing in number of questions. Then, with the largest QA4MRE dataset used, in the experiment QA4MRE+150D, with 4,495 questions, it took approximatelly 2 hours and 52 minutes. The same behavior has been observed in the experiments made with the RACE dataset, being the faster experiment the RACE_SMALL + GloVe, followed by RACE + GloVe.

We can also observe that the total execution time of the experiments made with GloVe is substantially lower than the ones that were made with ELMo.

If we look to the experiments with the Setup Dataset+Embbedings+Facts, we can state that they were slower than the ones with the Setup Dataset+Embbedings. The faster experiment with this setup was naturally the QA4MRE + GloVe + Facts, because it had the smallest dataset: it took almost 7 hours, while the QA4MRE+50 + GloVe + Facts took around 43 hours.

To conclude, it is fair to say that the slowest experiments were the ones using the Setup Dataset+Embbedings+Facts+EK. The QA4MRE + GloVe + Facts + EK took almost 16 hours.

So, to sum up, it is observable that the execution time increased as we were appending more knowledge to the system.

The Figure 5.3 shows the execution time evolution for the experiments using QA4MRE.

5.3.2 Accuracy

The accuracy results got higher as we appended knowledge to the experiments, but the most significant rise in the accuracy values was the one noted throughout the experiments using QA4MRE with the Setup Dataset+Embeddings. The enrichment of the Train set along this setup could improve the results.

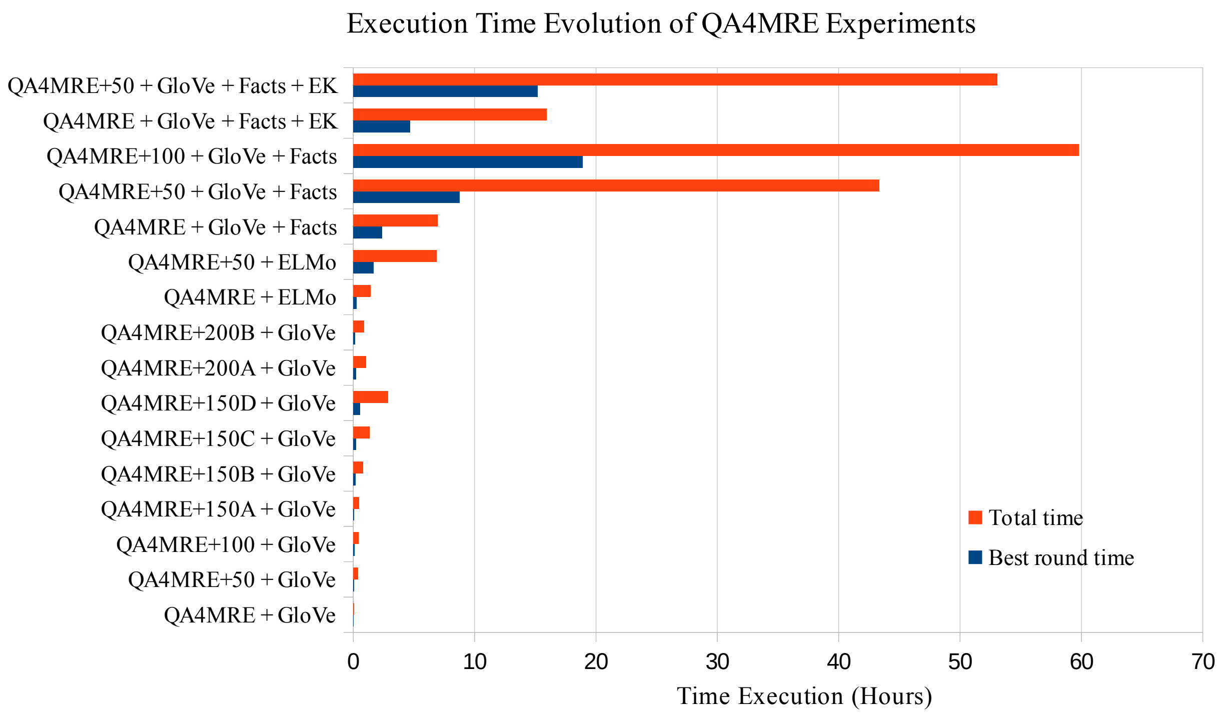

Figure 5.3: Execution Time Evolution for the Experiments Using QA4MRE.

In general, the increment of generated questions in the dataset increased the test accuracy, and a fine tune in the proportions of those questions through the three set files increased the accuracy even more.

We will deepen these topics along this subsection.

5.3.2.1 Importance of the Dataset size and Generated Questions

If we analyze the relation between the test accuracy obtained with the size of the datasets, we can see that, in relation to QA4MRE, for the experiments with the Setup Dataset+Embbedings, the test accuracy increased when the dataset size was bigger, so with a higher quantity of questions. In fact, if we did not have generated questions and feed the QA4MRE with them, we would have only poor results, since we achieved only 0.372 of test accuracy in the experiment QA4MRE + GloVe, that was with the raw dataset.

It is interesting to watch that, in the experiment QA4MRE+50 + GloVe, the results were even lower than in the QA4MRE + GloVe, and that they began to increase with the experiment QA4MRE+100 + GloVe. The test accuracy has increased until the largest QA4MRE dataset that we have used, in the QA4MRE+150D + GloVe experiment. The poor result obtained with QA4MRE+50 + GloVe can be justified with the fact that the system did not have a sufficient large train set to learn and consequently did not achieve a good test accuracy. The accuracy evolution for the experiments with the Setup Dataset+Embbedings using the QA4MRE dataset and GloVe can be observed in Figure 5.4.

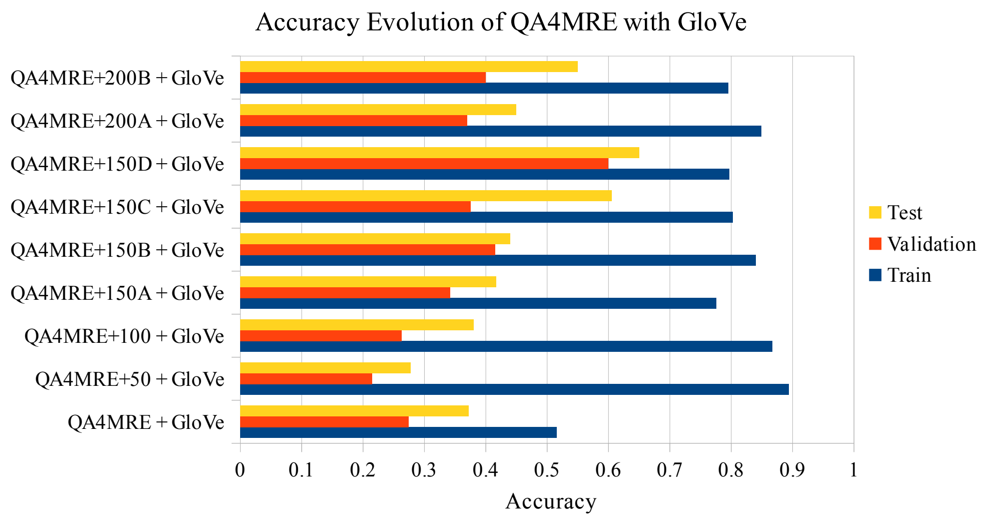

Figure 5.4: QA4MRE Accuracy Evolution Using Dataset+Embbedings.

5.3.2.2 Influence of Embeddings

When we compare the results obtained by the experiments using GloVe with the ones using ELMo, we can clearly affirm that the highest accuracy is assured by the experiments using ELMo. We can take for instance the example of the experiment RACE SMALL + ELMo, where the test accuracy was 0.352, while the similar experiment with GloVe has only reached 0.309. This happens because the Embedding System ELMo is a contextualized representation of words, and because of that can capture more information than GloVe, resulting in a better performance.

The principal Embbeding System that we have used in our experiments was GloVe. Like it was previously mentioned in the Setup Dataset+Embbedings, we could only run the experiments with ELMo in that setup, additionally, we have seen that the experiments with ELMo took significantly more time, and that would be too much affordable along this work.

5.3.2.3 Influence of Training Dimension

We can clearly see the influence of the dimension of the train set in the experiment setups where we have the same number of questions but with different proportions per each set. In the experiment QA4MRE+150A + GloVe, with 1,484 questions in the train set, we have obtained 0.417 of test accuracy, and in the experiment QA4MRE+150B + GloVe, that had 2,284 questions in the train set, we already obtained 0.440. The same pattern happened in the experiments QA4MRE+200A + GloVe and QA4MRE+200B + GloVe.

