MARCOS MENON JOSÉ

A DEEP REINFORCEMENT LEARNING QUESTION ANSWERING SYSTEM FOR COMPLEX QUESTIONS USING TEXTS AND TABLES

São Paulo
2024

MARCOS MENON JOSÉ

A DEEP REINFORCEMENT LEARNING QUESTION ANSWERING SYSTEM FOR COMPLEX QUESTIONS USING TEXTS AND TABLES

Dissertation presented to Escola Politécnica of Universidade de São Paulo to obtain Master of Sciences degree in Eletrical Engineering.

São Paulo
2024

MARCOS MENON JOSÉ

A DEEP REINFORCEMENT LEARNING QUESTION ANSWERING SYSTEM FOR COMPLEX QUESTIONS USING TEXTS AND TABLES

Final Version

Dissertation presented to Escola Politécnica of Universidade de São Paulo to obtain Master of Sciences degree in Eletrical Engineering.

Concentration Area
Computer Engineering

Advisor:
Fabio Gagliardi Cozman

São Paulo
2024

Catalogação-na-publicação

José, Marcos
A Deep Reinforcement Learning Question Answering System for Complex Questions using Texts and Tables / M. José -- versão corr. -- São Paulo, 2024.
81 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Perguntas e respostas 2.Aprendizado por Reforço 3.Multi-Hop 4.Redes Neurais Transformer 5.Inteligência Artificial I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

ACKNOWLEDGMENTS

This research has been carried out with support by *Itaú Unibanco S.A.* through the scholarship program *Programa de Bolsas Itaú* (PBI); it is also supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001, Brazil. Computational resources of the Center for Artificial Intelligence USP/IBM/FAPESP were employed, thanks to FAPESP and IBM Corporation for that support.

Any opinions, findings, and conclusions expressed in this manuscript are those of the authors and do not necessarily reflect the views, official policy or position of the Itaú-Unibanco, FAPESP, IBM, and CAPES.

I wish to express appreciation for the invaluable assistance and insights provided by Maria Fernanda Ribeiro, Flávio Nakasato, Paulo Pirozelli, and Rafael Cheang. Their help has been instrumental in the success of this work.

A special thanks goes to my supervisor, Prof. Fabio G. Cozman, for guiding me not only through my master's program but teaching me lessons that will shape my path as a researcher.

Finally, I extend my sincere thanks to my family—my mother, Márcia; my father, Artur; my brother, Luis; and my wife, Fernanda—for their support at every step of this journey.

"If a machine is expected to be infallible, it cannot also be intelligent"

-Alan Turing-

RESUMO

A geração de respostas a questões é um dos principais tópicos atuais em processamento de linguagem natural, podendo ser utilizado em diversas aplicações distintas. Este projeto propõe uma arquitetura original para resolver questões de domínio aberto e multi-hop entre textos e tabelas, utilizando o conjunto de dados OTT-QA para validação e treinamento. Para responder tais questões, é necessário buscar informações em um grande corpus percorrendo vários trechos e tabelas, pois a resposta não pode ser encontrada diretamente; é preciso raciocinar usando diferentes passagens. Uma das soluções mais comuns é recuperar as informações de forma sequencial, onde um texto encontrado ajuda na busca do próximo. Como diferentes modelos podem ter diferentes funções nessa busca iterativa de informações, um desafio é como coordená-los, visto que não há dados rotulados do caminho a ser seguido. Portanto, optou-se por utilizar um modelo treinado por meio de aprendizado por reforço para escolher entre diferentes ferramentas de última geração de forma sequencial até que, ao final, opte por chamar um bloco responsável pela geração da resposta. A nossa arquitetura atingiu F1-score de 19,03, um valor compatível com sistemas iterativos semelhantes da literatura.

Palavras-Chave – Perguntas e respostas, Aprendizado por Reforço, Multi-Hop, Redes Neurais Transformer, Inteligência Artificial.

ABSTRACT

Question Answering is one of the main current topics in natural language processing, as it can be used in many different applications. This project proposes an original architecture to solve open domain and multi-hop questions between texts and tables, using the OTT-QA dataset for validation and training. To answer such questions, it is necessary to search for information in a large corpus by going through several excerpts and tables, as the answer may not be found directly; it is necessary to reason over multiple passages. One of the most common solutions is retrieving information sequentially, where a selected text helps search for the next. As different models can have different functions in this iterative information search, a challenge is how to coordinate them, given that there is no labeled data of the path to be followed. Our architecture uses a model trained through reinforcement learning to choose between different state-of-the-art tools sequentially until, in the end, a block is selected as responsible for generating the answer. Our system achieved an F1-score of 19.03, a value compatible with similar iterative systems in the literature.

Keywords – Question Answering, Reinforcement Learning, Multi-Hop, Transformer Neural Networks, Artificial Intelligence.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathbb{E}$ - Expected value

$\mathcal{L}$ - Loss

t - Time instant in an episode

S; S_t; s - States of the environment.

A; A_t; a - Actions that the agent can perform.

R; R_t - Reward function.

T - State transition.

γ - Discount factor of future rewards.

π - Policy.

π_* - Optimal policy.

G_t - Return.

$V_\pi(s)$ - Value function of a state

$Q_\pi(s, a)$ - Value function of a state-action

δ_t - Temporal difference error

y_i - Target update

w - Weights for the value network

θ - Weights for the policy network

ϵ - hyperparameter that determines amount of exploration for DQN

ϵ - acceptable threshold of divergence for PPO

$A(s, a)$ - Advantage function

IDF - Inverse Document Frequency

$avgdl$ - Average document length

α - Learning rate

q_i - Keywords that are contained by a document

$n(q_i)$ - Number of documents that contain the keyword q_i

$f(q_i, D)$ - Frequency of the keyword in the document D N - total number of documents

k_i and b - both hyperparameters for BM25

a^{rc} - Predicted answer

a^g - Golden answer

τ - selected passage

q - Question

p; p_i; $p_{i,j}$ - Retrieved passage

b_i - Blocks containing the question and retrieved passages

B - Vector with blocks b_i

RI - Reader input

$A1$ - Action retrieve texts

$A2$ - Action retrieve tables

$A3$ - Action generate answer

e_q - Embedding of the question q

e_i - Embedding of the passage p_i

E - Vector with embeddings e_q and e_i

LIST OF ACRONYMS

A2C - Advantage Actor Critic

A3C - Asynchronous Advantage Actor Critic

ARC - AI2 Reasoning Challenge

BART - Bidirectional and Auto-Regressive Transformers

BERT - Bidirectional Encoder Representations from Transformers

BLAB - BLue Amazon Brain

BM25 - Best Matching

CARP - ChAincentric Reasoning and Pre-training framework

CORE - Chain Of REasoning

COS - Chain-of-Skills

DNN - Deep Neural Networks

DPR - Dense Passage Retrieval

DQN - Deep Q-Network

DRL - Deep Reinforcement Learning

DuRePa - Dual Reader-Parser

ELECTRA - Efficiently Learning an Encoder that Classifies Token Replacements Accurately

EM - Exact Match

ETC - Extended Transformer Construction

F1-score - F1 macro-average-score

FiD - Fusion-in-Decoder

FiE - Fusion in Encoder

GAN - Generative Adversarial Network

GPT - Generative Pre-trained Transformer

GPT-2 - Second generation Generative Pre-trained Transformer

GPT-3 - Third generation Generative Pre-trained Transformer

GRU - Gated Recurrent Unit

IDF - Inverse Document Frequency

IR - Information Retrieval

LSTM - Long Short-Term Memory

MDP - Markov Decision Process

MER - Modality-Enhanced Representation

MLM - Masked-Language Modeling

MLP - MultiLayer Perceptron

MMHN - Mixed-Modality Hard Negative

MPNet - Masked and permuted language modeling network

MSCQA - Multi-Step Coarse to Fine Question Answering

MSE - Minimum Square Error

MS MARCO - Microsoft Machine Reading Comprehension

NLP - Natural Language Processing

NQ - Natural Questions Dataset

OTTER - OpenQA Table-Text Retriever

OTT-QA - Open Table-and-Text Question Answering

PLM - Permuted Language Modeling

PPO - Proximal Policy Optimization

PTT5 - Portuguese Text-To-Text Transfer Transformer

QA - Question Answering

QA-pair - Question-Answer Pair

R^3 - Reinforced Ranker-Reader

RL - Reinforcement Learning

RLHF - Reinforcement Learning with Human Feedback

RNN - Recurrent Neural Network

RoBERTa - Robustly Optimized BERT

Seq2seq - Sequence-to-sequence

SOTA - State-Of-The-Art

SQuAD - Stanford Question Answering Dataset

T5 - Text-To-Text Transfer Transformer

T0 - T5 variation for zero shot

TaPas - Table Parser

TD - Temporal Difference

TF - Term Frequency

TRPO	Trust Region Policy Optimization
USP	University of São Paulo

CONTENTS

1 INTRODUCTION

Question Answering (QA) is a field dedicated to developing automated systems that can answer questions posed in natural language. Its history dates back to 1961, with early application noted in Green's work on a QA system about baseball (GREEN et al., 1961). Since then, QA has been widely used to dynamically assist human users by addressing their queries without the need for a human teacher or specialist.

In contemporary applications, QA typically relies on labeled datasets composed of question-answer pairs, which serve as training and evaluation resources for various models. These datasets are essential for objectively assessing and comparing the performance of different models. As modern State-Of-The-Art (SOTA) transformer-based models (VASWANI et al., 2017) require extensive training data, large datasets are of paramount importance. A prominent example of a QA dataset is the Stanford Question Answering Dataset (SQuAD) (RAJPURKAR et al., 2016), where questions are associated with supporting texts containing answers.

In real-world scenarios, QA often encounters questions without an accompanying context, mirroring how humans commonly ask questions. This case is regarded as open setting, wherein the answering model does not just have to comprehend the context and reason about it but has to deduce or find the answer elsewhere. There are two primary methods for knowledge retrieval: encoding all information within the model's parameters (parametric memory) or searching external corpora or databases (non-parametric memory).

A specific type of QA dataset in an open setting format is a multi-hop one. Here, finding the answer involves navigating multiple passages, because it impossible to directly locate the answer within a single passage. Instead, the model must gather information from various sources to reason and deduce the answer. This is important, because when seeking information on a specific topic, one may need to follow a series of interconnected facts and concepts to arrive at a comprehensive answer. Figure 1 shows one example of a multi-hop question.

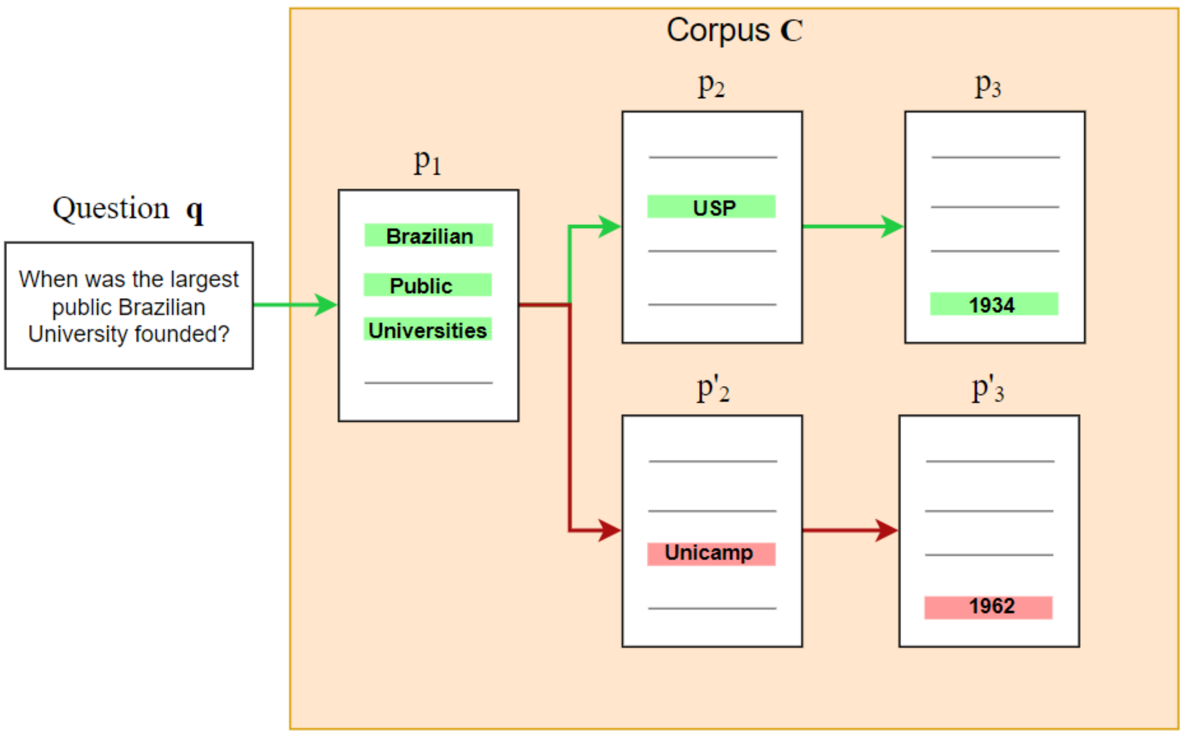

Figure 1: When asking the question q: "When was the largest public Brazilian University founded?" one may require first identifying that the University of São Paulo (USP) is the largest public Brazilian university and then searching for its founding date, 1934. If, for some reason, a system retrieves an incorrect passage p' from the corpus C in any part of the path, it will lead to wrong answer in the end.

While textual data is the most common choice for building corpora, other sources such as graphs (BERANT et al., 2013) and tabular data (CHEN et al., 2020) can be valuable. The latter is particularly important because it complements textual information with structured data, including recurring events, which can be challenging to find in pure textual databases. Consequently, combining textual and tabular data in open domain QA offers significant potential.

This study focuses on addressing challenges in tabular and textual QA, with emphasis on the Open Table-and-Text Question Answering (OTT-QA) dataset (CHEN et al., 2021). It supports open-domain questions, operates in an open setting, and involves multi-hop answering. Due to its complexity, the current best-performing model in its leaderboard only achieves an accuracy of 54.9% (MA et al., 2023), indicating substantial room for improvement. In contrast, simpler datasets like SQuAD have seen transformer-based systems achieve over 90% accuracy, surpassing human performance in 2020 (ZHANG; YANG; ZHAO, 2020).

To tackle open setting and open-domain QA datasets, systems typically employ two components: a retriever and a reader (CHEN et al., 2017; LEWIS et al., 2020b). The retriever fetches relevant passages from a given corpus, which are then concatenated with the question. The reader generates the final answer. This separation is essential since parametric memory has limitations in storing vast knowledge, and the retriever can access extensive databases like Wikipedia. Furthermore, using a retriever can mitigate possible errors due to the information coming from a curated corpus, and it helps in interpretability since it is possible to check the source of the information.

In multi-hop datasets like OTT-QA, a common approach is sequential retrieval, where the system iteratively selects between textual and tabular data retrieval to find the answer. A decision-making model is crucial for determining the optimal retrieval method and the reader to use at each step. Reinforcement Learning (RL) is a suitable choice for this decision-making task, as the sequence of tool/model selection depends on the evolving information gathered (CAÇÃO; COSTA, 2023). Supervised learning is not possible due to a lack of labeled data of the sequence of actions that the agent should make. RL can adapt to the situation using future rewards, like comparing the final system answer to the expected answer.

Given the extensive input space, a function approximator is necessary to reduce computational costs. Deep Neural Networks (DNNs) have shown their effectiveness in learning abstractions and generalizing, thus making Deep Reinforcement Learning (DRL) a fitting choice for the decision-making model.

In summary, we propose an architecture tested on the OTT-QA dataset that features a DRL-based decision maker whose responsibility is to iteratively select among various SOTA tools (primarily retrievers and readers)[1].

1.1 Objectives

This work is based on constructing a multi-hop, open-domain, and open setting QA system using a corpus of texts and tables. Our core system leverages Deep Reinforcement Learning (DRL) for decision-making. The goal is to enable the system to iteratively select the most appropriate tools for sequential document retrieval, ultimately assisting the reader in accurately answering questions.

We highlight the main objectives as follows:

[1]Data and code available at: ⟨https://github.com/MMenonJ/DRL_QA_TT⟩

- Our primary objective is to construct a DRL-based decision-making system that can intelligently select from a predefined set of tools. This system aims to answer open setting multi-hop questions about texts and tables. OTT-QA dataset was selected as the testbed for our system, a challenging benchmark for QA tasks.

- We aim to design a system that is not bound by static tools. It should possess the flexibility to adapt to emerging and superior technologies. Whether a more recent reader or retrieval method becomes available, our system must be able to seamlessly integrate such advancements.

- To ensure broad compatibility and integration with various model-free DRL algorithms, we adhere to the architectural standards set forth by the GYM Python library. By following GYM's conventions, our system can be seamlessly combined with different DRL frameworks (BROCKMAN et al., 2016).

This work represents a step towards building adaptable and sophisticated QA systems that can efficiently handle multimodal complex questions.

1.2 Organization of the dissertation

This dissertation is organized as follows: in Chapter 2 we present some basic concepts concerning Automatic Question Answering, Neural Networks, and Reinforcement Learning. In Chapter 3, some of the main works published in the area are discussed, divided by Open Domain QA, QA over Tables and Text, and the use of Reinforcement Learning on QA.

In Chapter 4, we present our architecture. Chapter 5 contains information about our experiments, and we present the results in Chapter 6. Finally, we conclude our work in Chapter 7.

1.3 Relevant Authored Publications

Throughout the master's program, the author contributed to multiple papers within the domains of QA and RL. In this context, we provide a brief overview of two papers that are particularly relevant to the this work. Additional details about other contributions can be found in Appendix A.

- The author took part in building an architecture called DeePagé, a system designed to answer questions in Portuguese related to Brazil's environment. We constructed a corpus by collecting information from Wikipedia articles and recent news sources on the subject. Retrieving pertinent text from this corpus was made possible through the utilization of the classical retrieval method BM25 (ROBERTSON; ZARAGOZA, 2009). The reader was a fine-tuned PTT5 (Portuguese T5) (CARMO et al., 2020). Our questions were derived from the Probably-Asked Questions dataset (LEWIS et al., 2021), specifically filtered for the Brazilian environment, and then translated from English to Portuguese using the Google Translate API. An article about that effort was published at the "Brazilian Conference on Intelligent Systems" (BRACIS 2021) (CAÇÃO et al., 2021).

  The work demanded an investigation of all the main tools available for retriever and reader modules, which was a crucial experience for tackling open setting QA.

- In a paper presented at the "International Conference on the Computational Processing of Portuguese" (PROPOR 2022) (JOSÉ et al., 2022), we introduced a system to answer questions based on both text and tabular data. The architecture combines transformers trained to answer questions based on textual data with neural models capable of responding to database queries in natural language, often referred to as text-to-SQL interfaces. Additionally, as there has been little effort in developing QA tools for the Portuguese language, we targeted this language. Our proposed architecture selects the appropriate answerer for natural-language questions using a text classifier. We experimented with two text classifiers, a naive Bayes classifier and a neural classifier based on the Portuguese pre-trained transformer network BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020), with the latter demonstrating superior performance.

  Upon classification, questions are directed either to a neural reader-retriever model comprised of a BM25 retriever (ROBERTSON; ZARAGOZA, 2009) and a PTT5 reader (CARMO et al., 2020) (same as DeePagé, as mentioned earlier) or to the text-to-SQL model mRAT-SQL (ARCHANJO; COZMAN, 2021), which employs an mT5 model (XUE et al., 2021) to generate SQL queries for obtaining answers.

  This architecture's construction revealed the question classifier's capability to determine question types with over 99% accuracy. This led to the question: "What about questions requiring reasoning from both text and tables simultaneously?" To address this, we decided to explore the OTT-QA multi-hop dataset, which presents a significantly more complex challenge in bridging text and tabular information.

2 BACKGROUND

This chapter reviews some of the main concepts that are necessary for the understanding of this work. The topics covered are Automatic Question Answering Datasets, Neural Networks, and Reinforcement Learning.

2.1 Automatic Question Answering Datasets

This section delves into key characteristics of Question Answering QA that are presented in this work such as: open-domain, open setting, and multi-hop.

2.1.1 Open-Domain

Closed-domain datasets confine their questions to specific domains, such as PUBMED-QA (JIN et al., 2019) and MIMICSQL (WANG; SHI; REDDY, 2020), which are centered around biomedicine, or Pirá (PASCHOAL et al., 2021), which pertains to the Brazil's maritime coast. In contrast, open-domain datasets encompass questions on a wide array of topics without such constraints.

There are many different open-domain datasets. Some of the more famous are Natural Questions (NQ) (KWIATKOWSKI et al., 2019), and Microsoft Machine Reading Comprehension (MS MARCO) (NGUYEN et al., 2016). The first consists of questions anonymized from search queries of users of the Google Search platform, while MS MARCO is derived from Bing. One of their significant advantages is that they are based on actual questions that people ask, unlike other artificially created datasets. Ultimately, human editors wrote down answers to each question using retrieved passages from Wikipedia for NQ and web pages for MS MARCO.

2.1.2 Open Setting

As there is no consensus on how to define this type of dataset, we use the terminology employed in the OTT-QA paper (CHEN et al., 2021); an open setting QA dataset means that the questions are not accompanied by a text, table, or image that contains the answer. This implies that an open setting question does not require additional context to convey its intended meaning. An open set question, such as "What was the mean temperature of Brazil in March 2019?" contains adequate information within itself to be answered.

One example of a multiple-choice, open-domain and open setting dataset is the AI2 Reasoning Challenge (ARC) (CLARK et al., 2018), with questions about science. It comes with a corpus, and to answer its questions correctly, one must retrieve the correct passages and reason about them.

2.1.3 Multi-Hop

Multi-hop question answering requires a more intricate retrieval process as compared to standard QA. Here it is necessary to traverse through multiple passages before reaching the text that holds the answer. This requires iterative retrieval methods, where information gathered in one text implicitly indicates where to find the next passage, continuing until the answer is located.

HotpotQA is a multi-hop dataset built using Wikipedia hyperlinks between articles (YANG et al., 2018). The questions in this dataset were crafted to exploit multiple connections between texts, ensuring that the answerer must hop between these passages to ultimately recover the final answer. Obtaining the conclusive response to a question is only possible by effectively accumulating knowledge from the intermediary passages. One QA-pair highlighting the sequential nature of a multi-hop QA dataset is shown in Figure 2.

2.1.4 Open Table-and-Text Question Answering

Numerous question-answering datasets and architectures are designed to address questions that require understanding textual information, such as SQuAD, NQ, MSMARCO, HotpotQA, or tabular data, as MIMICSQL exemplifies. However, there has been a scarcity of resources capable of handling both text and tables simultaneously. To bridge this gap, HybridQA was introduced as a closed-domain multi-hop multimodal question-answering dataset (CHEN et al., 2020). In HybridQA, each question is accompanied by a

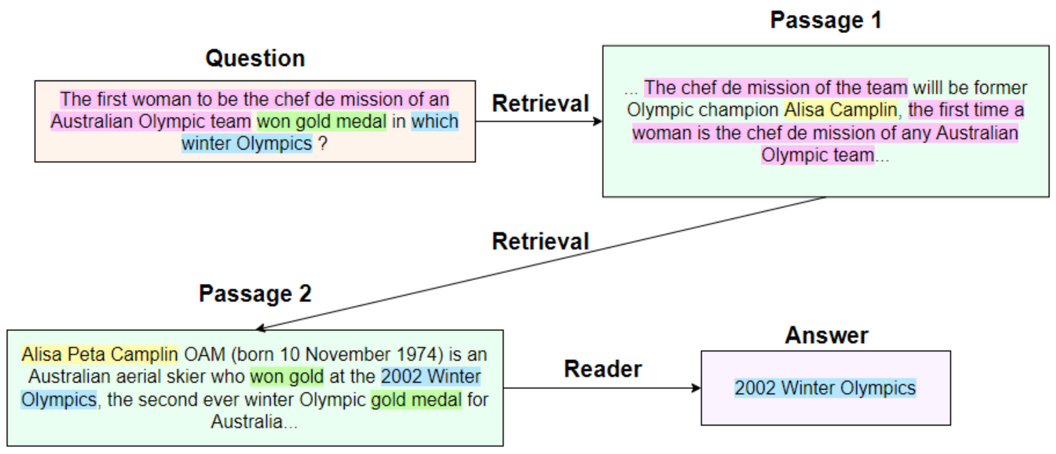
Question
The first woman to be the chef de mission of an Australian Olympic team won gold medal in which winter Olympics ?
Retrieval
Passage 1
... The chef de mission of the team willl be former Olympic champion Alisa Camplin, the first time a woman is the chef de mission of any Australian Olympic team...
Retrieval
Passage 2
Alisa Peta Camplin OAM (born 10 November 1974) is an Australian aerial skier who won gold at the 2002 Winter Olympics, the second ever winter Olympic gold medal for Australia...
Reader
Answer
2002 Winter Olympics

Figure 2: QA-pair example taken from hotptQA (YANG et al., 2018). This question asks about an athlete who won a gold medal in the Winter Olympics, but it is impossible to find this information directly in a single passage. Firstly, it is necessary to get the athlete's name by searching the information (in pink) "the first woman to be the chef de mission of an Australian Olympic team" and the first passage found contains the name (in yellow) "Alisa Camplin". With this data, one can find the second passage that has the information about the gold medal won (in green) and the corresponding Winter Olympics (in blue). Reasoning about the question and both passages, it is possible to generate the answer "2022 Winter Olympics".

table and multiple passages, and generating an answer necessitates the utilization of the combination of several of these sources. The dataset was built in three steps: firstly, 13000 different tables with corresponding passages and appropriate sizes were gathered from Wikipedia (the passages were retrieved using Wikipedia's hyperlink structure). Then, human annotators constructed QA-pairs that required two or more steps over the information in the table and passage. Finally, a de-biasing stage was implemented to mitigate potential annotator biases, such as favoring information from the top of the table or from the beginning of the passages.

Open Table-and-Text Question Answering (OTT-QA) (CHEN et al., 2021) is an English dataset with 45841 QA pairs built over HybridQA. The core aim behind OTT-QA was to create an open-domain, open setting dataset that combined both text and tabular information while simultaneously enhancing the complexity of the task. The first step in the construction of OTT-QA was the "decontextualization" of the questions, meaning that all the answers can be determined from the question alone. Then, new QA pairs were introduced for the development and testing phases to minimize potential bias in the evaluation process. The dataset has an accompanying corpus of tables and texts with answers for every question, so that the QA system can find them. One example of a QA-pair is shown in Figure 3.

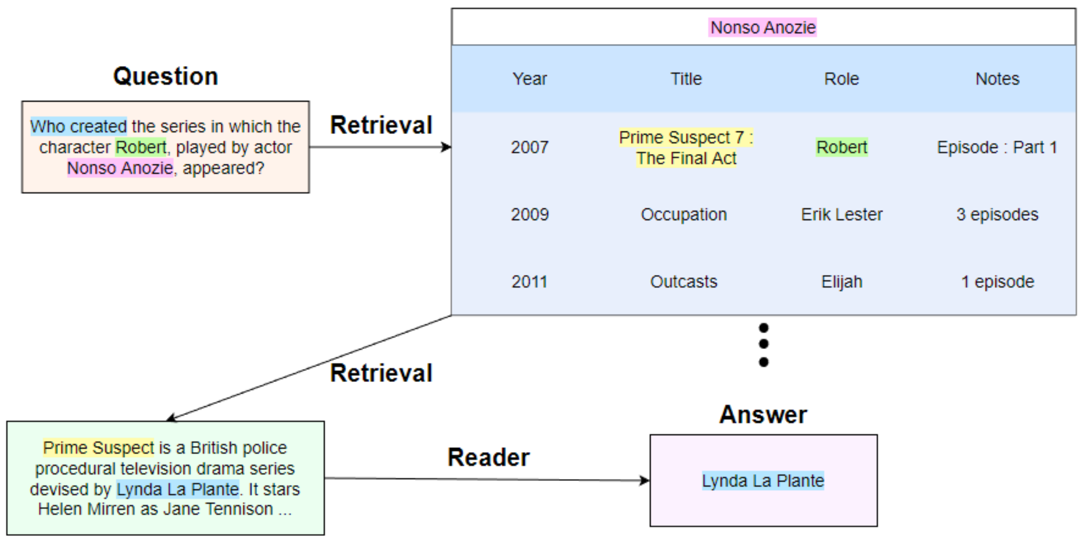
Question
Who created the series in which the character Robert, played by actor Nonso Anozie, appeared?
Retrieval
Nonso Anozie
Year
Title
Role
Notes
2007
Prime Suspect 7 : The Final Act
Robert
Episode : Part 1
2009
Occupation
Erik Lester
3 episodes
2011
Outcasts
Elijah
1 episode
Retrieval
Prime Suspect is a British police procedural television drama series devised by Lynda La Plante. It stars Helen Mirren as Jane Tennison ...
Reader
Answer
Lynda La Plante

Figure 3: QA-pair example taken from OTT-QA (CHEN et al., 2021). This question asks who created the series in which the character Robert appeared. To accurately answer this question, one must locate the relevant table labeled "Nonso Anozie" (highlighted in pink) and identify the cell that contain "Prime Suspect" (highlighted in yellow) by referencing the role "Robert" (highlighted in green). With this information, it is possible to retrieve the associated text, which reveals the creator of the movie was "Lynda La Plante" (highlighted in blue). Importantly, it should be noted that the table and the passage were not directly provided with the question but were instead gathered through information retrieval.

Table 1 shows a comparison between QA datasets mentioned in this section.

2.1.5 Question Answering Evaluation Metrics

This study aims to assess the performance of Table-and-Text Question Answering (QA) systems through the utilization of two widely recognized QA evaluation metrics: Exact Match (EM) and Macro Average F1-score. These metrics, popularized by the SQuAD dataset, are among the most commonly employed benchmarks in the field of Natural Language Processing (NLP). Additionally, they are featured on the leaderboard of OTT-QA, our testbed.

- **Exact Match:** measures the percentage of the predicted answers that are exactly equal to the golden answers.
- **Macro Average F1-score:** measures the overlap between the predicted and the golden answers. The precision is calculated as the ratio of shared words between the golden and prediction answers over the total number of prediction words. The recall is the ratio of shared words over the number of words in the golden answer.

	Open-Domain	**Created using free queries**	**Open Setting**	**Multi-hop**	**Tables**
SQuAD	X				
ARC	X	X	X		
MSMARCO	X	X	X		
NQ	X	X	X		
HotpotQA	X		X	X	
PUBMED-QA					
Pirá			X		
MIMICSQL					X
HybridQA	X			X	X
OTT-QA	X		X	X	X

Table 1: This table presents the main differences between the datasets mentioned. An open-domain dataset can have questions about any subject. NQ and MSMARCO were created using queries from Google and Bing, respectively. In ARC, the questions were made using scientific knowledge. The questions in the other datasets were developed by directly looking at passages. Open setting datasets have questions that do not need an accompanying context and could be answered freely. Multi-hop questions require more than one passage in the corpus to be answered effectively. Finally, tables indicate the datasets that require some form of table manipulation to find the answer.