Due to the fact that this comparison was not 100% fair, because we have changed the test set for both scenarios (from A and B), we have added two another experiments to one of the setups, they were the QA4MRE+150C + GloVe and QA4MRE+150D + GloVe. On these experiments we have used the same development and test sets than in the QA4MRE+150B, but making the

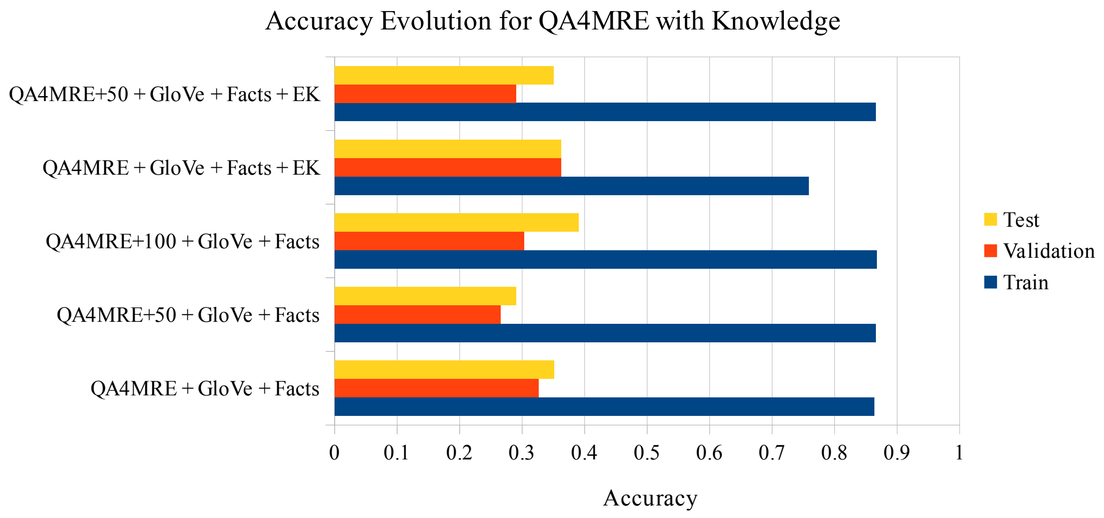

Figure 5.5: QA4MRE Accuracy Evolution with Knowledge.

train sets more robust, adding to them more 400 questions in the scenario C and more 1,411 in the scenario D.

We have achieved the biggest test accuracy until that moment with scenario C, with a value of 0.605, and after that we have run the D, that surpassed it and is our best score, with 0.650.

Therefore we can say that, when we increase the train set with more questions, in this case they were generated, the bigger will be the test accuracy. So it is really important that the train set has an expressive amount of questions in order to the system learn about the source knowledge of the questions, without having it in the raw format.

5.3.3 Results using External Knowledge

Regarding the experiments with setups having external knowledge, we have obtained the expected results. This means that the test accuracy of the experiments using the setup Dataset + Embbedings + Facts was higher than the respective experiment with the setup Dataset + Embbedings, but lower than the one with setup Dataset + Embbedings + Facts + EK.

The experiment QA4MRE+50 + GloVe + Facts had 0.290 of test accuracy, more 0.012 than the QA4MRE+50 + GloVe, and the QA4MRE+50 + GloVe + Facts + EK had 0.350.

With this, we assure that the experiments having external knowledge maintain the accuracy higher over the experiments without this knowledge, when using datasets having generated questions.

The accuracy evolution in the experiments with knowledge can also be consulted in Figure 5.5.

Conclusions and Future Work 6

This chapter summarizes the results obtained in the scope of this thesis. We will also note the contributions of this thesis to the scientific community. In the end, we will summarize the next remaining steps for future work in order to continue the research made here.

6.1 Conclusions

This thesis tackles the problem of automatically answering questions related to the contents of a given book. In order to achieved this goal we have started by performing experiments with a well-known project, OpenBookQA, using the datasets QA4MRE and RACE. We have enriched the QA4MRE dataset with generated questions, using the Question-Generation project, for several experimental setups to study its impact. The experiments provided significant accuracy results.

Concerning the answers to our Research Questions, presented in Section 1.3, we can say that, according to our literature review, we have not found any QA System over books. The system that we found that is closest to do so it is OpenBookQA [2], which original input data is not a book, but a set of science facts. The authors were able to achieve a test accuracy of 50%. The most effective approach used by OpenBookQA was composed by a model trained with the dataset facts and external knowledge. The model used a baseline model adapted from [20] — the BiLSTM Max-Out —, which process starts by the encoding of the tokens representing the question and the answers, followed by a bi-directional context encoder that retrieves their context representations. From these representations, a vector is built upon certain element-wise aggregation operations. Finally, OpenBookQA explored several configurations to retrieve the correct answer based on the vector described before. We highlight the Question Match, that gives an attention score to every question-answer pair, and it was one of the configurations that achieved better results.

Like the authors of OpenBookQA [2] suggested, an improvement of the external knowledge sources and its selection could be very interesting to explore. The authors of [23] could increase the test accuracy results, with an average increase of 11.3% when compared to OpenBookQA, by incorporating two new sources of external knowledge, in one of them by enriching the corpus with *Wikipedia* relevant texts. On the other hand, the authors of [23] improved their results using a more careful selection of knowledge, improving the accuracy results in 11.6% and sharing two new approaches to extracting it.

Our work shows that, increasing the size of the dataset, using generated questions, increases the test accuracy. Concerning the Embedding Systems, we could see that, for our case, using ELMo lead to better results than using GloVe.

Relatively to our last Research Question, we did not achieved any new conclusion. With the incorporation of the External Knowledge sources from OpenBookQA, we could retrieve the expected results, based on the ones retrieved by [2]. When adding them to an experiment, we have achieved higher test accuracies than without External Knowledge.

The major conclusion we have obtained is that, a significant quantity of generated questions in a dataset leads to a higher test accuracy. This allows us to say that giving questions about the book to the questions dataset is giving the book to it, which positively answers our main research question.

As it was expected, in relation to the execution time of the experiments, as we appended more components to an experiment, the bigger was the time execution. So, the *Dataset + Embeddings* experiments took normally less time than *Dataset + Embeddings + Facts* experiments, and this last set of experiments took normally less time than experiments with the *Dataset + Embeddings + Facts + EK* setup. The reasons that can interfere in the previous order of execution time are: the dataset size being bigger increases the execution time; in what concerns Embbedings, we could observe that the experiments using GloVe were substantially faster than the ones using ELMo; an experiment running with external knowledge will take more time than a similar one without knowledge.

We have achieved good test accuracy results in the experiments with the Setup *Dataset + Embeddings*. We have shown that the bigger the amount of generated questions in the dataset, the higher will be the test accuracy. So the dataset size has also a significantly importance to achieve consistent results. To have a well trained model it is necessary to have an expressive set of questions. We could see exactly this in three experiments we made having the same test set. One having a smaller train set, and the others increasing it exponentially. In the experiment with the larger train set we achieved our best test accuracy, which was of 0.650. Additionally, regarding this matter, we have also shown that using generated questions when using a smaller dataset can be a huge help to improve the accuracy of the system, like it was the case with QA4MRE.

We have also seen that the results using GloVe were worse than the ones using ELMo, but this last one has the downside of the execution time be much higher.

We could also verify the expected behavior related with the experiments using external knowledge, that the test accuracy was higher than the similar ones with less knowledge. So, the experiment using the Setup *Dataset + Embeddings + Facts + EK* achieved a higher accuracy than one using the *Dataset + Embeddings + Facts*. This also allow us to state that these experiments, with external knowledge, maintain a higher accuracy when using datasets with generated questions.

6.2 Future Work

As guidelines for future work and research that need to be made, we want to emphasize the importance that automatic generated multiple-choice question-answers can bring to a system, in order to increase the accuracy of Domain Specific Question Answering Systems.

Several long experiments can still be made incorporating the different scenarios that we propose. Namely the ones with ELMo, since it let to better results than with GloVe. So it would very interesting to investigate how to use ELMo with an experiment with knowledge in order to obtain new results for those setups. We think that there is still a lot to explore regarding the incorporation of generated questions and external knowledge, so it would be important to coordinate both areas.