The F1-score is then determined by

$$F1 = 2 * \frac{precision * recall}{precision + recall}.$$

The final metric is the average value of all the F1-scores for all QA pairs.

Table 2 illustrates the use of both metrics, as well as some of their advantages and limitations. Although one metric can be better depending on the context, they are both deeply flawed because they only match strings and do not consider any meaning.

2.2 Neural Networks

This section covers all the networks used or mentioned in this work.

2.2.1 Long Short-Term Memory Neural Networks (LSTM)

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) (HOCHREITER; SCHMIDHUBER, 1997) that has been widely used in many different

True Answer	**Predicted Answer**	**Exact Match**	**F1-score**	**Notes**
University of São Paulo	University of São Paulo	100	100	The true and predicted answers are the same. Therefore, both metrics gave perfect scores.
Two	2	0	0	This example shows that even if the predicted answer is correct as '2" and "Two" have the same meaning, the metrics can only match strings.
São Paulo	Brazil	0	0	If the question "Where is the University of São Paulo located?" the model could answer Brazil, which is technically correct. However, both metrics returned zero since the strings do not match.
27 people	27	0	66.7	This is one of the advantages of using F1-score instead of EM. The model answered correctly, but EM returns 0 since the strings are not identical. F1-score detects that "27" is present on both strings and assesses that at least part of the answer is correct.
University of Campinas	University of São Paulo	0	57.1	This case shows that the F1-score may be fooled by coinciding words in distinct answers, and for this reason, may overestimate the score.

Table 2: Examples for the metrics Exact Match and F1-score.

applications. LSTMs are engineered to effectively handle sequences of varying lengths by employing an encoder for input sequences and a decoder for generating output. These encoder-decoder networks are also known as sequence-to-sequence (seq2seq) models.

One of the significant advances of LSTM networks is that they suffer relatively fewer problems of vanishing gradients. In a vanilla RNN, the gradient update tends to be small in the earlier layers, resulting in poor training. RNNs may forget the beginning of long input sequences; therefore, they only have short-term memory (BENGIO; SIMARD; FRASCONI, 1994). In contrast, LSTMs use three different gates that control the amount of information passed in and out of each cell, meaning that the network learns what part of the input sequence is important to keep and what can be forgotten.

2.2.2 Gated Recurrent Unit (GRU)

A Gated Recurrent Unit (GRU) (CHO et al., 2014) is a type of RNN that is very similar to LSTM but has two gates (reset and update gates) instead of the three gates. As it has a relatively straightforward structure, it is less computationally expensive than LSTMs, and, even with the same number of parameters, it can outperform LSTMs on some datasets in terms of convergence time, parameter updates, and generalization (CHUNG et al., 2014).

2.2.3 Transformer Neural Networks

Transformer neural networks, presented by Vaswani (VASWANI et al., 2017), are another milestone in sequence-to-sequence neural network architectures. These networks have seen widespread applications across various fields, with a significant impact in NLP. Consider the SQuAD dataset mentioned before (RAJPURKAR et al., 2016): transformers have achieved state-of-the-art performance, with accuracy rates surging from around 70% in 2018 to over 90% in 2020, as reported in Zhang, Yang and Zhao (2020) work, thus surpassing human performance.

Unlike RNNs, transformer networks utilize a self-attention mechanism, which differentially weighs the significance of each component in the input sequence. This addresses a key limitation of sequential processing, as it allows transformer networks to consider all components simultaneously, preventing the network from forgetting distant elements. Additionally, this self-attention mechanism can be parallelized, leading to significantly faster computation.

Transformer networks follow a two-step process akin to language learning, beginning with self-supervised learning techniques in pre-training, followed by fine-tuning for specific tasks. A common pre-training strategy involves Masked-Language Modeling (MLM), wherein the network learns to predict masked words in a given context. For example, for the sentence "I finished [MASK] a book yesterday", the network is trained to predict the masked word, which, in this case, might be "reading".

While there are many different pre-trained transformer networks, we describe here the ones that are the most relevant for this dissertation:

Encoder models:

- **BERT:** One of the most famous pre-trained transformer neural networks is the Bidirectional Encoder Representations from Transformers (BERT) (DEVLIN et al., 2019). BERT is an encoder network that reads the entire input sequence at once and not sequentially from just left to right. This property allows a better awareness of the whole surroundings of a word, thus improving context awareness.

- **RoBERTa:** Robustly Optimized BERT (RoBERTa) (LIU et al., 2019) is an improved version of BERT with key differences: it underwent more extensive training with larger batch size, used more data, employed longer sequences in the Masked Language Model (MLM) task, and introduced dynamic MLM, which randomizes masked data preprocessing for more varied training.

- **ELECTRA:** Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA) (CLARK et al., 2020) is an encoder transformer network that is not trained using MLM token prediction but rather by training a discriminative model that predicts whether a generator replaced each token. This means that instead of passing the sentence "I finished [MASK] a book yesterday" and requesting what word would fit, it would send a "corrupted" like "I finished *watching* a book yesterday" and requiring the model to correctly predicting that "*watching*" was not the original word in the corpus. This adversarial training style resembles the discriminator of a Generative Adversarial Network (GAN) (GOODFELLOW et al., 2014).

- **MPNET:** Masked and permuted language modeling network (MPNET) is another encoder network proposed by Song et al. (2020) that proposes a novel pre-training method. It is based on BERT, however, since the MLM training does not consider dependency between tokens, it is pre-trained incorporating ideas from the Permuted

Language Modeling (PLM) (YANG et al., 2019).

- **ETC:** One of the main difficulties of using transformer neural networks is the quadratic computational complexity with respect to the input size. With this in mind, Ainslie et al. (2020) created the encoder model Extended Transformer Construction (ETC) by introducing the *global-local attention*, which divides the input sentence into global input and the long input. This technique reduces the complexity and allows the use of bigger inputs.

- **Longformer:** Similar to ETC, Longformer (BELTAGY; PETERS; COHAN, 2020) is an encoder transformers network (there is also a sequence-to-sequence Longformer variant, but in the context of this manuscript, we will refer to Longformer as the encoder version) that uses a different strategy attention strategy to allow more extensive input texts. Instead of using the standard self-attention, the authors proposed an attention mechanism that scales linearly with sequence length that combines local windowed attention with task-specific global attention, enabling the processing of longer input texts efficiently.

Decoder models:

- **GPT-2:** Second generation Generative Pre-trained Transformer (GPT-2) (RADFORD et al., 2019) is an unidirectional decoder network that can perform many different text generation tasks. It has approximately ten times more parameters than its predecessor GPT-1 (RADFORD et al., 2018), and trained in a much larger dataset than his predecessor.

- **GPT-3:** (BROWN et al., 2020) is the third generation of GPT models and is a decoder transformer with approximately 175 billion parameters. The authors also evaluated it in several tasks using only few-shot learning: in this setting, GPT-3 achieved SOTA performance even without direct fine-tuning.

Encoder-Decoder models:

- **BART:** (Bidirectional and Auto-Regressive Transformers) (LEWIS et al., 2020a) is an encoder-decoder network architecture with a bidirectional encoder (like BERT) and a left-to-right decoder (like GPT). It is pre-trained using denoising techniques: a noised/corrupted input sentence is fed to the model, and it has to reconstruct the original text. Various corruption methods were tested: token masking (as in BERT), token deletion, text infilling, sentence permutation, and document rotation.

- **T5:** Text-To-Text Transfer Transformer (T5) (RAFFEL et al., 2020) is also an encoder-decoder transformers network that is pre-trained in multiple tasks in a text-to-text setup. It can work in different tasks with the same network by just changing the prefix, like summarization and translation.

- **T0** (SANH et al., 2022) is a model based on the architecture of T5; however, it distinguishes itself by being designed explicitly for zero-shot tasks. The researchers conducting this study have demonstrated that through explicit training on a diverse range of tasks, a technique known as multitask learning, the model can be induced to exhibit zero-shot generalization. In other words, it can perform tasks for which it was not originally trained.

The most widely used Python library for working with transformer networks and their applications is the *Transformers* library maintained by the HuggingFace team.[1] This library provides APIs to download SOTA transformers models in various languages and contains code for training networks. It offers a wide range of off-the-shelf models for different applications, including text, images, and audio, and including all the models used in this work.

2.3 Reinforcement Learning

Reinforcement Learning (RL) is a key machine learning paradigm, alongside supervised and unsupervised learning, in which an agent takes actions to maximize cumulative rewards within an environment. Unlike supervised learning, there is no need for labeled data; instead, in RL, the only intervention is assigning a scalar reward value, either positive or negative, for each interaction. The concept of learning in RL comes from the agent's desire to explore the environment in a way that generalizes into intelligent behavior. This process is illustrated in Figure 4.

The agent-environment interaction loop can be described by a Markov Decision Process (MDP), defined as a tuple (S, A, R, T, γ) (SUTTON; BARTO, 2018), where:

- S: represents the states of the environment in which the agent can be placed.

- A: corresponds to the actions that the agent can perform.

- R: is the reward function.

[1]Available here: ⟨https://huggingface.co/docs/transformers/index⟩

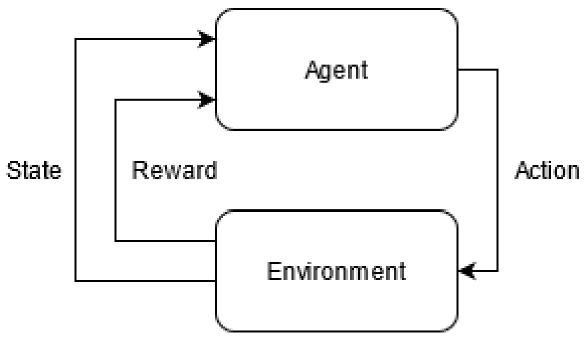

Figure 4: The basic loop of Reinforcement Learning: an agent interacts with the environment, obtains a reward, and changes the environment as a result of the action. Adaptated from Sutton and Barto (2018).

- T: is the state transition.
- γ: is the discount factor of future rewards. It is a scalar value between zero and one that determines the importance of future rewards.

The agent's objective is to find an optimal policy, denoted as π_*, which prescribes the best action for each state. Optimality is defined as maximization of the expected value of the return function, which sums up the discounted future rewards. For a timestep t, and the reward R_i, the return is defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{2.1}$$

The expected return for a given policy is captured by the value function. For a state s, it is expressed as

$$V_\pi(s) = \mathbb{E}[G_t | S_t = s], \tag{2.2}$$

or, for the state-action pair,

$$Q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]. \tag{2.3}$$

Temporal Difference (TD) learning is a key concept in Reinforcement Learning. It enables agents to update their value estimates continually as they explore the environment. They take into account the immediate rewards and the expected value of the subsequent state instead of the return, incorporating the bootstrapping concepts. The TD error quantifies the difference between the predicted value of a state or state-action pair and the observed reward. For a state-value function (V), the TD error formula is as follows:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t), \tag{2.4}$$

and the agent adjusts its value estimates by reducing the TD error through learning and exploration.

When compared to other types of learning, RL can be thought as being based on trial and error. This allows the learning process to be carried out somewhat autonomously and without direct supervision. As the discounted rewards obtained are accumulated over an entire episode, the agent is trained to consider future states; actions with low rewards in the present may bring higher rewards in the future, and vice versa. The capacity to learn through exploration and experimentation endows the agent with adaptability, enabling it to adjust to novel tasks and excel in unforeseen scenarios, extending beyond the scope originally envisaged by the programmer.

However, tabular RL has faced difficulties in scaling to more complex tasks. The "curse of dimensionality", coined by Bellman (BELLMAN, 1957), highlights the computational difficulties that arise due to high-dimensional input data. To mitigate these challenges, various function approximators have been explored to replace tabular storage and reduce the computational costs. Deep Reinforcement Learning (DRL) utilizes deep neural networks as such function approximator.

2.3.1 Deep Reinforcement Learning Algorithms

Two DRL algorithms were used in our work. We explain them below.

2.3.1.1 Deep-Q-Network (DQN)

One of the most famous RL algorithms is Q-Learning proposed by Watkins and Dayan (1992). Its main idea is to make the update for the value of the state-action pair by approximating π_* taking into consideration the maximum value of all possible future actions and not what the agent actually took as action. The update is done as follows:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]. \tag{2.5}$$

To balance exploration and exploitation, Q-Learning employs the ϵ-greedy policy. This policy is determined by the probability of action a:

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{if } a = arg\max_{a'} Q(a', s) \\ + \frac{\epsilon}{|\mathcal{A}(s)|} & \text{for every other } a \end{cases} \tag{2.6}$$

where ϵ is an hyperparameter with values between 0 and 1 and $|\mathcal{A}(s)|$ is the number of actions for a given state s. The main idea is that the bigger ϵ, the more the agent will explore and, therefore, will exploit less what has already been learned.

Mnih et al. (2013) built upon Q-Learning to create Deep Q-network (DQN) using a neural network with weights w as a function approximator. However, transitioning from a Q-table to a neural network presented new challenges, primarily related to stability. The first challenge that was addressed was that training is sequential, following a particular trajectory, which may lead to forgetting of the others. To solve this issue, they utilized an experience replay, which stored past transitions. Instead of training just with the last transition, the training occurred by randomly sampling a minibatch of stored transitions.

In 2015, further improvements were made to the DQN by Mnih et al. (2015) to solve another issue: the training target depends on the network that is being trained, leading to instability. They introduced the concept of a target network with separate weights w', a copy of the Q-network that is only updated periodically. The updates of the Q-network followed the target network, reducing instability. The target update is given by:

$$y_i = r_i + \gamma \max_{a'} Q(s'_i, a', w'), \tag{2.7}$$

and the loss to make the updates is the Mean Squared Error (MSE) with respect to the target, given a minibatch with size N:

$$\mathcal{L}(w) = \frac{1}{N} \sum_i (Q(s_i, a_i, w) - y_i)^2. \tag{2.8}$$

Using the loss above, the updates on the network follow the gradient descent optimization. The pseudo-code for DQN is shown in Algorithm 1.

2.3.1.2 Proximal Policy Optimization (PPO)

Unlike value-based methods that optimize a value network and generalize a policy from it, policy gradient methods optimize a policy network directly. Williams (1992) described the REINFORCE method, which applies gradient ascent optimization to the policy network in the direction that maximizes the return G_t:

$$\mathcal{L}(\theta) = G_t \frac{\pi(a|s, \theta)}{\pi(a|s, \theta_{\text{old}})}, \tag{2.9}$$

where $\pi(a|s, \theta)$ indicate the policy given the weights θ and state s.