There are still a lot to improve in the test accuracy results, but a good starting point could be, like we have mentioned in the previous Section, related to giving questions about a “book”: it would be interesting to provide even more generated questions to a dataset and compare it to Information Retrieval QA, until a level that it would not lead to an increase in the results.

Another really interesting approach to take would be to build a dataset totally composed by generated questions, and evaluate the questions resorting to humans, in comparison to the approach described in Section 4.3.

Finally, an interesting field that has a lot of space to be explored is the Question Generation. As seen in Section 4.2, the approach we have used has some limitations. Exploring other possibilities and assuring its impact can help us to understand how to better *read* books automatically.

Bibliography

[1] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," *CoRR*, vol. abs/1708.05148, 2017.

[2] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? A new dataset for open book question answering," *CoRR*, vol. abs/1809.02789, 2018.

[3] S. Quarteroni and S. Manandhar, "Designing an interactive open-domain question answering system," *Natural Language Engineering*, vol. 15, no. 1, p. 73, 2009.

[4] D. Jurafsky and J. Martin, "Speech and language processing (question answering chapter)," Prentice Hall, 2006.

[5] A. R. Hevner, "A three cycle view of design science research," *Scandinavian journal of information systems*, vol. 19, no. 2, p. 4, 2007.

[6] K. R. Bokka, S. Hora, T. Jain, and M. Wambugu, *Deep Learning for Natural Language Processing: Solve your natural language processing problems with smart deep neural networks*. Packt Publishing Ltd, 2019.

[7] R. M. da Conceição Rodrigues, *RAPPORT: a fact-based question answering system for Portuguese*. PhD thesis, Universidade de Coimbra (Portugal), 2017.

[8] S. Qaiser and R. Ali, "Text mining: use of tf-idf to examine the relevance of words to documents," *International Journal of Computer Applications*, vol. 181, no. 1, pp. 25–29, 2018.

[9] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, vol. 4, pp. 9–56, 2008.

[10] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[11] M. Geffet and I. Dagan, "The distributional inclusion hypotheses and lexical entailment," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 107–114, 2005.

[12] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[13] P. Gupta and V. Gupta, “A survey of text question answering techniques,” *International Journal of Computer Applications*, vol. 53, no. 4, 2012.

[14] D. Chen, A. Fisch, J. Weston, and A. Bordes, “Reading wikipedia to answer open-domain questions,” *CoRR*, vol. abs/1704.00051, 2017.

[15] B. Green, A. Wolf, C. Chomsky, and K. Laugherty, “Baseball: An automatic question answerer,” in *Proceedings Western Joint IRE-AIEE-ACM Computing Conference*, vol. 19, (Los Angeles, CA), pp. 219–224, 1961.

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

[17] M. Richardson, C. J. Burges, and E. Renshaw, “Mctest: A challenge dataset for the open-domain machine comprehension of text,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 193–203, 2013.

[18] P. Banerjee and C. Baral, “Knowledge fusion and semantic knowledge ranking for open domain question answering,” *arXiv preprint arXiv:2004.03101*, 2020.

[19] P. Clark, O. Etzioni, T. Khot, A. Sabharwal, O. Tafjord, P. D. Turney, and D. Khashabi, “Combining retrieval, statistics, and inference to answer elementary science questions,” in *AAAI*, 2016.

[20] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, “Supervised learning of universal sentence representations from natural language inference data,” *CoRR*, vol. abs/1705.02364, 2017.

[21] T. Mihaylov and A. Frank, “Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 821–832, Association for Computational Linguistics, July 2018.

[22] X. Pan, K. Sun, D. Yu, H. Ji, and D. Yu, “Improving question answering with external knowledge,” *CoRR*, vol. abs/1902.00993, 2019.

[23] P. Banerjee, K. K. Pal, A. Mitra, and C. Baral, “Careful selection of knowledge to solve open book question answering,” *CoRR*, vol. abs/1907.10738, 2019.

[24] X. Du, J. Shao, and C. Cardie, “Learning to ask: Neural question generation for reading comprehension,” *arXiv preprint arXiv:1705.00106*, 2017.

[25] M. Heilman and N. A. Smith, “Good question! statistical ranking for question generation,” in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 609–617, 2010.

[26] H. Ali, Y. Chali, and S. A. Hasan, “Automation of question generation from sentences,” in *Proceedings of QG2010: The Third Workshop on Question Generation*, pp. 58–67, 2010.

[27] P. Mannem, R. Prasad, and A. Joshi, "Question generation from paragraphs at upenn: Qgstec system description," in *Proceedings of QG2010: The Third Workshop on Question Generation*, pp. 84–91, 2010.

[28] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the AI2 reasoning challenge," *CoRR*, vol. abs/1803.05457, 2018.

[29] A. Peñas, E. Hovy, P. Forner, Á. Rodrigo, R. Sutcliffe, and R. Morante, "Qa4mre 2011-2013: Overview of question answering for machine reading evaluation," in *Information Access Evaluation. Multilinguality, Multimodality, and Visualization* (P. Forner, H. Müller, R. Paredes, P. Rosso, and B. Stein, eds.), (Berlin, Heidelberg), pp. 303–320, Springer Berlin Heidelberg, 2013.

[30] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. H. Hovy, "RACE: large-scale reading comprehension dataset from examinations," *CoRR*, vol. abs/1704.04683, 2017.

A OpenBookQA Configuration Files

Specification of the Three Dataset Paths

```
"train_data_path": "data/qa4mre/train.jsonl",
"validation_data_path": "data/qa4mre/dev.jsonl",
"test_data_path": "data/qa4mre/test.jsonl",
```

Specification of the Model Path

GloVe

```
"model": {
  "type": "qa_multi_choice_know_reader_v1",
  "text_field_embedder": {
    "tokens": {
      "type": "embedding",
      "pretrained_file": "data/glove/glove.840B.300d.txt.gz",
             "embedding_dim": 300,
      "trainable": false
      }
  }
```

ELMo

```
"model": {
  "type": "qa_multi_choice_max_att",  //This model is used for Question-
     to-Chocie, Choice-To-Choice, Choice-only
  "text_field_embedder": {
    "elmo":{
      "type": "elmo_token_embedder",
      "options_file": "https://s3-us-west-2.amazonaws.com/allennlp/
         models/elmo/2x4096_512_2048cnn_2xhighway/elmo_2x4096_512_2048
         cnn_2xhighway_options.json",
      "weight_file": "https://s3-us-west-2.amazonaws.com/allennlp/models
         /elmo/2x4096_512_2048cnn_2xhighway/elmo_2x4096_512_2048cnn_2
         xhighway_weights.hdf5",
      "do_layer_norm": false,
      "dropout": 0.0
    }
  },
```

Number of Epochs and Other Property Configurations

```
"trainer": {
  "num_epochs": 40,
  "patience": 20,
  "cuda_device": -1,
  "validation_metric": "+accuracy",
  "num_serialized_models_to_keep": 1,
  "optimizer": {
    "type": "adam",
    "lr": 0.001
  },
  "grad_norm":10,
  "learning_rate_scheduler": {
    "type": "reduce_on_plateau",
    "factor": 0.8,
    "mode": "max",
    "patience": 8
  }
}
```

External Knowledge Configuration

```
"external_knowledge": {
  "sources": [
    {
      "type": "flexible-json",
      "name": "cn5omcs",
      "use_cache": true,
      "dataset_to_know_json_file": {
        "any": "data/OpenBookQA-V1-Sep2018/Data/Main/ranked_knowledge/cn
           5omcs/knowledge.json"
      },
      "dataset_to_know_rank_file": {
        "any": "data/OpenBookQA-V1-Sep2018/Data/Main/ranked_knowledge/cn
           5omcs/full.jsonl.ranking.json"
      },
            "rank_reader_type": "flat-q-ch-values-v1",

"max_facts_per_argument": 10
    },
        {
      "type": "flexible-json",
      "name": "cn5wordnet",
      "use_cache": true,
      "dataset_to_know_json_file": {
        "any": "data/OpenBookQA-V1-Sep2018/Data/Main/ranked_knowledge/cn
           5wordnet/knowledge.json"
      },
          "dataset_to_know_rank_file": {
        "any": "data/OpenBookQA-V1-Sep2018/Data/Main/ranked_knowledge/cn
           5wordnet/full.jsonl.ranking.json"
      },
            "rank_reader_type": "flat-q-ch-values-v1",                "
               max_facts_per_argument": 10
    },
    {
      "type": "flexible-json",
      "name": "openbook",
      "use_cache": true,
      "dataset_to_know_json_file": {
        "any": "data/qa4mre/ranked_knowledge/knowledge.json"
      },
      "dataset_to_know_rank_file": {
        "any": "data/qa4mre/ranked_knowledge/full.jsonl.ranking.json"
      },
      "rank_reader_type": "flat-q-ch-values-v1",                    "
         max_facts_per_argument": 10
    },
  ],
  "sources_use": [false, true, true]
}
```