Actor-critic methods combine both policy gradient methods with value-based methods using two neural networks. Mnih et al. (2016) developed the Asynchronous Advantage

Algorithm 1: Deep Q-Network (DQN)

```
Environment with states $S$, actions $A$, and reward function $R$
Initialize Q-network with random weights $w$
Initialize Replay memory $D$
Initialize Target Q-network with weights $w'$ Parameters: $\epsilon$-greedy exploration,
  learning rate $\alpha$, discount factor $\gamma$, target network update rate $C$
while not converged do
    Observe current state $s$
    Choose action $a$ using $\epsilon$-greedy policy based on $Q(s, a, w)$
    Execute action $a$, observe reward $r$ and new state $s'$
    Store transition $(s, a, r, s')$ in $D$
    Sample mini-batch from $D$
    Compute target values: $y_i = r_i + \gamma \max_{a'} Q'(s'_i, a', w')$
    Update the weights $w$ of the Q-network with gradient descent on:
      $\mathcal{L}(w) = \frac{1}{N} \sum_i (Q(s_i, a_i, w) - y_i)^2$
    Every $C$ steps, update the target Q-network: $w' \leftarrow w$
end while
```

Actor-Critic (A3C) algorithm, that uses an *advantage function* in place of G_t, defined as:

$$A(s, a) = R_{t+1} + \gamma V(S_{t+1}, w) - V(S_t, w). \tag{2.10}$$

The use of the advantage, instead of the return, has two benefits. First, bootstrapping can be more effective. Second, instead of the absolute value of a state-action pair, the advantage can precisely represent how one action compares to just following the policy.

Trust Region Policy Optimization (TRPO), presented by Schulman et al. (2015), makes similar updates as A3C to the policy network but adds constraints to limit the size of the updates. This is done to improve stability, as a single bad step or overshoot will not have an excessive impact on the network.

However, since TRPO requires expensive computation, Proximal Policy Optimization (PPO), proposed by Schulman et al. (2017), serves as a simpler alternative for limiting the updates:

$$\mathcal{L}(\theta) = \min \left(\frac{\pi(a|s, \theta)}{\pi(a|s, \theta_{\text{old}})} A(s, a), \text{clip} \left(\frac{\pi(a|s, \theta)}{\pi(a|s, \theta_{\text{old}})}, 1 - \epsilon, 1 + \epsilon \right) A(s, a) \right). \tag{2.11}$$

Here, ϵ is a small hyperparameter that limits how far the new policy is allowed to deviate from the old one. Note that this ϵ is not the same as the one used in Q-Learning or DQN. The main idea behind PPO is to prevent significant updates, as $\mathcal{L}(\theta)$ is clipped on both the positive and negative boundaries.

The pseudo-code for PPO is shown in Algorithm 2.

Algorithm 2: Proximal Policy Optimization (PPO)

```
Environment with states $S$, actions $A$, and reward function $R$
Initialize V-network with random weights $w$
Initialize Policy network with weights $\theta$ Parameters: Clip parameter $\epsilon$, learning
  rate $\alpha$, discount factor $\gamma$
while not converged do
    Collect a batch of trajectories using policy $\pi_\theta$
    Compute advantages $A(s,a) = R_{t+1} + \gamma V(s', w) - V(s, w)$
    for each epoch do
        Compute surrogate objective:
        $$\mathcal{L}(\theta) = \min\left(\frac{\pi(a|s,\theta)}{\pi(a|s,\theta_{\text{old}})}A(s,a), \text{clip}\left(\frac{\pi(a|s,\theta)}{\pi(a|s,\theta_{\text{old}})}, 1-\epsilon, 1+\epsilon\right)A(s,a)\right)$$
        Update the weights $\theta$ policy network with gradient ascent on $\mathcal{L}(\theta)$
        Update value function $V(s,w)$ using mean squared error
    end for
    Update old policy: $\theta_{\text{old}} \leftarrow \theta$
end while
```

While DQN may be more sample-efficient than PPO in most cases due to its use of batching and experience replay (which store past experiences and make constant updates based on that), PPO is usually more stable and easier to converge.

2.3.2 Available Libraries

DRL applications are divided into two systems: the agent and the environment. The first performs actions and learns to maximize the discounted cumulative rewards by interacting with the latter.

A prevalent approach for implementing the environment is by constructing a gym-like environment. This is achieved through the utilization of OpenAI Gym (BROCKMAN et al., 2016)[2], a comprehensive open-source Python library. OpenAI Gym provides a robust framework, including valuable wrappers and standardization, facilitating the development and experimentation of various reinforcement learning algorithms.

The primary motivation for adopting a gym-like environment is to ensure compatibility with various Python libraries equipped with built-in agents. In this context, our emphasis is on Stable-Baselines3 (RAFFIN et al., 2021) [3], an open-source framework that offers efficient implementations of state-of-the-art R) algorithms. Notable examples in-

[2]https://gym.openai.com/

[3]https://stable-baselines3.readthedocs.io/en/master/

clude DQN, PPO, and the Advantage Actor Critic (A2C) (MNIH et al., 2016), which is a synchronous variant of the A3C.

3 RELATED WORK

This chapter discusses important previous works, starting with open-domain and open setting QA and discussing relevant modules. We then review Tables and Texts Question Answering and noteworthy publications concerning OTT-QA. To conclude, we examine proposals using RL to tackle QA challenges.

3.1 Open-Domain and Open Setting Question Answering

One of the most interesting tasks in automatic QA is the open-domain and open setting task. In this case, the system only receives a question and has to deduce or search in a corpus for the answer. Question Answering systems usually resort to two blocks: a retriever and a reader (as illustrated in Figure 5). The retriever's role is to search the corpus for relevant texts and passages while the reader processes this information to generate an appropriate answer. Some proposals employ neural transformers in both the retriever and the reader (LEWIS et al., 2020b; GUU et al., 2020), while others opt for keyword-based models for the retriever (CHEN et al., 2017; CAÇÃO et al., 2021).

3.1.1 Retriever

One of the most widely used keyword-based retriever models in QA is BM25 (ROBERTSON; ZARAGOZA, 2009), which ranks the passages based on matching the words on the passages of the corpus to the question. BM25 is an improvement over TF-IDF (Term Frequency–Inverse Document Fequency), as it incorporates document length normalization. This means that a smaller document will be ranked higher than a lengthier document with the same matched words. It ranks the pages using the formula of the score, being D the Document and Q the question:

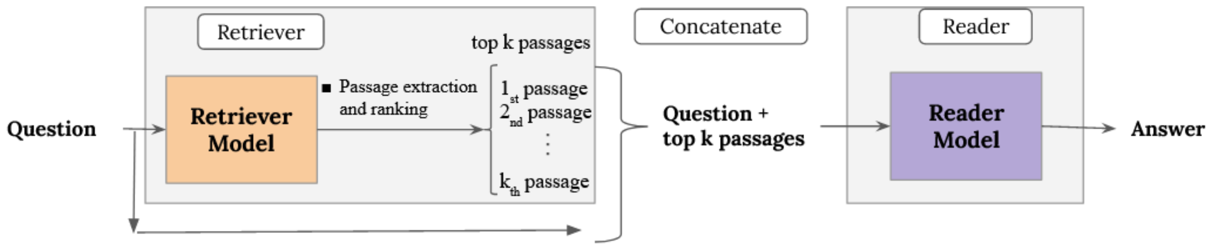

Figure 5: Neural Question Answering Reasoner using the retriever-reader architecture. Source: authors, taken from José et al. (2022).

$$score(D,q) = \sum_{i=1}^{n} IDF(q_i) * \frac{f(q_i, D) * (k_i + 1)}{f(q_i, D) + k_1 * (1 - b + b * \frac{|D|}{avgdl})}, \tag{3.1}$$

where $avgdl$ is the average document length. IDF is the inverse document frequency of the keyword compared to the entire corpus calculated by the formula:

$$IDF(q_i) = \ln\left(\frac{N - n(q_i + 0.5)}{n(q_i) + 0.5} + 1\right), \tag{3.2}$$

where:

- q_i: is any keyword that the document contains.
- $n(q_i)$: is the number of documents that contain the keyword q_i.
- $f(q_i, D)$: is the frequency of the keyword in the document D
- N: is the total number of documents.
- k_i and b: are both hyperparameters.

However, relying solely on keyword matching introduces two challenges: a lack of contextual understanding and the inability to identify synonymous words. For the first case, imagine the phrase "I did not eat the cake, I only ate the pizza" in contrast to "I did not eat the pizza, I only ate the cake". Both have opposite meanings, but for BM25, they are a perfect match. For the second problem, some words may not match, even though they are synonyms like "terrible" and "horrible".

Dense Passage Retrieval (DPR) (KARPUKHIN et al., 2020) can be described as a cornerstone of transformers neural retrievers for text in QA. It was developed to diminish problems indicated in the previous paragraph, as DPR's idea is to rank passages based on their meaning rather than just keywords. The architecture is depicted in Figure 6. DPR consists of two BERT models, one for the passages and one for the questions. Both

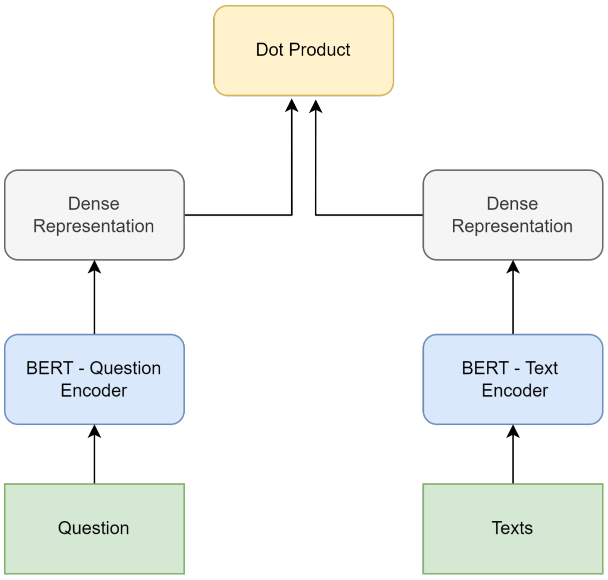

Figure 6: Architecture of the "Dense Passage Retrieval" by Karpukhin et al. (2020).

networks are trained jointly to approximate the dense vector representation of the golden passage (passage which contains the answer) and the question, using the dot product as a similarity function. By contrast, there is also the training using hard negative passages (passages that does not contain the answer) but diminishing the proximity. The intuition is if the passage contains the answer, the closer the resulting dense vector should be to the question.

The training of the networks uses the loss function L for a question q_i, golden passage p_i^+ and hard negative passages $p_{i,n}^-$:

$$\mathcal{L}(q_i, p_i^+, p_{i,1}^-, ..., p_{i,n}^-) = -log \frac{e^{sim(q_i,p_i^+)}}{e^{sim(q_i,p_i^+)} + \sum_{j=1}^{n} e^{sim(q_i,p_{i,j}^-)}}. \tag{3.3}$$

It is noteworthy that DPR often surpasses BM25 in terms of performance. While BM25 shines when answers precisely match the keywords in the question, DPR offers advantages in scenarios where exact keyword matching is not the sole determinant of success.

One potential drawback of DPR is its strong association with Wikipedia-based training datasets. Fine-tuning is often necessary for diverse domains, incurring computational costs. In contrast, BM25 does not require such domain-specific adjustments, making it a more straightforward choice for certain applications.

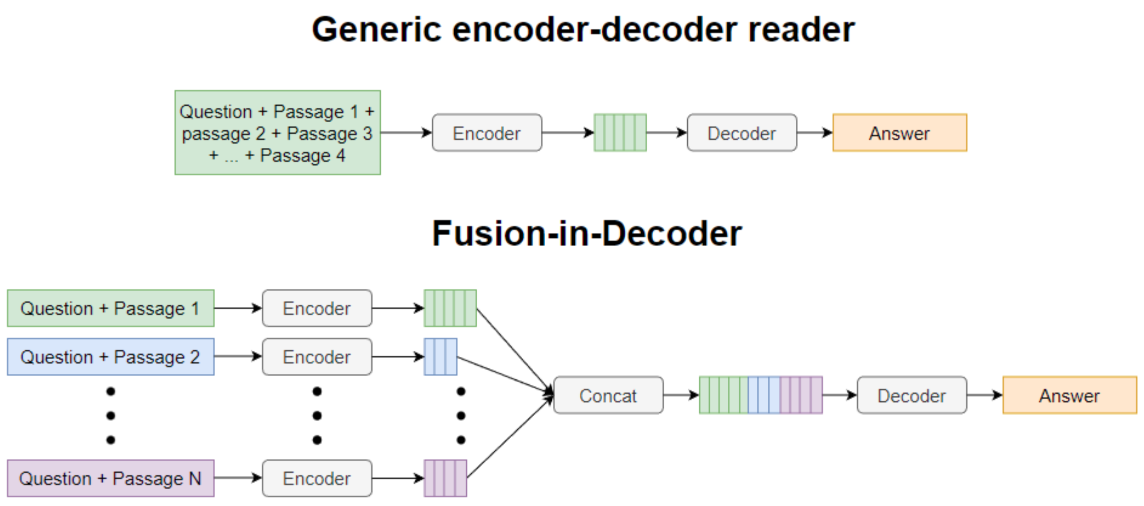

Figure 7: Comparison of a generic encoder-decoder transformers reader and Fusion-in-Decoder by Izacard and Grave (2021). The former concatenates the text of the question and all passages, while the latter processes each passage with the question individually in the encoder first and then concatenates the resulting vectors for the decoder.

3.1.2 Reader

State-of-the-art reader models are typically fine-tuned neural transformer networks. These models usually receive a concatenated input of the question and the passages retrieved by the retriever and are designed to generate answers. Some architectures utilize an encoder model for the reader, like BERT (GUU et al., 2020), in which the predicted answer is a substring of the retrieved passages. Others utilize encoder-decoder models treating the problem as a text-to-text task, like T5(KHASHABI et al., 2020), PTT5 or Portuguese T5 (CAÇÃO et al., 2021), and BART (LEWIS et al., 2020b).

Another reader that is getting attention is the Fusion-in-Decoder (IZACARD; GRAVE, 2021). Instead of simply concatenating the question with all retrieved passages into a single text and using an encoder-decoder network directly, this system processes each passage independently in the encoder and then concatenates the resulting vectors as input to the decoder, as shown in Figure 7. The main advantage of this approach is the ability to process a larger number of documents efficiently. The self-attention mechanism is applied to each passage independently, reducing computational costs and enabling more focused information gathering without significantly losing critical details.

Similarly, Fusion in Encoder (FiE) (KEDIA; ZAIDI; LEE, 2022) is an architecture that allows the fusion of information from multiple passages in the encoder rather than the decoder. FiE creates a global representation and performs cross-sample attention over all tokens across different samples. The authors have also proposed an alternative approach for calculating answer span probability to consider all passages effectively. This design enables the incorporation of multiple passages without dealing with the challenges of handling extensive concatenated inputs. FiE has achieved state-of-the-art performance on datasets such as NQ, even with a comparatively smaller and non-generative model.

3.2 Table and Text Question Answering

As previously discussed, accessing a wide range of databases and tables can significantly enhance a model's performance. An illustrative example is found in the landmark work on the Watson engine by Ferrucci et al. (2010), where a question is input to an ensemble of predictors. Their confidence scores play a pivotal role in determining which answer, if any, to return.

To solve the HybridQA dataset, its authors proposed Hybrider (CHEN et al., 2020). The architecture consists of two main phases: Linking and Reasoning. In the Linking phase, Hybrider links questions to related cells from two sources - explicitly mentioned cells in the question and implicitly mentioned cells via hyperlinked passages. The Reasoning phase models multi-hop reasoning in the table and passage using three stages: Ranking, Hop, and Reading Comprehension. The architecture leverages neural networks and a BERT encoder to make predictions and select relevant cells, hop to neighboring cells, and extract answers.

In Chapter 2, we explored many transformer models that deal with textual data. Table Parser (TaPas) (HERZIG et al., 2020) is another transformer neural network, but designed and pre-trained using tables and related text segments from Wikipedia. TaPas has been evaluated across a wide range of tasks involving tables and text; notably, in the HybridQA dataset, it has demonstrated one of the best results to date.