How to Run Experiences

Dataset with Embedding

```
config_file=training_config/qa/multi_choice/qa4mre/
   reader_mc_qa_question_to_choice_qa4mre.json bash scripts/experiments/
   qa/run_experiment_openbookqa.sh ${config_file}
```

Dataset with Embedding Plus External Knowledge

Here it is necessary to first Rank the Dataset knowledge facts for the given questions:

```
DATA_DIR_ROOT=data
KNOWLEDGE_DIR_ROOT=${DATA_DIR_ROOT}/knowledge/qa4mre
OPENBOOKQA_DIR=${DATA_DIR_ROOT}/qa4mre

ranking_out_dir=${DATA_DIR_ROOT}/qa4mre/ranked_knowledge/
mkdir -p ${ranking_out_dir}
data_file=${OPENBOOKQA_DIR}/full.jsonl
know_file=${KNOWLEDGE_DIR_ROOT}/articles.csv

PYTHONPATH=. python obqa/data/retrieval/knowledge/
   rank_knowledge_for_mc_qa.py \                      -o ${
   ranking_out_dir} -i ${data_file} \
        -k ${know_file} -n tfidf --max_facts_per_choice 100 \
                                        --limit_items 0
```

Then it is necessary to run the experience using the configuration file explained in the previous subsection.

```
config_file=training_config/qa/multi_choice/qa4mre/knowreader_v1
   _mc_qa_multi_source_qa4mre.json bash scripts/experiments/qa/
   run_experiment_openbookqa.sh ${config_file}
```

B Question-Generation Configuration

The following snippet code allows us to generate questions and answers giving a specific text:

```
# This example will generate and write two questions about the text into
    questions.txt, with ID starting in 1.
text = "Electronic music intended primarily for dancing at nightclubs
   and raves."
generateQuestions(text, 2, "questions.txt", 1)

# Content of questions.txt
{"id": "1",
  "question": {
    "stem": "Electronic music _____ primarily for dancing at nightclubs
       and raves.",
    "choices": [
      {"text": "meant", "label": "A"},
      {"text": "designed", "label": "B"},
      {"text": "would", "label": "C"},
      {"text": "to", "label": "D"},
      {"text": "intended", "label": "E"}]
  }, "answerKey": "E"}

{"id": "2",
  "question": {
    "stem": "Electronic music intended primarily for dancing at
       nightclubs and _____.",
    "choices": [
      {"text": "rave", "label": "A"},
      {"text": "plaudits", "label": "B"},
      {"text": "kudos", "label": "C"},
      {"text": "raves", "label": "D"},
      {"text": "kudos", "label": "E"}]
  }, "answerKey": "D"}
```

Noname manuscript No.
(will be inserted by the editor)

Learning to Answer Questions by Asking Questions

Nuno Barradas · Ricardo Ribeiro · Fernando Batista

Received: date / Accepted: date

Abstract In general, systems answering questions in natural language use a knowledge base or search different information sources to find an answer. This work explores the possibility of teaching a system to answer questions over a specific domain by automatically generating questions about that domain. Our approach incorporates two existent projects, the OpenBookQA and Question-Generation. We have used QA4MRE as domain specific dataset. Our experiments use neural network models to study the impact of this approach. The obtained results suggest that having an adequate coverage, using generated questions, of a dataset, leads to higher accuracy results.

Keywords question answering · question generation · machine learning · chatbot

1 Introduction

The human being has a natural will to ask for knowledge, that is why we have tried, since the decade of 1960, to add the feature of "ask information" to a computer. Given a specific dataset or group of documents, Question Answer (QA) systems attempts to find the correct answer for a specific question in natural language [8]. According to [10], there are two major paradigms that are used by QA systems: Information-Retrieval-based (IR) and Knowledge-based (KB) approaches.

Information Retrieval-based QA approaches rely on the textual information existent in the Web or in collections of documents, such as PubMed or Wikipedia. Systems based on this paradigm start with an analysis on the question; then, using

Nuno Barradas
Iscte - Instituto Universitário de Lisboa
E-mail: Nuno_Alexandre_Barradas@iscte-iul.pt

Ricardo Ribeiro
Iscte - Instituto Universitário de Lisboa, Portugal and INESC-ID Lisboa
E-mail: ricardo.ribeiro@inesc-id.pt

Fernando Batista
Iscte - Instituto Universitário de Lisboa, Portugal and INESC-ID Lisboa
E-mail: fernando.batista@inesc-id.pt

the results of the analysis they perform a search for relevant documents or text passages. Finally, these QA systems resort to reading comprehension algorithms to build the answer directly from parts of the text [10]. The first phase, Question Processing, is responsible for collecting keywords and sometimes further information, such as answer type, focus, and question type. The Query Formulation is the task of creating an adequate query to send to an IR System to get documents that might have the desired answer. The answer type classification is common in many systems, as it helps to decrease the number of documents to search. The next step will be Passage Retrieval, where an IR engine will search for relevant passages in the documents. Finally, the Answer Extraction phase has as goal to extract the answer from the retrieved passages. There are two main types of Answer Extraction methodologies, feature-based and neural networks-based. Feature-based methods use several approaches to analyze which can be the correct answer for the inputted question: answer type, pattern matching, number of matched question keywords, keyword distance, novelty factor, apposition features, punctuation location, and sequences of question terms. The neural networks-based methods try to find the semantic similarity between the question and the answer, through reading comprehension algorithms. For a basic neural reading comprehension algorithm, there are two arguments, the question and the passages of the documents.

Knowledge-based QA systems are based on mapping the semantic representation of a given question to logic query, to be run on structured databases. This type of systems first appeared in 1961: the BASEBALL system [7], answered questions about baseball based on a database of games and statistics. The mapping process is performed by semantic parsing. The logic form can be a predicate calculus formula or a query written in a querying language, such as SQL or SPARQL.

Domain Specific Question Answering is one of the challenging tasks in Question Answering, because a system that implements it needs to gather the domain specific knowledge and perform multi-hop reasoning to choose the best answer to a question. We can compare the Domain Specific Question Answering task to the scenario of when a question about a book is asked to a human being and he can only consult that book [2]. We explore the generation of questions as a method of providing the knowledge about which the system will answer.

This document is organized as follows: !TO DO

2 Learning to Answer Questions by Asking Questions

To develop this work we have focused in two main projects: OpenBookQA and Question- Generation, both freely available on GitHub. The OpenBookQA was the main project used, while the Question-Generation is a complement in the sense that we have made use of it to generate questions and answers to feed the datasets we chose, used in OpenBookQA project, to evaluate them.

2.1 OpenBookQA and Recent Improvements

Domain Specific Question Answering is one of the challenging tasks in Question Answering, because a system that implements it needs to gather the domain specific knowledge and perform multi-hop reasoning to choose the best answer to a question. We can compare the Domain Specific Question Answering task to the scenario of when a question about a book is asked to a human being and he can only consult that book [2].

A very interesting work regarding this matter is OpenBookQA [12]. It has a system that joins Domain Specific QA and Open-Domain QA, in the sense that it uses the domain specific source of knowledge but also external knowledge. This work is composed by a dataset and a system to answer the questions of the dataset. The dataset consists in 6,000 multiple-choice questions, each of those associated with one of 1,326 core facts and an auxiliary book of 6,000 secondary facts. Each of those core facts can trigger several questions. The authors report the use of a crowdfunding process to generate and compile those questions, resorting to workers with Master's Level to help them. The work was based on combining the core facts with the secondary ones and then to elect the multiple choice answers. The questions were analysed and it was discovered that, 21% of the questions did not rely on the science facts (main ones). The additional facts have been classified in five categories: ISA, PROPERTY, DEFINITION, CAUSAL and BASIC. The majority of them have been categorized with PROPERTY.

The authors studied several experimental setups to seek the most effective one:

- No Training, External Knowledge only;
- No Training, Core Facts and External Knowledge;
- Trained Models and no External Knowledge;
- Trained Model with External Knowledge.

The No Training, External Knowledge is a setup in which there were used existing pre-trained systems relying specifically in their background knowledge, without taking into account the core facts of OpenBookQA.

Several succeeded solvers used to determine the right answer in scholar exams, did not have much success with OpenBookQA [12]. The authors have considered four such solvers, PMI, TableILP, TupleInference and DGEM. The first, PMI, uses pointwise mutual information to score every answer, based on their 280 GB of information (plain text). TableILP is an Integer Linear Programming (ILP), which is based on several relational database tables. Unfortunately, the small set of knowledge tables did not allow the system to answer 24% of the OpenBookQA questions. TupleInference, like TableILP, it is an ILP System, the system builds tuples composed by subject-verb-object, while it is getting information from their database based on the specific question. Finally, the DGEM, that is a neural model using an Open Information Extraction System.

Regarding No Training, Core Facts and External Knowledge setup, the authors have considered to provide the core facts to two existing solvers: the Information Retrieval Solver of [4] and with the TupleInference solver.

Regarding the Trained Models and no External Knowledge, they have experimented several neural baselines that were prepared for using the OpenBookQA dataset. Some of those baselines were:

- Embeddings + Similarities as Features – the authors based themselves on the studies of [5]; Using centroid vectors built from the question, they have found the cosine similarities between the question and each possible answer;
- BiLSTM Max-Out Baselines – the authors have adopted a similar approach to the BiLSTM max-out model proposed by [5].

Finally, the Trained Model with External Knowledge setup helped to understand the importance of external knowledge in finding a correct answer. The model in this setup was a variant of the model described by [13], implemented using a BiLSTM baseline, in which for each question and possible answers, the external knowledge is analysed independently [12]. There were some remarks achieved, namely that there is a clear need for external knowledge and deeper reasoning - there was 5% of improvement in the performance results when the science fact with the correct answer was provided into the knowledge-enhanced reader. For instance, [12] have reached the conclusion that the WordNet improved the score in 0.5%, while the ConceptNet reduced the score due to the distraction it caused in the system. The setup that has achieved the best results was the combined one, Training Model with External Knowledge.

Now that we know more about OpenBookQA, is time to analyze some more research regarding the users of OpenBookQA dataset.

We have seen already that the external knowledge can improve significantly the correct answers accuracy. The authors of [14] have used ARC-Easy, ARC-Challenge and OpenBookQA datasets to demonstrate it, and they have increased the accuracy results, in relation with the values obtained by [12], in 8.1, 13.0 and 12.8 p.p., respectively. They have incorporated two new sources of external knowledge: open-domain and in-domain. In relation with open-domain, they have enriched the corpus with Wikipedia relevant texts, while the in-domain refers to additional instances and reference corpora added for training. They also mentioned the challenge of incorporating unstructured and structured knowledge simultaneously as a future work.

In [3], the authors improved the state of the art achieved in [12] by 11.6 p.p., using careful selection of knowledge. They have shared two new approaches to extract knowledge over a knowledge base. One is based on language abduction to generate queries in order to retrieve missing knowledge, and the other has the goal of removing redundant knowledge, resorting to an Information Gain-based Re-ranking to reduce distractions.

2.2 Question Generation (QG)

Question generation has the goal of creating questions in natural language given an input text. This area can have several applications but probably two of the most important are education [6] and chatbots.

Most of the existent systems address QG using a rule-based approach, having four abstract stages, like stated in the Figure 1. After Content Selection, there is a Sentence or Text Simplification step, where each of the sentences is split into a set of derived declarative sentences, changing sometimes the lexical items and syntactic organization. The next module is Question Formation, where the system tries to generate interrogative sentences [9]. There can be some complex input texts, so this task it is not easy to complete, that is why that are usually a lot of small steps during the generation [1], such as:

- Elementary sentence construction – The system tries to simplify more complex texts, pars- ing them and analyzing a resultant syntactic tree representation;

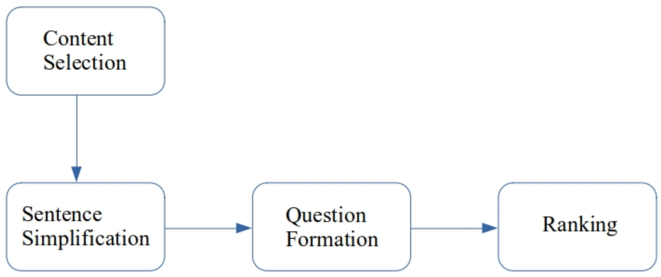

Fig. 1 Stages of a Common QG System.

- Sentence classification – This step is used for the most basic sentences. The parser obtains the normal sentence components, like subject, objects, preposition ans verb. After that, the system classifies each sentence in one of the five classifications: Human, Entity, Location, Time and Count.

The systems that have a QG module, often have also a module for ranking the questions for evaluation. The authors of [11] have it as the end stage of the process. As this stage they rank the generated questions and select the best ones. They divide the process in two steps, the first will rank the questions by the depth of the predicate and guarantees that the questions from a main clause gets an higher rank that the ones that come from a secondary clause. The second step will order the questions with the same rank by the number of pronouns present in each question, and a lower score is attributed to the questions with more pronouns.

2.3 Methodology

We have divided our project in three main phases. Figure 2 illustrates the general workflow of our project.

1. Question generation and dataset population;
2. Dataset adaption and experiment setup selection;
3. Run experiment and result analysis.

In the first phase we start with the generation of questions using the Question-Generation project. As mentioned in the previous Section, to use the project is necessary to provide the facts or articles from the selected dataset to the project, and then choose the number of questions desired to be generated per each fact. We have generated from 50 to 250 questions per each input. Before we pass to the next phase, it is necessary to populate the dataset with the generated questions. This stage can be extensive due to the fact that it depends on the desired goals, in our case we have tried to build several dataset setups with the generated questions, but we have focused in two different paradigms:

- For each set of questions generated per fact, we built two dataset setups, oscillating in the number of questions per the three sets: train, development and test;

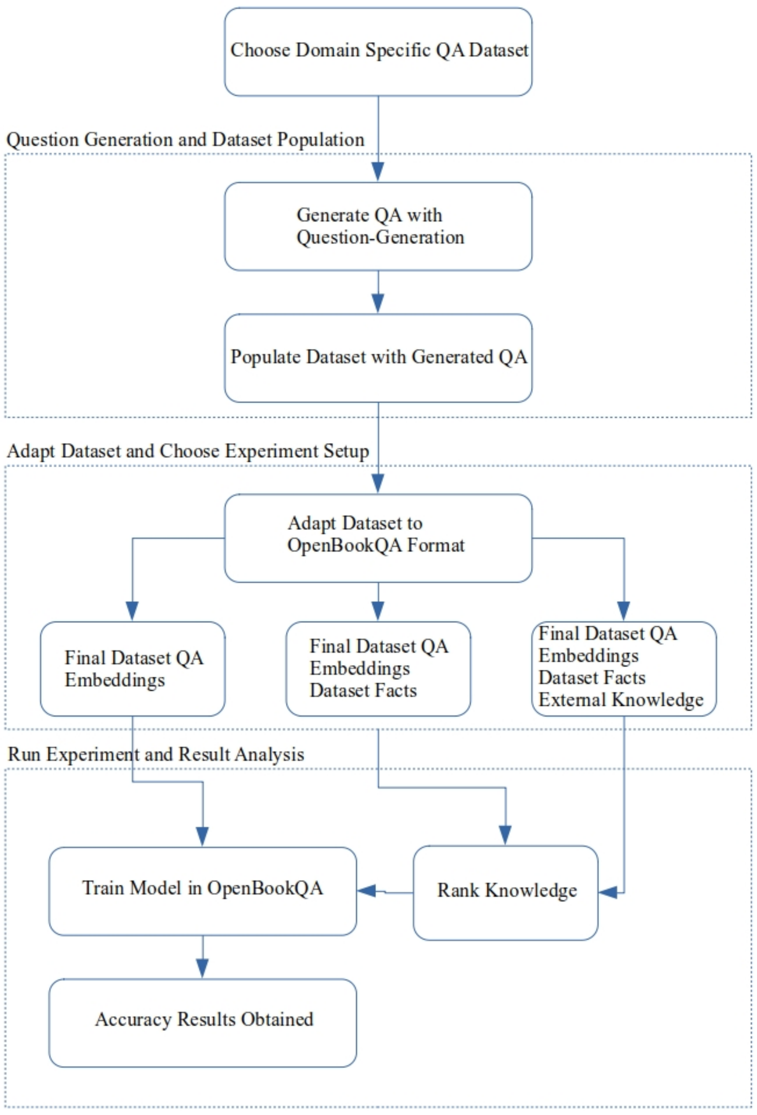
Choose Domain Specific QA Dataset
Question Generation and Dataset Population
Generate QA with Question-Generation
Populate Dataset with Generated QA
Adapt Dataset and Choose Experiment Setup
Adapt Dataset to OpenBookQA Format
Final Dataset QA Embeddings
Final Dataset QA Embeddings Dataset Facts
Final Dataset QA Embeddings Dataset Facts External Knowledge
Run Experiment and Result Analysis
Train Model in OpenBookQA
Rank Knowledge
Accuracy Results Obtained

Fig. 2 Workflow of the adopted approach.