3.2.1 Architectures for OTT-QA

The authors of OTT-QA (CHEN et al., 2021) presented a variety of solutions, which we will now discuss. The first solution, known as "BM25-HYBRIDER" serves as a baseline for OTT-QA. It involves retrieving the top 1, 2, 3, or 4 texts or tables from the corpus using the BM25 algorithm. These combinations of retrieved texts or tables (from 1 to 4) are then input to the hybrider reader mentioned in the previous section, and the answer selected was the one with the highest confidence interval. However, this approach is somewhat simplistic as it does not consider that the reader was originally trained for the closed setting of HybridQA. Furthermore, it does not actively seek passages addressing the multi-hop problem since the query is only based on the question.

The second solution we will discuss is the "Iterative-Retrieval + Cross-Block Reader", which features two iterative retriever variations:

- Sparse: This variant employs BM25 to retrieve 10 text passages and 10 table segments. For each retrieved text passage, it further retrieves 5 table segments using the concatenation of the question with each retrieved text passage, and vice-versa for each table segment.
- Dense: In this approach, a model resembling DPR is used for retrieval. It starts by searching for 8 text or table passages. For each of these 8 passages, it searches for 4 more passages, and for each of those 8 blocks (concatenation of the question with 5 passages), it conducts an additional search for 2 passages.

The retrieved passages are then fed to the Cross-Block Reader, which is a finetuned ETC, generating the answer.

The authors also introduced a novel solution called the "Fusion Retriever + Cross-Block Reader" which demonstrated improved performance compared to the previous one. This approach first combines tables and text into blocks by linking them with BM25 and enhancing queries through GPT-2. The Fusion Retriever is a dual-encoder setup, based on DPR, that retrieves k blocks based on their similarity to the question while maintaining the same Cross-Block Reader to generate the answer.

Other techniques have also been tested in OTT-QA, like the work of Li et al. (2021), which proposes a Retriever-Reader system and a Joint Reranking model known as DuRePa. The Retriever component utilizes BM25 to select 100 textual and 100 table passages relevant to the question. These passages are then fed into the Joint Reranking model, which employs a BERT-based architecture to assign scores to each passage, ultimately selecting the top 50 passages. The Reader component, based on a Fusion-in-Decoder approach with T5, determines whether the answer relies on a table, in which case it generates a SQL query, or if it is text-based, the model generates the final response.

It is worth noting that while DuRePa has demonstrated impressive results in other datasets, it falls short of achieving SOTA performance in the context of OTT-QA. This limitation is attributed to its non-iterative retrieval approach, which solely relies on question keywords and lacks multi-hop capability. Furthermore, the Reader is primarily trained to locate answers within either textual or tabular data and does not effectively link both information sources. Nevertheless, the application of Joint Reranking can enhance the performance of a retrieve+reader model in OTT-QA.

The ChAincentric Reasoning and Pre-training framework (CARP) (ZHONG et al., 2022) incorporates an innovative approach for building text and passage blocks by utilizing early fusion techniques. This process involves employing a BERT model to establish entity links known as BLINK (PARTALIDOU; CHRISTOU; TSOUMAKAS, 2022), while a DPR, trained from RoBERTa, serves as the retriever for searching these blocks. Subsequently, CARP employs a Hybrid chain extractor, which is based on BART, to identify and extract links of knowledge. CARP's reader, based on Longformer, then leverages the assembled text blocks and knowledge chains to generate the answer.

OpenQA Table-Text Retriever (OTTER) (HUANG et al., 2022) is another QA system tested on the OTT-QA dataset that utilizes an early fusion method equal to CARP, and the Cross-Block Reader from OTT-QA's original paper. However, it makes significant improvements in the retriever. First, it enhances mixed-modality representation learning through modality-enhanced representation (MER) and mixed-modality hard negative sampling (MMHN). MER enriches the semantics by incorporating fine-grained representations of both tabular and textual data. MMHN, on the other hand, generates challenging hard negatives by substituting partial information within tables or texts to encourage better discrimination of relevant evidence. To overcome the data sparsity problem, the authors implement retrieval-centric mixed-modality synthetic pre-training. This involves constructing a large-scale synthesized corpus through mining relevant table-text pairs and generating pseudo questions using a BART model.

The CORE (Chain Of REasoning) QA system employs a retriever-reader architecture but includes two intermediary components, the Linker and the Chainer (MA et al., 2022). The retriever is built on DPR and is responsible for retrieving 100 tables from the corpus. Subsequently, a Linker system based on BERT models retrieves passages from the corpus based on the rows obtained from the retrieved tables. However, since the linker can provide an excessive amount of information for the reader to process, the Chainer's role is to select the top 50 chains (table row and text passage), which consist of a table row and a corresponding text passage. The Chainer accomplishes this task by employing a T0 network in a zero-shot manner. It assesses the probabilities of the model generating a question based on specific text passages and table rows (the higher the probability of that question being generated, the higher the rank of the chain). Finally, to answer the question, the system uses a FiD model, which relies on the information curated by the Chainer.

Chain-of-Skills (COS) (MA et al., 2023) is, at the moment of writing, the architecture that achieves the best performance on OTT-QA. Drawing inspiration from the sparse Transformer (FEDUS; ZOPH; SHAZEER, 2022), the authors introduced a modularization approach that facilitates efficient multi-task training across various Open-Domain QA datasets. The tasks tackled include single retrieval, expanded query retrieval, entity span proposal, entity linking, and reranking. In the inference phase, the system initiates by identifying 100 tables through single retrieval. These tables are subsequently divided into rows, and the reranking process selects the top 200 rows. For each row, the system collects 10 passages from the expanded query retrieval and one passage from linked entities. The system then leverages the same Chainer used in CORE to select the top 100 passages, which are subsequently input into a fine-tuned FiE reader tailored for OTT-QA.

Table 3 provides a comprehensive list with the performances of each systems in OTT-QA dataset.

It is also worth highlighting the work "Multi-modal Retrieval of Tables and Texts Using Tri-encoder Models" (KOSTIĆ; RISCH; MÖLLER, 2021), which builds upon the foundation laid by DPR to develop a neural retrieval system capable of efficiently retrieving both tables and texts. They achieved state-of-the-art results in retriever performance across table and text datasets. The training utilized various datasets, and, for OTT-QA, the authors only utilized the golden tables, since the golden texts passages are not available for the public. The architecture employed in this work leverages a Tri-encoder design, where three small-BERT networks are utilized for encoding questions, texts, and tables individually, as visually depicted in Figure 8. This research underscores the significance of using distinct neural networks for handling different types of knowledge bases, leading to significantly improved retrieval results. As a simplification, we refer this retriever as Tri-encoder.

3.3 Reinforcement Learning in Question Answering

One of the first proposals to successfully combine DRL and QA is the architecture Reinforced Ranker-Reader (R^3) (WANG et al., 2018) in 2017. The primary concept revolves around the joint training of both the retriever (referred to as the ranker in the paper for the purpose of ranking passages) and the reader, using the DRL algorithm REINFORCE

Model	**Reference**	**Dev-EM**	**Dev-F1**	**Test-EM**	**Test-F1**
COS	Ma et al. (2023)	**56.9**	**63.2**	**54.9**	**61.5**
CORE	Ma et al. (2022)	49.0	55.7	47.3	54.1
OTTeR	Huang et al. (2022)	37.1	42.8	37.3	43.1
CARP	Zhong et al. (2022)	33.2	38.6	32.5	38.5
Fusion-Retrieval + Cross-Block Reader	Chen et al. (2021)	28.1	32.5	27.2	31.5
Iterative-Retrieval (sparse) + Cross-Block Reader	Chen et al. (2021)	17.1	20.7	16.9	20.9
Iterative-Retrieval (dense) + Cross-Block Reader	Chen et al. (2021)	14.4	18.5	-	-
Dual Reader-Parser	Li et al. (2021)	15.8	-	-	-
BM25-Hybrider	Chen et al. (2021)	10.3	13.0	9.7	12.8

Table 3: Adapted OTT-QA leaderboard presented at ⟨https://github.com/wenhuchen/OTT-QA⟩ at the time of writing this document.

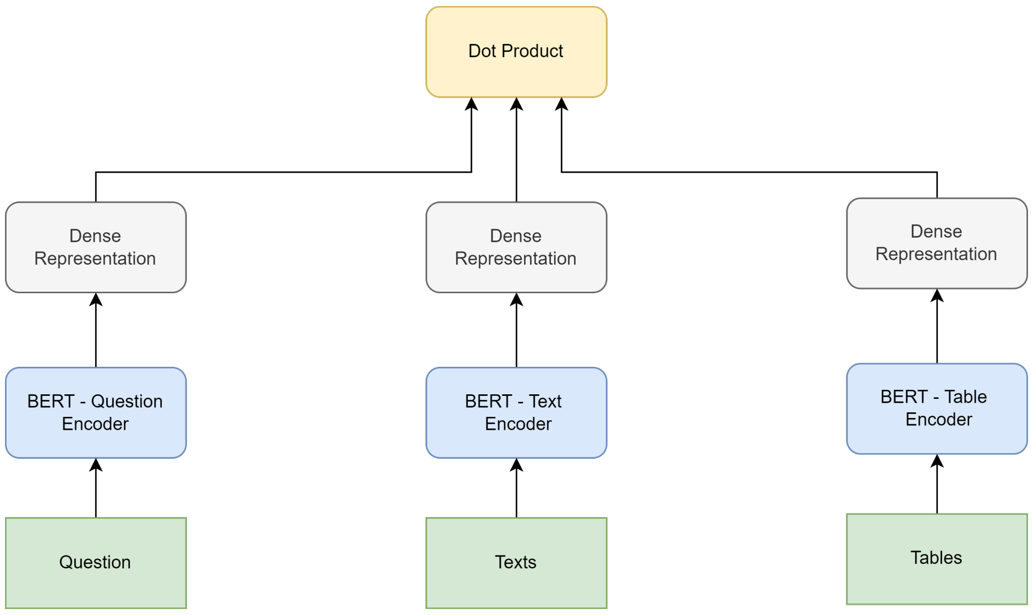

Figure 8: Representation of the Tri-encoder architecture of the work "Multimodal Retrieval of Tables and Texts Using Tri-encoder Models" (KOSTIĆ; RISCH; MÖLLER, 2021).

for the former and a supervised approach for the latter. A key advantage of this approach lies in the absence of supervision during retriever training, with rewards solely based on the reader's output and the golden answer. The retriever reward mechanism can be defined as follows:

$$R(a^g, a^{rc}|\tau) = \begin{cases} 2, \text{if } a^g = a^{rc}, \\ F_1(a^g, a^{rc}), \ \text{else if } a^g \cap a^{rc} \neq \emptyset, \\ -1, \text{else.} \end{cases} \tag{3.4}$$

In this reward system, a retriever earns a reward of 2 if the generated answer a^{rc} perfectly matches the golden answer a^g, and τ is the selected passage. If there is an intersection between the two answers, the reward is computed based on the F1-score discussed in Section 2.1.5. In cases where no matching words are found, the reward is -1. Notably, the reader and retriever networks in this approach are based on LSTM networks. The Figure 9 shows (R^3) training workflow.

Another relevant previous work is the "A Deep Reinforcement Learning Based Multi-Step Coarse to Fine Question Answering (MSCQA) System" (WANG; JIN, 2019). This proposal embraces a DRL approach for training an action selector, which, in turn, chooses among three different trained components for QA. The available actions include answering the question by invoking the reader, selecting additional passages through the retriever

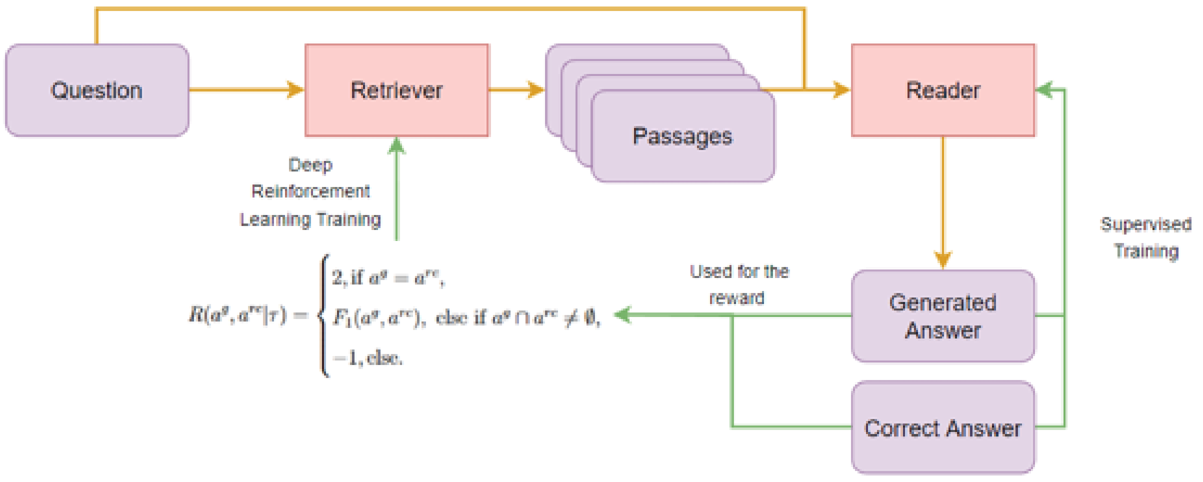
Question
Retriever
Passages
Reader
Deep Reinforcement Learning Training
Supervised Training
Used for the reward
$R(a^g, a^{rc}|\tau) = \begin{cases} 2, \text{if } a^g = a^{rc}, \\ F_1(a^g, a^{rc}), \text{ else if } a^g \cap a^{rc} \neq \emptyset, \\ -1, \text{else.} \end{cases}$
Generated Answer
Correct Answer

Figure 9: Training workflow of Reinforced Ranker-Reader (R^3) system by Wang et al. (2018).

module, and removing a potentially incorrect answer from the document memory.

In alignment with MSCQA, the research outlined in “A deep reinforcement learning approach to complex open-domain question answering” (CAÇÃO; COSTA, 2023) also employs a DRL agent with three similar actions: retriever, reader, and a cleaner tasked with removing passages that may be detractors. The primary contribution of this work lies in utilizing the DRL approach to address multi-hop questions, where the retriever operates iteratively, using previously retrieved passages to obtain new ones.

An interesting proposal that uses Reinforcement Learning for automatic QA is NLP-GYM by Ramamurthy, Sifa and Bauckhage (2020). It is a Python library made for testing RL Algorithms in three different NLP tasks: multiple choice QA, label sequence generation, and the sequence tagging. The QA task is based on the QASC dataset (KHOT et al., 2019), the state comprises the embedding of a question, two facts, and an alternative. The agent is equipped with two actions: answering using the provided alternative or requesting a new one. While it does not provide the best performance for multiple choice QA in general, it is a handy Python library for QA and RL that is compatible with GYM and various essential RL Python libraries, facilitating agent training and experimentation.

Reinforcement Learning with Human Feedback (RLHF) is an approach that has gained traction in the development of natural language models. It addresses the need for language models to better align with users, reduce instances of misinformation, toxicity, and harmful sentiment expression. RLHF involves training models through a process that combines Reinforcement Learning, where models learn from their actions and consequences, with Human Feedback, which involves human reviewers providing feedback and rankings on model-generated content. One example is ChatGPT (or GPT3.5), a conversational agent built upon the foundation of GPT3. While the specifics of ChatGPT's training process are not fully disclosed, available information indicates similarities to InstructGPT (OUYANG et al., 2022), which was also trained from GPT3. The RLHF training was executed through the PPO algorithm to enhance ChatGPT's alignment with users and minimizing issues related to misinformation, toxicity, and harmful sentiments.