Table 1 Description of the Datasets.

Datasets	train	dev	test	facts	# quest.	words per question avg	min	max	avg choices
OpenBookQA	4957	500	500	1326	5957	10.6	1	68	4
QA4MRE	95	95	94	16	284	10.6	3	31	5
QA4MRE+50	684	200	200	16	1084	23.5	1	89	5
QA4MRE+100	1084	400	400	16	1884	25.3	1	89	5
QA4MRE+150A	1484	600	600	16	2684	25.8	1	89	5
QA4MRE+150B	2284	200	200	16	2684	25.8	1	89	5
QA4MRE+150C	2684	200	200	16	3084	27.4	1	89	5
QA4MRE+150D	4095	200	200	16	4495	26.3	1	89	5
QA4MRE+200A	2684	400	400	16	3484	26.4	1	89	5
QA4MRE+200B	3084	200	200	16	3484	26.4	1	89	5

– Maintain the same development and test sets for several dataset setups and increase the number of questions in the train set.

The second phase starts with the dataset adaptation to the OpenBookQA project. Like previusly explained, it is very likely that a chosen dataset must be adapted to the JSON format used in OpenBookQA, because its structure needs to obey to certain rules. After that we need to choose which one of the three experiment setups we want to use, or even choose all, but not to run simultaneously, assuming that there is only one machine. We have made more experiments using the first, Final Dataset QA with Embeddings. This setup uses the adapted dataset and one of the two Embedding Systems we have used, GloVe or ELMo. The middle setup, Final Dataset QA with Embeddings and Facts, will use also the facts that support the dataset questions, as knowledge. Finally, the Final Dataset QA with Embeddings, Facts and External Knowledge, will also use External Knowledge, which is ConceptNet 5.

In the last phase we run the experiment and analyze the results obtained. If we chose the experimental setup without knowledge we only need to train the model configured, otherwise we first need to rank the knowledge for the dataset questions.

Finally, we obtain results and it is possible to analyze the results, principally the test accuracy, which was our main evaluation metric.

3 Experimental Evaluation

3.1 Data

We have used four datasets for the experimental part of the work, the Table 1 describes them: it shows the number of questions per dataset, the number of questions per each set of dataset, the total number of questions, the average number of words per question and per answer, and the minimum and maximum number of words that the questions of each dataset have.

The label QA4MRE+50 means that, besides the QA4MRE base questions, there were generated 50 new questions, with 5 answers each, per each fact, so 800 new questions, that were distributed for the 3 sets. The same pattern was

```
<test-set>
  <topic t_id="1" t_name="Alzheimer">
    <reading-test r_id="1">
      <doc d_id="1"> Of mice and men: an Alzheimer's cure for our
          murine brethren. Alzheimer's Disease Alzheimer's disease
          is the most common form of dementia...
      </doc>
      <q  q_id="1" >
        <q_str>For what purpose were some mice injected with human
            genes that cause Alzheimer's?
        </q_str>
        <answer a_id='1'>cell generation</answer>
        <answer a_id='2'>sun exposure</answer>
        <answer a_id='3'>cell degeneration</answer>
        <answer a_id='4'>higher life expectancy</answer>
        <answer a_id='5'>None of the above</answer>
      </q>
    </reading-test>
  </topic>
</test-set>
```

Fig. 3 QA4MRE Dataset Excerpt

applied to QA4MRE+100, meaning that 100 questions were generated per each fact, giving the total of 1,600 questions.

The labels QA4MRE+150A and QA4MRE+150B have the same explanation than the ones before, for the part of the number of generated questions, that were, for this case, 150 new questions per fact. The capital letters A and B stands for different question distributions for the three sets. This allowed us to experiments with the same dataset but with different question proportions per each set of questions. So, we have configured two different setups, with the 2400 (150*16) questions. The QA4MRE+150C and QA4MRE+150D are two particular cases that have the same development and test sets of QA4MRE+150B, but having larger train sets. These train sets were populated with more generated questions, through the 16 available articles. Like this we can analyze the test accuracy with the same test set but with a different sized train.

QA4MRE This dataset is composed by 284 multiple-choice question-answers and we have obtained it through the work done by [15]. These questions and answers were produced by the participants of a task released by the Conference and Labs of the Evaluation Forum (CLEF), where the focus was to build multiple-choice questions and answers based on 16 scientific articles. The number of choices per question has an average of five, as it can be consulted in the Table 1. In the Figure 3 we can see an excerpt of the QA4MRE dataset in the original format, XML.

External Knowledge We have used ConceptNet 5[1], a semantic network representing words, sentences and relationships between them, to help machines understand the meaning of words used by humans. This network is fed by several sources of data, such as:

[1] ConceptNet 5: https://github.com/commonsense/conceptnet5/wiki [Accessed: August 2020].

- Crowd-sources like Open Mind Common Sense[2];
- DBPedia[3], which is a project that extracts information boxes from Wikipedia articles;
- Wiktionary[4], a free multilingual dictionary.

3.2 Setup

Our experiments use three different information sources that were combined int order to produce results for each possible combination: the *dataset*, the *embedding models* and the *external knowledge*.

Dataset+Embbedings: Experiments with this setup use the dataset and the Embbedings System (GloVe and ELMo) as source. Due to system incompatibilities in OpenBookQA, we could only use GloVe or ELMo with this setup. The system is not prepared to use ELMo as Embbedings System for experiments with knowledge, which are the following 2 setups.

Dataset+Embbedings+Facts: This setup also adds domain specific knowledge, namely the facts or articles.

Dataset+Embbedings+Facts+EK: This last setup also adds external knowledge, ConceptNet.

We have used 5 rounds of 40 epochs for all the experiments, which correspond the default configuration parameters.

3.3 Results

The experiment results can be consulted in the Table 2. As previously explained, we ran the experiments with three different base setups. It is possible to see some slice variations of them in Table 2, namely in the column *Dataset Setup*, where we have the following format: [DATASET NAME] + [Embbeding Name] + Facts + External Knowledge. *Embeddings Name* refers to one of the two Embbedings Systems that we have used, GloVe and ELMo, the facts are the knowledge presented in the dataset and the External Knowledge. The column *Accuracy* has three sub-columns that represent respectively the train, validation and test accuracies. The column *Time per Round* represents the time spent in the round with better results. Finally we have the Total Time, that represents the time spent in the experiment, that were 5 rounds of 40 epochs for each experiment.

[2] Open Mind Common Sense: https://github.com/commonsense/omcs [Accessed: August 2020].

[3] DBPedia: https://wiki.dbpedia.org/ [Accessed: August 2020].

[4] Wiktionary: https://www.wiktionary.org/ [Accessed: August 2020].

Table 2 Experiment Results.