4 PROPOSED ARCHITECTURE

The multi-hop and multimodal nature of OTT-QA presents a significant challenge due to the sequential retrieval process it requires. To address these challenges, we propose a novel architecture. Our approach involves training a Deep Reinforcement Learning (DRL) agent that selects from a set of already trained modules in the existing literature.

As done for MSCQA (WANG; JIN, 2019), we employ a DRL agent as an action selector to determine which already trained component should be activated. As depicted in Figure 10, we have the following actions: Retrieve Texts, Retrieve Tables, and Generate the Answer.

Given that OTT-QA does not specify the correct path for information retrieval, including the sequence of passages to retrieve, we have adopted a Reinforcement Learning approach. We incorporate a delayed reward mechanism, which compares the predicted answers to the gold standard answers at the end of each episode.

One notable advantage of our proposed architecture is its flexibility. All components, including the reader and retrievers, can be easily replaced with newer and superior models as they become available for public use. Furthermore, the system can be enhanced by incorporating additional components, such as a graph retrieval model, or employing multiple readers optimized for specific scenarios. The RL agent learns when to use each component in different situations through experiential training.

In the following sections, we will provide a more detailed explanation of each architecture component.

4.1 Initial Concepts

Our approach draws inspiration from the Iterative-Retrieval + Cross-Block Reader architecture introduced by Chen et al. (2021) and described in Section 3.2.1.

We also propose an iterative system. For a given question q, the initial retrieval step

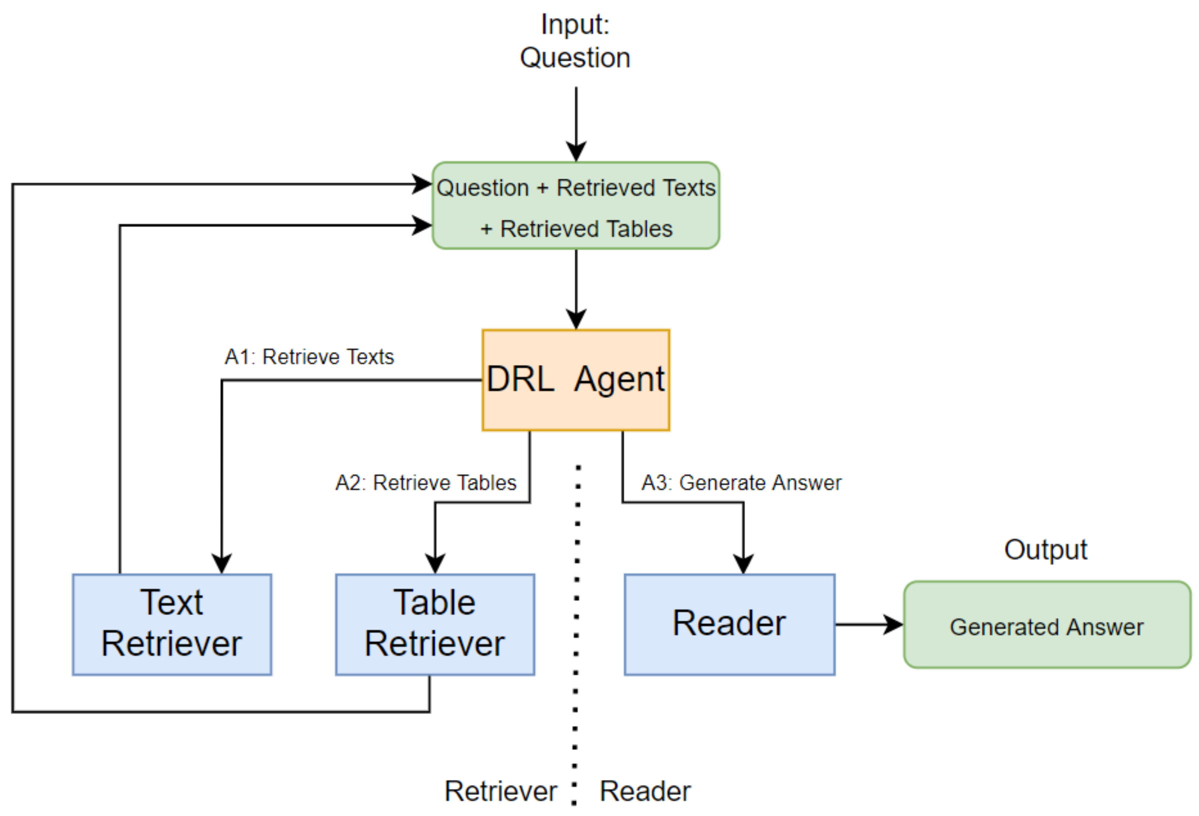

Figure 10: Proposed architecture. At each time step, the agent selects one of three actions: Retrieve Texts, Retrieve Tables, or Generate the Answer, based on the question and the information gathered so far.

involves searching for 10 passages $p_{i,j}$, which can be either textual ($te_{i,j}$) or tabular ($ta_{i,j}$) in nature. These passages are used to create 10 different blocks b_i as follows:

$$\begin{aligned} B &= [b_1, b_2, ..., b_{10}], \\ B &= [[q, p_{1,1}], [q, p_{2,1}], ..., [q, p_{10,1}]], \end{aligned} \tag{4.1}$$

where i denotes the block number, and j represents the index of the passage within the block.

Should the agent opt to perform an additional retrieval step, it concatenates the items within each block to retrieve four new passages, be they textual or tabular. This process generates a new set of blocks:

$$B = [[q, p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}, p_{1,5}], ..., [q, p_{10,1}, p_{10,2}, p_{10,3}, p_{10,4}, p_{10,5}]]. \tag{4.2}$$

The agent may undergo one more retrieval step, leading to the selection of 4 new passages for each block. To maintain the quality of retrieved information and prevent excessive noise, we limit the maximum number of retrieval steps to three. This limitation is intended to avoid an accumulation of uncertainty, as additional passages introduce more information, making it more challenging for the retrieval process to distinguish between valuable information and noise.

Suppose the agent decides to generate the answer or has reached the maximum number of steps. In that case, the reader is presented with the question and all non-repeated passages (note that different blocks may contain the same passage). This consolidated set of information is referred to as “Reader Input”:

$$RI = [q, p_1, p_2, ...]. \tag{4.3}$$

The reader utilizes the Reader Input to generate a predicted answer a^{rc}, which is subsequently compared to the golden answer a^g. In cases where the reader is invoked before any passage has been retrieved, a preliminary Table Retrieval is run to ensure that a minimum of 10 documents are available for processing. We opted for table retrieval due to its smaller document pool compared to texts. This decision increases the likelihood of reaching the answer, even without resorting to multi-hop steps.

4.2 Reward

Defining the reward function is arguably the most critical task in an RL environment because it is precisely what the agent seeks to maximize. However, it is not straightforward, as it provides an indirect training signal. We do not have prior knowledge of the correct sequence of actions for the agent to follow (if we did, we could use supervised learning). In our scenario, we must rely on a single signal: the comparison between the expected (golden) answer and the agent’s predicted answer to construct the reward function.

We adopted the reward function from R^3 (WANG et al., 2018) as the base for our work, which indirectly trains the retriever using the final answer (as expressed in Equation 3.4). However, given the complexity of OTT-QA, the penalty of -1 for cases where the generated answer has no intersection with the golden answer appeared excessive, particularly when considering that the Iterative-Retrieval + Cross-Block Reader, which served as our inspiration, achieved only a 20.9 F1-score on the test set, below the results for R^3 in all datasets that it was tested. Consequently, we decided to reduce this penalty to -0.5.

$$R(a^g, a^{rc}) = \begin{cases} 2, \text{if } a^g = a^{rc}, \\ F_1(a^g, a^{rc}),\ \text{else if } a^g \cap a^{rc} \neq \emptyset, \\ -0.5, \text{else}, \end{cases} \tag{4.4}$$

where the reward is assigned a value of 2 when the golden answer a^g precisely matches the predicted answer a^{rc}. In cases of partial match, the reward is determined by the F1-score as described in the Section 2.1.5. Lastly, when no match occurs, the reward defaults to -0.5, as mentioned before. We also introduced a minor penalty of -0.02 for each retrieval action taken. This penalty encourages the agent to minimize retrieving unnecessary texts and tables.

4.3 Actions

As mentioned previously, we have three different actions:

- $A1$ **- Retrieve Texts:** If this action is selected, a retriever model is activated to search for relevant textual information based on the already retrieved passages concatenated with the question.
- $A2$ **- Retrieve Tables:** If this action is chosen, a retriever model searches for relevant tables based on the already retrieved passages concatenated with the question.
- $A3$ **- Generate Answer:** When this action is selected, a reader model generates the answer using the extracted textual passages and tables. This action terminates the episode.

In the following subsections, we discuss our choices for the models for the reader and retriever. We decided not to retrain any of the models taken from the literature, so we had the restriction of using only the ones that were made publicly available.

4.3.1 Retriever

The retriever module is responsible for gathering information for the reader. We conducted tests using two distinct retrievers: BM25 (ROBERTSON; ZARAGOZA, 2009) and Tri-encoder (KOSTIĆ; RISCH; MÖLLER, 2021). This approach allowed us to evaluate both classical sparse retrieval and neural dense retrieval methods, mirroring the approach taken in the Iterative-Retrieval + Cross-Block Reader architecture.

4.3.2 Reader

One of the most important modules of a QA system is the reader, responsible for generating answers. For OTT-QA, the publicly available reader with the best performance is the Fusion-in-Encoder (FiE) model from the COS system (MA et al., 2023). A distinguishing feature of this model is its ability to process passages separately, making it highly versatile for handling numerous passages as input. In line with the recommendations provided by the authors, we limited the number of input passages to 50, as this configuration yielded the best results in their validation experiments.

4.4 State/Observation

The observation is the information that the agent uses for the decision-making process. In our case, this information comprises the question and the passages that have been retrieved up to that point. Rather than working directly with pure texts and tables, we first convert these elements into numerical data using embeddings.

We employ two distinct encoders to perform this transformation into embeddings. When the retriever is the Tri-encoder, we utilize its embeddings to represent the question, retrieved texts, and retrieved tables. However, in the case when we utilized BM25, we leverage the representations provided by a fine-tuned MPNET model trained for semantic search in QA[1]. Each embedding is a vector with length of 512 for Tri-encoder (same as small-BERT) and 768 for the MPNET model.

This information is then translated into a sequence of 11 vectors, with one vector for the question (e_q) and one for each of the 10 blocks (e_i). For each block, we calculate the average of the representation values for each passage, considering the question in isolation, generating the embedding sequence E:

$$E = [e_q, e_1, e_2, ...e_{10}]. \tag{4.5}$$

4.5 Agent

In DRL applications, the agent's primary function is to process observations and generate corresponding actions. Our specific observations consist of a sequence comprising

[1]The MPNET model can be at ⟨https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1⟩

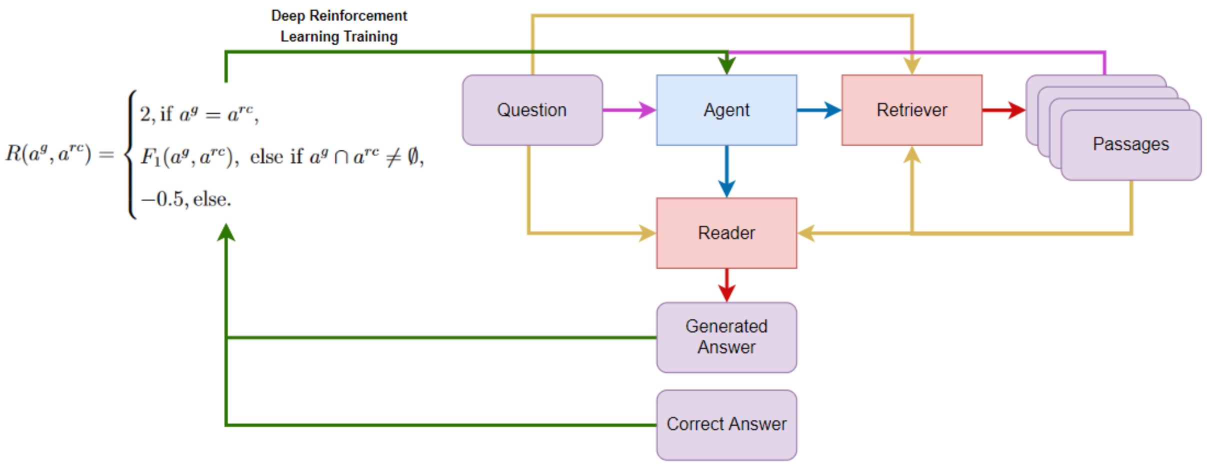
Deep Reinforcement Learning Training
$R(a^g, a^{rc}) = \begin{cases} 2, \text{if } a^g = a^{rc}, \\ F_1(a^g, a^{rc}), \text{ else if } a^g \cap a^{rc} \neq \emptyset, \\ -0.5, \text{else.} \end{cases}$
Question
Agent
Retriever
Passages
Reader
Generated Answer
Correct Answer

Figure 11: The training workflow of the proposed architecture. The green arrows represent the rewards utilized for training the agent, while the yellow arrows depict raw data, consisting of questions and passages. The red arrows signify the outputs of the retriever and reader modules. Blue arrows represent the actions taken by the agent. Finally, the pink arrows denote the embeddings employed as input to the DRL agent.

11 vectors, and in our exploration, we experimented with four distinct neural network architectures: MultiLayer Perceptron (MLP), LSTM, GRU, and Transformer.

For the MLP, we flattened the entire sequence into a single vector, which meant adding zeros in the first step to account for the lack of retrieved passages at that point. In contrast, the other networks process this information as a sequence.

To train the agent, we explored two widely-used algorithms: DQN and PPO. Our implementation is based on the Stable-Baselines3 Python library, selected for its robustness and reliability. We chose these algorithms for specific reasons—DQN is known for its sample-efficiency as it stores experiences in a replay buffer, while PPO offers stability due to loss clipping in network updates.

The training workflow of the agent is depicted in Figure 11.

4.6 Environment

As mentioned before, the environment in a RL setting includes everything that is outside the agent. As shown in Figure 4, it receives the actions dictated by the agent and returns the reward and new observation. The complete pseudo-code for the environment is presented in Algorithm 3.

Algorithm 3: Environment

```
Initialize Retriever and Reader
Initialize Agent
foreach episode do
    Sample a random question $q$ with the corresponding answer $a^g$
    $step \leftarrow 1$
    Get the embedding $E$ from $q$
    $Done \leftarrow True$
    while not Done do
        Agent acts according to its policy $\pi(a|s)$ and returns Action
        if Action is A1 then
            if step is 0 then
                Retrieve 10 texts passages using $q$ to create blocks vector $B$
            else
                foreach block $b_i$ in $B$ do
                    $p_{i,j} \leftarrow$ Retrieve 4 texts passages using $b_i$
                    Append $p_{i,j}$ to $b_i$
                end foreach
            end if
            $R \leftarrow -0.02$
        end if
        if Action is A2 then
            if step is 0 then
                Retrieve 10 tables using $q$ to create blocks vector $B$
            else
                foreach block $b_i$ in $B$ do
                    $p_{i,j} \leftarrow$ Retrieve 4 tables using $b_i$
                    Append $p_{i,j}$ to $b_i$
                end foreach
            end if
            $R \leftarrow -0.02$
        end if
        if Action is A3 or step is 3 then
            if step is 0 then
                Retrieve 10 tables using $q$ to create blocks vector $B$
            end if
            Transformation of $B$ to create the reader input $RI$
            With $RI$, the Reader generates the predicted answer $a^{rc}$
            $R \leftarrow \begin{cases} 2, \text{if } a^g = a^{rc}, \\ F_1(a^g, a^{rc}), \text{ else if } a^g \cap a^{rc} \neq \emptyset, \\ -0.5, \text{else.} \end{cases}$
            $Done \leftarrow True$
        end if
        $step \leftarrow step + 1$
        Calculate the embedding sequence vector $E$
        The agent receives $E$ as the state and the reward $R$
    end while
end foreach
```

5 EXPERIMENTS

In this chapter, we present the experimental evaluation of our architecture. The goal is to assess a DRL agent's performance by exploring various baselines and training approaches. We conducted a comprehensive set of experiments, testing 28 different baselines and running 16 distinct training sessions.

5.1 Baselines

Before training the DRL agent, we explored baseline scenarios using our architecture. These baselines involved fixed sets of actions chosen independently of the question and retrieved information. Each conceivable combination of actions was systematically evaluated across a validation dataset. For example, one baseline involved the agent consistently choosing to retrieve texts (Action $A1$) twice and then invoking the reader to generate the answer. In total, we explored 14 different configurations of actions, encompassing two retriever types——BM25 and Tri-encoder——resulting in 28 baselines.

The primary purpose of these baseline tests was to establish a benchmark for comparison, determining if the trained DRL agent could outperform or at least match the performance of fixed action paths without contextual understanding.

5.2 Training

The training of the DRL agent involved sampling a random question from the OTT-QA training set for each episode. We evaluated the impact of training both retriever solutions and compared their performance. For the Tri-encoder, we trained for a total of one million timesteps, a reasonable number considering the dataset size of 41,469 training questions. Each question could take up to three timesteps, allowing for revisiting the same question multiple times. Due to the higher inference time of BM25 (up to 20 times more than Tri-encoder), we opted to train for only 100 thousand timesteps.

We implemented the DQN and PPO algorithms using the Stable-Baselines3 Python library, employing various neural network architectures, including MLP, LSTM, GRU, and transformer. Subsequent subsections provide detailed explanations of these implementations.

5.2.1 DQN

For training DQN, we utilized most of the default hyperparameters of Stable-Baselines3[1]. However, we made a few adjustments, primarily to account for the number of training steps. We set the buffer size to 500,000 (maximum number of transitions stored in the experience replay) and learning starts (number of steps taking random actions just to store some transitions in the buffer before training) to 50,000 for the system with Tri-encoder as the retriever. For BM25, we decreased the learning starts by a factor of 5, taking into account that the training has only 100,000 steps.

Concerning network configurations, we used the following:

- **MLP:** The MLP network is a simple network with three hidden layers having 512, 128, and 64 neurons.
- **LSTM:** We employed two LSTM layers with a hidden size of 512 for the Tri-encoder and 768 for the BM25 architecture to match the size of the vectors of the embedding sequence E, with a dropout of 0.1. On top of that, there are two hidden layers, 128 and 64 nodes, respectively.
- **GRU:** Similar to LSTM, but replacing the LSTM layers with GRU layers.
- **Transformer:** We opted for two encoder transformer layers with two attention heads, a feature dimension matching the input embedding size, and the same 128 and 64 nodes hidden layers.

5.2.2 PPO

Similar to DQN, we primarily relied on the default hyperparameters for training using the PPO implementation from Stable-Baselines3[2]. Our modifications were minimal,

[1]For more information about DQN hyperparameters: ⟨https://stable-baselines3.readthedocs.io/en/master/modules/dqn.html⟩

[2]For more information about PPO hyperparameters: ⟨https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html⟩

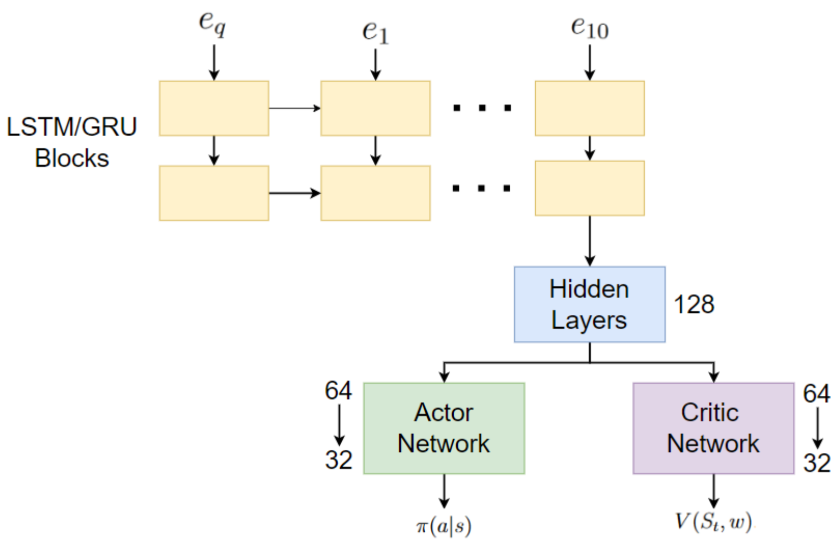

Figure 12: Feature Extractor for the LSTM in our setting. The agent receives the embedding sequence $E = [e_q, e_1, ...e_{10}]$ that is processed through two LSTM/GRU layers and a hidden layer with 128 nodes. The result is then fed to both the actor and the critic networks with two hidden layers with sizes 64 and 32.

primarily addressing the need for more frequent updates due to potentially limited experience accumulation in our context. We set the number of steps to run before training as 128, with a batch size of 32, and 60 as the number of epochs to optimize the surrogate loss.

As PPO is an Actor-Critic algorithm, it employs two different networks: the actor and the critic. Instead of using entirely separate networks, a common approach is to utilize a feature extractor to avoid redundant computations. This involves using shared layers between the networks for pre-processing the input, with these shared layers being trained jointly. Figure 12 illustrates our setup for LSTM and GRU networks as an example.

We employed two hidden layers with 64 and 32 neurons for both the actor and critic networks. Below are the configurations for the feature extractors we utilized, maintaining consistency with the configurations used for DQN:

- **MLP:** A two-layered network with hidden sizes of 512 and 128.
- **LSTM:** We used two LSTM layers with a hidden size of 512 for the Tri-encoder and 768 for the BM25 architecture to match the size of the vectors of the embedding sequence E, with a dropout of 0.1. Additionally, there is one hidden layer with 128 nodes.
- **GRU:** Similar to LSTM, but with the LSTM layers replaced by GRU layers.

- **Transformer:** We opted for two encoder transformer layers with two attention heads, a feature dimension matching the input embedding size, and the same 128 hidden layer.

6 RESULTS AND DISCUSSION

In this chapter, we present and discuss the outcomes of our architecture on the OTT-QA dataset, utilizing the experiments outlined in the previous chapter. The evaluation is based on the metrics EM and F1-score explained in section 2.1.5.

6.1 Baselines

The complete results of our baselines are provided in Table 4 for the system with BM25 as the retriever and in Table 5 for the Tri-encoder Retriever.

For the BM25 retriever, it is evident that retrieving tables is less effective than retrieving texts. This imbalance results in the best sequence of actions always being the retrieval of texts and never tables, yielding an F1-score of 19.03.

For the Tri-encoder, the results are more balanced. The optimal sequence of actions involves retrieving texts, followed by tables, and then retrieving texts again, resulting in a total F1-score of 8.24. However, it is notable that the performance when retrieving texts is considerably lower than the BM25 retriever.

6.2 Deep Reinforcement Learning Agent Results

In this section, we present detailed results for the trained DRL agents, which are conveyed through two formats: firstly, a graphical representation of training curves, and secondly, a table showcasing model performance on the validation set.

The left graphs illustrates the Average Episodic Training reward, highlighting variations among different neural networks. Simultaneously, the right side focuses on the Average Episode Length. To facilitate the analysis, all graphs were generated with a smoothing factor of 0.95. This choice enhances clarity in the visual representation, particularly given the inherent high variance in results, where outcomes are typically binary

Action 1	**Action 2**	**Action 3**	**Action 4**	**EM**	**F1-score**
$A1$	$A3$	-	-	12.92	16.80
$A2$	$A3$	-	-	2.17	4.20
$A1$	$A1$	$A3$	-	14.60	18.79
$A1$	$A2$	$A3$	-	13.15	16.88
$A2$	$A1$	$A3$	-	5.01	7.97
$A2$	$A2$	$A3$	-	2.12	4.07
$A1$	$A1$	$A1$	$A3$	**14.81**	**19.03**
$A1$	$A1$	$A2$	$A3$	14.23	18.21
$A1$	$A2$	$A1$	$A3$	13.32	17.36
$A1$	$A2$	$A2$	$A3$	7.59	10.52
$A2$	$A1$	$A1$	$A3$	5.15	8.08
$A2$	$A1$	$A2$	$A3$	3.61	6.17
$A2$	$A2$	$A1$	$A3$	3.21	5.45
$A2$	$A2$	$A2$	$A3$	2.08	4.15

Table 4: Baseline results for the OTT-QA validation dataset using our framework with BM25 as the retriever. In this experiment, we assumed that the agent always takes the same sequence of actions, regardless of the question. In this experiment, $A1$ corresponds to retrieving texts, $A2$ is retrieving tables, and $A3$ calls the reader to generate the answer.

Action 1	**Action 2**	**Action 3**	**Action 4**	**EM**	**F1-score**
$A1$	$A3$	-	-	3.52	6.32
$A2$	$A3$	-	-	2.94	4.41
$A1$	$A1$	$A3$	-	4.34	7.11
$A1$	$A2$	$A3$	-	4.83	7.54
$A2$	$A1$	$A3$	-	5.42	7.89
$A2$	$A2$	$A3$	-	2.66	4.40
$A1$	$A1$	$A1$	$A3$	4.65	7.35
$A1$	$A1$	$A2$	$A3$	4.79	7.56
$A1$	$A2$	$A1$	$A3$	**5.55**	**8.24**
$A1$	$A2$	$A2$	$A3$	3.66	5.81
$A2$	$A1$	$A1$	$A3$	5.51	8.06
$A2$	$A1$	$A2$	$A3$	4.29	6.45
$A2$	$A2$	$A1$	$A3$	4.07	6.24
$A2$	$A2$	$A2$	$A3$	2.89	4.52

Table 5: Baseline results for the OTT-QA dataset using our framework with Tri-encoder as the retriever.

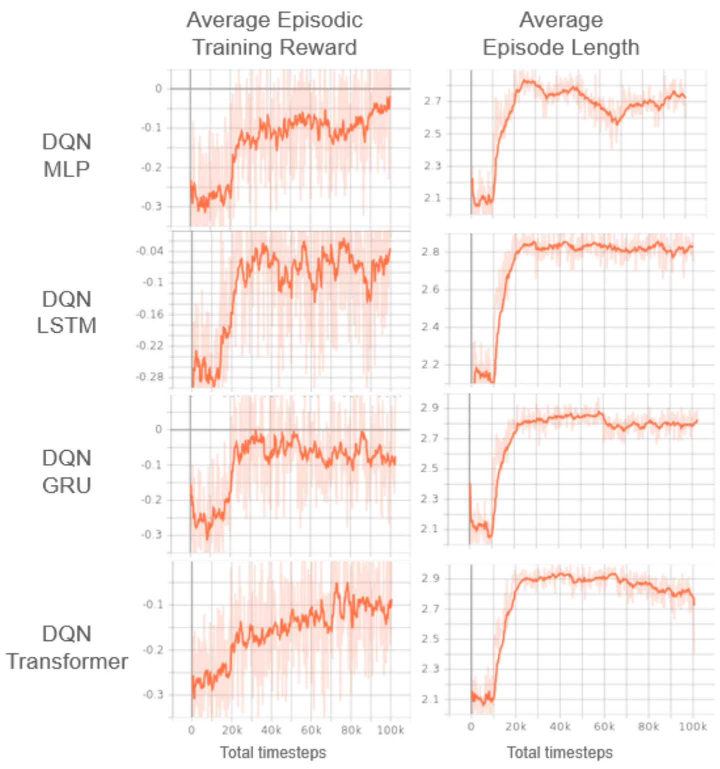

Figure 13: Training curves for DQN using the BM25 retriever.

(correct or incorrect), resulting in rewards ranging from 2 to -0.5.

6.2.1 BM25

The training curves for DQN, as illustrated in Figure 13, showcase the agent's ability to enhance its performance amidst the considerable variance and complexity of the environment. Despite the challenges presented, a discernible upward trend in performance is evident. In contrast, the improvements for PPO, depicted in Figure 14, do not exhibit the same level of clarity as observed in the case of DQN.

The performance of trained algorithms using the BM25 retriever is detailed in Table 6. Among the various training algorithms, the PPO agent equipped with a transformer neural network demonstrated the best result. Surpassing other configurations, this agent converged to always acting the same as the best baseline, which involved consistently selecting the $A1$ action regardless of the question or context.

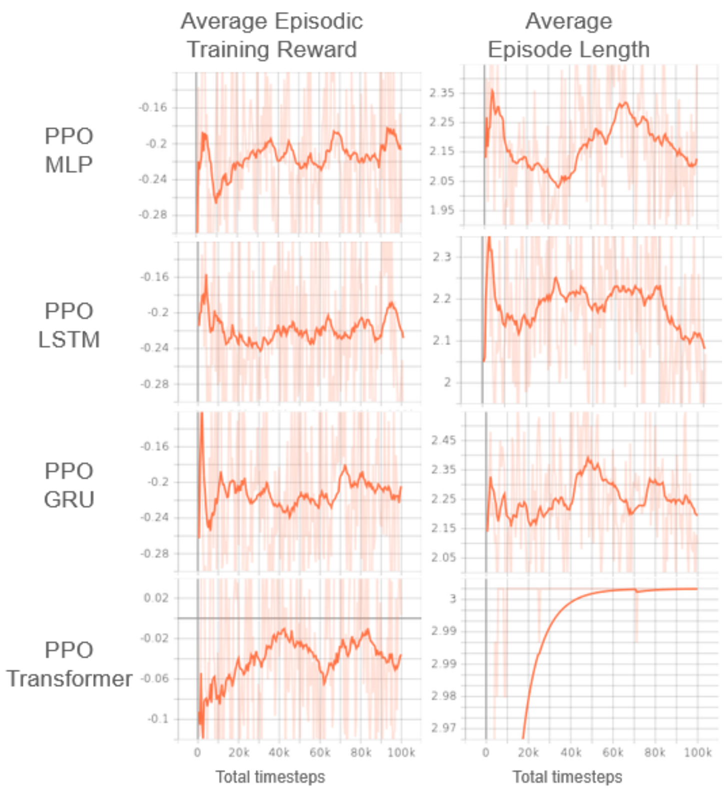

Figure 14: Training curves for PPO using the BM25 Retriever.

Training Algorithm	**Network**	**EM**	**F1-score**
DQN	MLP	11.70	15.44
DQN	LSTM	12.33	16.42
DQN	GRU	13.23	17.16
DQN	Transformer	11.65	15.12
PPO	MLP	8.54	11.63
PPO	LSTM	6.78	9.85
PPO	GRU	11.25	14.90
PPO	**Transformer**	**14.81**	**19.03**

Table 6: Results for different networks and training algorithms on OTT-QA validation set using BM25.