Dataset Setup	Accuracy			Time per Round	Total Time
	Train	Val.	Test		
QA4MRE + GloVe	0.516	0.274	0.372	00:00:30	00:02:41
QA4MRE+50 + GloVe	0.894	0.215	0.278	00:05:00	00:23:08
QA4MRE+100 + GloVe	0.867	0.263	0.380	00:07:09	00:25:49
QA4MRE+150A + GloVe	0.776	0.342	0.417	00:04:53	00:28:12
QA4MRE+150B + GloVe	0.840	0.415	0.440	00:10:12	00:47:59
QA4MRE+150C + Glove	0.803	0.375	0.605	00:14:46	01:20:29
QA4MRE+150D + Glove	0.797	0.600	0.650	00:32:26	02:52:18
QA4MRE+200A + GloVe	0.849	0.370	0.450	00:14:16	01:02:38
QA4MRE+200B + GloVe	0.795	0.400	0.550	00:10:08	00:52:27
QA4MRE + ELMo	1.000	0.295	0.373	00:17:24	01:25:58
QA4MRE+50 + ELMo	0.997	0.185	0.304	01:40:10	06:51:46
QA4MRE + GloVe + Facts	0.863	0.326	0.351	02:21:40	06:56:45
QA4MRE+50 + GloVe + Facts	0.866	0.265	0.290	08:45:09	43:21:39
QA4MRE+100 + GloVe + Facts	0.867	0.303	0.390	18:54:53	59:47:48
QA4MRE + GloVe + Facts + EK	0.758	0.362	0.362	04:40:54	15:56:08
QA4MRE+50 + GloVe + Facts + EK	0.858	0.290	0.350	15:12:12	53:03:17

3.3.1 Execution Time

Throughout the experiments made with all kind of setups, we can observe that the total execution time grew up continuously. If we first look closely to the Setup Dataset+Embbedings, the experiment QA4MRE + GloVe, that was the one with the smallest dataset, only 284 questions, took only 2 minutes and 41 seconds. Then, with QA4MRE+50 + GloVe, it already took sensitively 23 minutes. Each experiment took longer as the dataset was increasing in number of questions. Then, with the largest QA4MRE dataset used, in the experiment QA4MRE+150D, with 4,495 questions, it took approximatelly 2 hours and 52 minutes. We can also observe that the total execution time of the experiments made with GloVe is substantially lower than the ones that were made with ELMo. If we look to the experiments with the Setup Dataset+Embbedings+Facts, we can state that they were slower than the ones with the Setup Dataset+Embbedings. The faster experiment with this setup was naturally the QA4MRE + GloVe + Facts, because it had the smallest dataset: it took almost 7 hours, while the QA4MRE+50 + GloVe + Facts took around 43 hours.

To conclude, it is fair to say that the slowest experiments were the ones using the Setup Dataset+Embbedings+Facts+EK. The QA4MRE + GloVe + Facts + EK took almost 16 hours. So, to sum up, it is observable that the execution time increased as we were appending more knowledge to the system.

The Figure 4 shows the execution time evolution for the experiments using QA4MRE.

3.3.2 Accuracy

The accuracy results got higher as we appended knowledge to the experiments, but the most significant rise in the accuracy values was the one noted throughout the experiments using QA4MRE with the Setup Dataset+Embeddings. The enrichment of the Train set along this setup could improve the results.

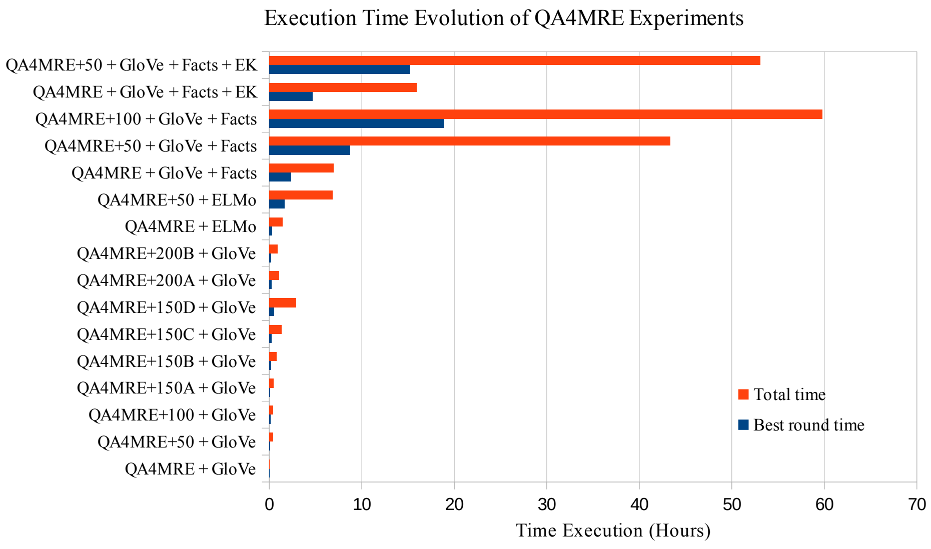

Fig. 4 Execution Time Evolution for the Experiments Using QA4MRE.

In general, the increment of generated questions in the dataset increased the test accuracy, and a fine tune in the proportions of those questions through the three set files increased the accuracy even more.

We will deepen these topics along this subsection.

Importance of the Dataset size and Generated Questions If we analyze the relation between the test accuracy obtained with the size of the datasets, we can see that, in relation to QA4MRE, for the experiments with the Setup Dataset+Embbedings, the test accuracy increased when the dataset size was bigger, so with a higher quantity of questions. In fact, if we did not have generated questions and feed the QA4MRE with them, we would have only poor results, since we achieved only 0.372 of test accuracy in the experiment QA4MRE + GloVe, that was with the raw dataset.

It is interesting to watch that, in the experiment QA4MRE+50 + GloVe, the results were even lower than in the QA4MRE + GloVe, and that they began to increase with the experiment QA4MRE+100 + GloVe. The test accuracy has increased until the largest QA4MRE dataset that we have used, in the QA4MRE+150D + GloVe experiment. The poor result obtained with QA4MRE+50 + GloVe can be justified with the fact that the system did not have a sufficient large train set to learn and consequently did not achieve a good test accuracy. The accuracy evolution for the experiments with the Setup Dataset+Embbedings using the QA4MRE dataset and GloVe can be observed in Figure 5.

Influence of Embeddings Comparing the results obtained in the experiments using GloVe to the ones using ELMo, we can clearly affirm that the highest accuracy is achieved in the experiments using ELMo. This happens because ELMo generates contextualized representations of words, capturing more information than GloVe.

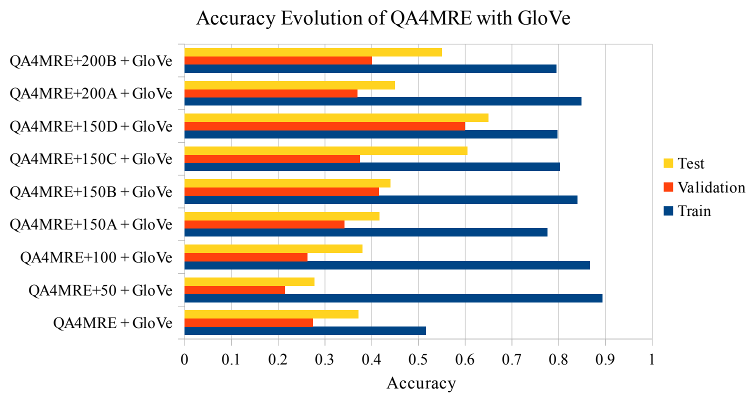

Fig. 5 QA4MRE Accuracy Evolution Using **??**.

Influence of Training Dimension We can clearly see the influence of the dimension of the train set in the experiment setups where we have the same number of questions but with different proportions per each set. In the experiment QA4MRE+150A + GloVe, with 1,484 questions in the train set, we have obtained 0.417 of test accuracy, and in the experiment QA4MRE+150B + GloVe, that had 2,284 questions in the train set, we already obtained 0.440. The same pattern happened in the experiments QA4MRE+200A + GloVe and QA4MRE+200B + GloVe.

Due to the fact that this comparison was not 100% fair, because we have changed the test set for both scenarios (from A and B), we have added two another experiments to one of the setups, they were the QA4MRE+150C + GloVe and QA4MRE+150D + GloVe. On these experiments we have used the same development and test sets than in the QA4MRE+150B, but making the train sets more robust, adding to them more 400 questions in the scenario C and more 1,411 in the scenario D.

We have achieved the biggest test accuracy until that moment with scenario C, with a value of 0.605, and after that we have run the D, that surpassed it and is our best score, with 0.650.

Therefore we can say that, when we increase the train set with more questions, in this case they were generated, the bigger will be the test accuracy. So it is really important that the train set has an expressive amount of questions in order to the system learn about the source knowledge of the questions, without having it in the raw format.

3.3.3 Results using External Knowledge

Regarding the experiments with setups having external knowledge, we have obtained the expected results. This means that the test accuracy of the experiments using the setup Dataset + Embbedings + Facts was higher than the respective experiment with the setup Dataset + Embbedings, but lower than the one with setup Dataset + Embbedings + Facts + EK.

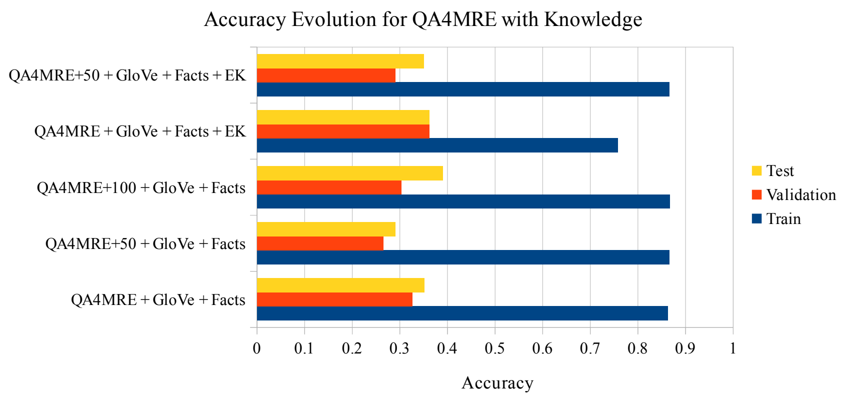

Fig. 6 QA4MRE Accuracy Evolution with Knowledge.

The experiment QA4MRE+50 + GloVe + Facts had 0.290 of test accuracy, more 0.012 than the QA4MRE+50 + GloVe, and the QA4MRE+50 + GloVe + Facts + EK had 0.350.

With this, we assure that the experiments having external knowledge maintain the accuracy higher over the experiments without this knowledge, when using datasets having generated questions.

The accuracy evolution in the experiments with knowledge can also be consulted in Figure 6.

4 Conclusion

In this work, we explore the generation of questions as means to improve the knowledge of a question answering system. In order to achieved this goal we used the OpenBookQA framework for question answering and the dataset QA4MRE as domain of knowledge. We have enriched the QA4MRE dataset with generated questions, using the Question-Generation project, for several experimental setups to study its impact. The experiments provided significant accuracy results.

The major conclusion we have obtained is that, a significant quantity of generated questions in a dataset leads to a higher test accuracy. This allows us to say that giving questions about the book to the questions dataset is giving the book to it, which positively answers our main research question. As expected, in relation to the execution time of the experiments, as we appended more components to an experiment, the bigger was the time execution. So, the *Dataset + Embeddings* experiments took normally less time than *Dataset + Embeddings + Facts* experiments, and this last set of experiments took normally less time than experiments with the *Dataset + Embeddings + Facts + EK* setup. The reasons that can interfere in the previous order of execution time are: the dataset size being bigger increases the execution time; in what concerns Embbedings, we could observe that the experiments using GloVe were substantially faster than the ones using ELMo; an experiment running with external knowledge will take more time than a similar one without knowledge.

We have achieved good test accuracy results in the experiments with the Setup *Dataset + Embeddings*. We have shown that the bigger the amount of generated questions in the dataset, the higher will be the test accuracy. So the dataset size has also a significantly importance to achieve consistent results. To have a well trained model it is necessary to have an expressive set of questions. We could see exactly this in three experiments we made having the same test set. One having a smaller train set, and the others increasing it exponentially. In the experiment with the larger train set we achieved our best test accuracy, which was of 0.650. Additionally, regarding this matter, we have also shown that using generated questions when using a smaller dataset can be a huge help to improve the accuracy of the system, like it was the case with QA4MRE.

We have also seen that the results using GloVe were worse than the ones using ELMo, but this last one has the downside of the execution time be much higher.

We could also verify the expected behavior related with the experiments using external knowledge, that the test accuracy was higher than the similar ones with less knowledge. So, the experiment using the Setup *Dataset + Embeddings + Facts + EK* achieved a higher accuracy than one using the *Dataset + Embeddings + Facts*. This also allow us to state that these experiments, with external knowledge, maintain a higher accuracy when using datasets with generated questions.

As guidelines for future work and research that need to be made, we want to emphasize the importance that automatic generated multiple-choice question-answers can bring to a system, in order to increase the accuracy of Domain Specific Question Answering Systems. Several long experiments can still be made incorporating the different scenarios that we propose. Namely the ones with ELMo, since it let to better results than with GloVe. So it would very interesting to investigate how to use ELMo with an experiment with knowledge in order to obtain new results for those setups. We think that there is still a lot to explore regarding the incorporation of generated questions and external knowledge, so it would be important to coordinate both areas. There are still a lot to improve in the test accuracy results, but a good starting point could be, like we have mentioned in the previous Section, related to giving questions about a "book": it would be interesting to provide even more generated questions to a dataset and compare it to Information Retrieval QA, until a level that it would not lead to an increase in the results. Another interesting approach would be to build a dataset totally composed by generated questions, and evaluate the questions resorting to humans, in comparison to our adopted approach. Finally, knowing thar our approach concerning Question Generation has limitations, it would be interesting to explore other possibilities and assess their performance when applied to books.

Acknowledgements This work was supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020

References

1. Ali, H., Chali, Y., Hasan, S.A.: Automation of question generation from sentences. In: Proceedings of QG2010: The Third Workshop on Question Generation, pp. 58–67 (2010)
2. Banerjee, P., Baral, C.: Knowledge fusion and semantic knowledge ranking for open domain question answering. arXiv preprint arXiv:2004.03101 (2020)

3. Banerjee, P., Pal, K.K., Mitra, A., Baral, C.: Careful selection of knowledge to solve open book question answering. CoRR **abs/1907.10738** (2019). URL `http://arxiv.org/abs/1907.10738`
4. Clark, P., Etzioni, O., Khot, T., Sabharwal, A., Tafjord, O., Turney, P.D., Khashabi, D.: Combining retrieval, statistics, and inference to answer elementary science questions. In: AAAI (2016)
5. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. CoRR **abs/1705.02364** (2017). URL `http://arxiv.org/abs/1705.02364`
6. Du, X., Shao, J., Cardie, C.: Learning to ask: Neural question generation for reading comprehension. arXiv preprint arXiv:1705.00106 (2017)
7. Green, B., Wolf, A., Chomsky, C., Laugherty, K.: Baseball: An automatic question answerer. In: Proceedings Western Joint IRE-AIEE-ACM Computing Conference, vol. 19, pp. 219–224. Los Angeles, CA (1961)
8. Gupta, P., Gupta, V.: A survey of text question answering techniques. International Journal of Computer Applications **53**(4) (2012)
9. Heilman, M., Smith, N.A.: Good question! statistical ranking for question generation. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 609–617 (2010)
10. Jurafsky, D., Martin, J.: Speech and Language Processing, chap. Question Answering. Prentice Hall (2006)
11. Mannem, P., Prasad, R., Joshi, A.: Question generation from paragraphs at upenn: Qgstec system description. In: Proceedings of QG2010: The Third Workshop on Question Generation, pp. 84–91 (2010)
12. Mihaylov, T., Clark, P., Khot, T., Sabharwal, A.: Can a suit of armor conduct electricity? A new dataset for open book question answering. CoRR **abs/1809.02789** (2018). URL `http://arxiv.org/abs/1809.02789`
13. Mihaylov, T., Frank, A.: Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 821–832. Association for Computational Linguistics, Melbourne, Australia (2018). DOI 10.18653/v1/P18-1076. URL `https://www.aclweb.org/anthology/P18-1076`
14. Pan, X., Sun, K., Yu, D., Ji, H., Yu, D.: Improving question answering with external knowledge. CoRR **abs/1902.00993** (2019). URL `http://arxiv.org/abs/1902.00993`
15. Peñas, A., Hovy, E., Forner, P., Rodrigo, Á., Sutcliffe, R., Morante, R.: Qa4mre 2011-2013: Overview of question answering for machine reading evaluation. In: P. Forner, H. Müller, R. Paredes, P. Rosso, B. Stein (eds.) Information Access Evaluation. Multilinguality, Multimodality, and Visualization, pp. 303–320. Springer (2013)