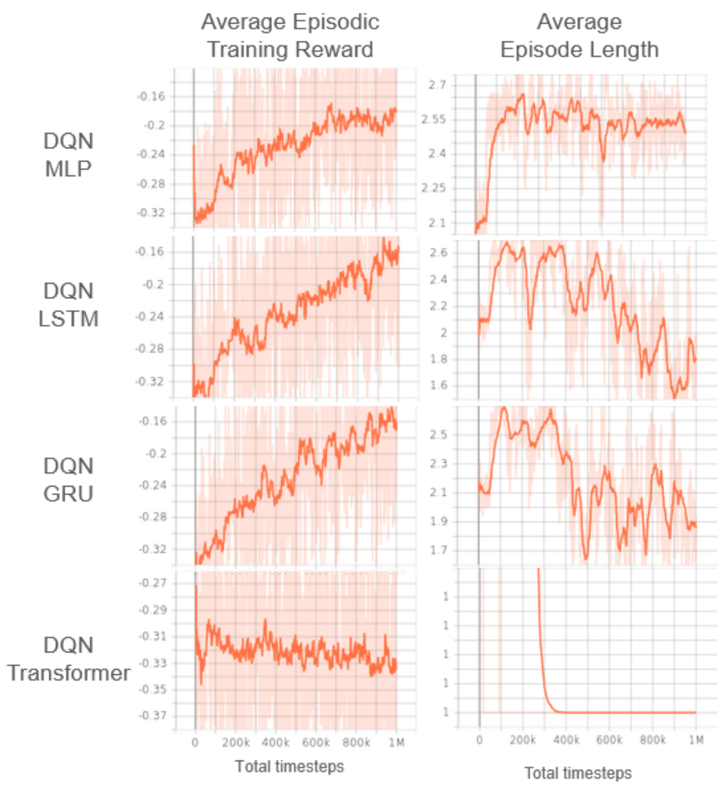

Figure 15: Training curves for DQN using the Tri-encoder Retriever.

6.2.2 Tri-encoder

Similar to the observed trend with the BM25 retriever, a comparable pattern is noticeable when employing the Tri-encoder Retriever. In Figure 15, the training performance of DQN exhibits a more consistent and stable improvement over time compared to PPO, as depicted in Figure 16.

The outcomes of trained algorithms utilizing the Tri-encoder retriever are presented in Table 7. Notably, the best-performing agent emerged from the PPO algorithm, employing the MLP network, demonstrating a F1-score of 8.18.

6.3 Discussion

Analyzing the baseline results reveals a substantial advantage for text retrieval over table search for BM25. This discrepancy may stem from the inherent characteristics of tables as structured data entities. BM25, being a text-based ranking function, may struggle to discern relevance in structured tables where factors like column presence and data types play a pivotal role.

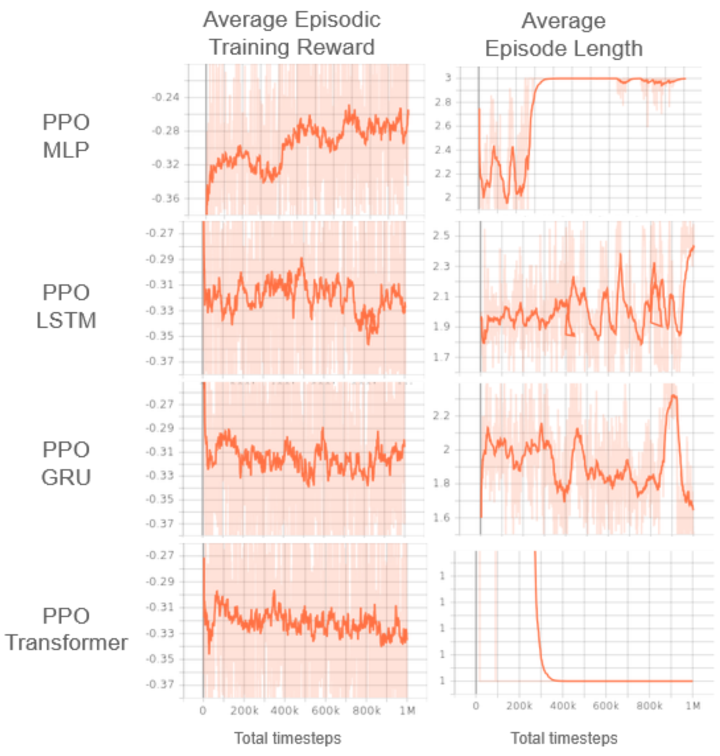

Figure 16: Training curves for PPO using the Tri-encoder Retriever.

Training Algorithm	Network	EM	F1-score
DQN	MLP	3.61	6.01
DQN	LSTM	3.52	5.68
DQN	GRU	3.79	6.13
DQN	Transformer	3.34	5.09
PPO	**MLP**	**5.65**	**8.18**
PPO	LSTM	3.75	5.53
PPO	GRU	2.94	4.93
PPO	Transformer	2.80	4.34

Table 7: Results for different networks and training algorithms on OTT-QA validation set using Tri-encoder.

Despite being a prominent choice in OTT-QA solutions, the Tri-encoder exhibited F1-scores below 10. Two potential contributing factors are identified. First, the absence of multi-hop training may have limited its ability to navigate complex information hierarchies effectively. Also, the data used to train it had text retrieval taken from other datasets since this information is not present in OTT-QA.

Before discussing the results for the trained agents, we must acknowledge that we ran the training experiments with only one seed because each training execution took weeks. Therefore, they may be representative and just outliers, which makes it difficult to be conclusive about comparisons.

The DRL agents demonstrated comparable or slightly inferior performance to the best baseline, particularly for BM25. This trend aligns with the DRL agents often favoring the action of retrieving texts, mirroring the optimal baseline strategy.

Although the PPO algorithm demonstrated a slightly lower median performance, it yielded the best results for both retrievers. Particularly, the PPO agent equipped with a Transformer network for BM25 achieved the best performance by converging to the trivial solution of consistently selecting the best action: always retrieving texts. In the case of the Tri-encoder, the PPO agent utilizing an MLP network achieved an F1-score of 8.18, demonstrating a performance closely comparable to the best baseline of 8.24.

Furthermore, the neural network type revealed no discernible trend, underscoring the necessity for additional experimentation with an expanded set of trials to arrive at a more conclusive understanding in this regard

The best-performing DRL agent achieved an F1-score of 19.03 using BM25, approaching the performance 20.7 from the Iterative-Retrieval (sparse) + Cross-Block solution, which served as our primary inspiration. However, when comparing our dense approach, our best result was only 8.24 compared to theirs, 18.5, indicating that the our setup with Tri-encoder needs further refinement.

6.4 Limitations

The inherent exploratory character of RL and the stochastic behavior of neural networks underscore the importance of conducting multiple test runs for each experiment, followed by calculating the average outcomes. Unfortunately, we could not execute this in our experiments due to time constraints, which extended beyond a week for each training execution.

Moreover, when training the solution with the BM25 retriever, we utilized only 100,000 training steps. That is too few considering the dataset size of 41,469 training questions, each potentially taking up to three steps.

Another limitation of our work was the performance of the Tri-encoder in our setting. As mentioned before, it was not trained to retrieve in an iterative setting, which may have hindered its performance. A possible solution is utilizing other solutions or adapting the model's training for multi-hop.

Furthermore, the process of retrieving tables is noisy, particularly in iterative search. This happens because tables have too much data and, usually, only one row has the desired information. Addressing this issue may involve breaking each table into rows or smaller content sets, aligning with approaches proposed by other researchers (MA et al., 2022).

7 CONCLUSION AND FUTURE WORK

In this work, we introduced a novel system designed to tackle open-domain multi-hop questions across texts and tables. Our approach relies on a DRL decision-maker system that iteratively selects among three distinct modules: Text Retriever, Table Retriever, and Reader.

Our best system achieved an F1-score of 19.03, demonstrating competitive performance compared to the non DRL similar system Iterative-Retrieval (sparse) + Cross-Block solution, which scored 20.7. However, the architecture falls short comparing to other systems with entirely different retrieval strategies on the literature, such as the 63.2 achieved by the COS architecture on the dev set.

While our results may appear comparatively lower than those presented in existing literature, we posit that there exists untapped potential for performance enhancement by modifying various modules within the system.

Our forthcoming efforts involve the implementation of the Joint-Reranking system from DuRePa. A key focus in this augmentation is the reinforcement of the architecture's ability to clear excess passages and tables from memory. This refinement aims to mitigate potential adverse effects on the reader's performance by eliminating undesirable information.

Furthermore, we are exploring alternative methods to generate observations for the agent. The current approach involves averaging the embeddings of each block, but another possibility is utilizing the observation as sequence with the encoding of all distinct retrieved passages.

An approach for improving our system would be to integrate our DRL agent with the modules of the COS architecture, which is known for achieving the best performance in OTT-QA at the time of writing this document. COS implements a retrieval with different skills that can be called in any order, so our DRL agent could decide the paths depending on the question and retrieved passages.

Lastly, to bolster the robustness of our findings, we intend to conduct experiments with different seeds for a more comprehensive statistical analysis.

REFERENCES

AINSLIE, J.; ONTANON, S.; ALBERTI, C.; CVICEK, V.; FISHER, Z.; PHAM, P.; RAVULA, A.; SANGHAI, S.; WANG, Q.; YANG, L. ETC: Encoding long and structured inputs in transformers. In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020. p. 268–284. Available at: ⟨https://aclanthology.org/2020.emnlp-main.19⟩.

ARCHANJO, J. M.; COZMAN, F. G. mRAT-SQL+GAP: A Portuguese Text-to-SQL Transformer. In: *Intelligent Systems*. [S.l.]: Springer International Publishing, 2021, (Lecture Notes in Computer Science, v. 13074). p. 511–525. ISBN 978-3-030-91698-5.

BELLMAN, R. *Dynamic Programming*. 1. ed. Princeton, NJ, USA: Princeton University Press, 1957.

BELTAGY, I.; PETERS, M. E.; COHAN, A. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, v. 5, n. 2, p. 157–166, 1994. ISSN 1045-9227 (Print).

BERANT, J.; CHOU, A. K.; FROSTIG, R.; LIANG, P. Semantic parsing on freebase from question-answer pairs. In: *Conference on Empirical Methods in Natural Language Processing*. [s.n.], 2013. Available at: ⟨https://api.semanticscholar.org/CorpusID:6401679⟩.

BROCKMAN, G.; CHEUNG, V.; PETTERSSON, L.; SCHNEIDER, J.; SCHULMAN, J.; TANG, J.; ZAREMBA, W. Openai gym. *CoRR*, abs/1606.01540, 2016. Available at: ⟨http://arxiv.org/abs/1606.01540⟩.

BROWN, T. B.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A.; AGARWAL, S.; HERBERT-VOSS, A.; KRUEGER, G.; HENIGHAN, T.; CHILD, R.; RAMESH, A.; ZIEGLER, D. M.; WU, J.; WINTER, C.; HESSE, C.; CHEN, M.; SIGLER, E.; LITWIN, M.; GRAY, S.; CHESS, B.; CLARK, J.; BERNER, C.; MCCANDLISH, S.; RADFORD, A.; SUTSKEVER, I.; AMODEI, D. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. Available at: ⟨https://arxiv.org/abs/2005.14165⟩.

CAÇÃO, F. N.; JOSÉ, M. M.; OLIVEIRA, A. S.; SPINDOLA, S.; COSTA, A. H. R.; COZMAN, F. G. DEEPAGÉ: Answering questions in portuguese about the brazilian environment. In: BRITTO, A.; DELGADO, K. V. (Ed.). *Intelligent Systems*. Cham: Springer International Publishing, 2021. p. 419–433. ISBN 978-3-030-91699-2.

CARMO, D.; PIAU, M.; CAMPIOTTI, I.; NOGUEIRA, R.; LOTUFO, R. de A. PTT5: pretraining and validating the T5 model on brazilian portuguese data. *CoRR*, abs/2008.09144, 2020. Available at: ⟨https://arxiv.org/abs/2008.09144⟩.

CAÇÃO, F. N.; COSTA, A. H. R. *A deep reinforcement learning approach to complex open-domain question answering.* Dissertação (Mestrado) — Universidade de São Paulo, 2023.

CHEN, D.; FISCH, A.; WESTON, J.; BORDES, A. Reading Wikipedia to answer open-domain questions. In: *Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada: Association for Computational Linguistics, 2017. p. 1870–1879.

CHEN, W.; CHANG, M.-W.; SCHLINGER, E.; WANG, W. Y.; COHEN, W. W. Open question answering over tables and text. In: *International Conference on Learning Representations*. [s.n.], 2021. Available at: ⟨https://openreview.net/forum?id=MmCRswl1UYl⟩.

CHEN, W.; ZHA, H.; CHEN, Z.; XIONG, W.; WANG, H.; WANG, W. Y. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In: *Findings of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. p. 1026–1036. Available at: ⟨https://aclanthology.org/2020.findings-emnlp.91⟩.

CHO, K.; MERRIËNBOER, B. van; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: MOSCHITTI, A.; PANG, B.; DAELEMANS, W. (Ed.). *Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1724–1734. Available at: ⟨https://aclanthology.org/D14-1179⟩.

CHUNG, J.; GULCEHRE, C.; CHO, K.; BENGIO, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: *NIPS 2014 Workshop on Deep Learning, December 2014*. [S.l.: s.n.], 2014.

CLARK, K.; LUONG, M.; LE, Q. V.; MANNING, C. D. ELECTRA: pre-training text encoders as discriminators rather than generators. In: *International Conference on Learning Representations*. OpenReview.net, 2020. Available at: ⟨https://openreview.net/forum?id=r1xMH1BtvB⟩.

CLARK, P.; COWHEY, I.; ETZIONI, O.; KHOT, T.; SABHARWAL, A.; SCHOENICK, C.; TAFJORD, O. Think you have solved question answering? try ARC, the AI2 Reasoning challenge. *CoRR*, abs/1803.05457, 2018. Available at: ⟨http://arxiv.org/abs/1803.05457⟩.

COTRIM, L.; JOSÉ, M.; CABRAL, E. Reinforcement learning control of robot manipulator. *Revista Brasileira de Computação Aplicada*, v. 13, n. 3, p. 42–53, 2021. Available at: ⟨http://seer.upf.br/index.php/rbca/article/view/12091⟩.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Available at: ⟨https://aclanthology.org/N19-1423⟩.

FEDUS, W.; ZOPH, B.; SHAZEER, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, v. 23, n. 120, p. 1–39, 2022. Available at: ⟨http://jmlr.org/papers/v23/21-0998.html⟩.

FERRUCCI, D.; BROWN, E.; CHU-CARROLL, J.; FAN, J.; GONDEK, D.; KALYANPUR, A. A.; LALLY, A.; MURDOCK, J. W.; NYBERG, E.; PRAGER, J.; SCHLAEFER, N.; WELTY, C. Building watson: An overview of the deepQA project. *AI Magazine*, v. 31, n. 3, p. 59–79, 2010. ISSN 07384602.

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial nets. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014. v. 27. Available at: ⟨https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf⟩.

GREEN, B. F.; WOLF, A. K.; CHOMSKY, C. L.; LAUGHERY, K. Baseball: an automatic question-answerer. In: *IRE-AIEE-ACM '61 (Western)*. [S.l.: s.n.], 1961.

GUU, K.; LEE, K.; TUNG, Z.; PASUPAT, P.; CHANG, M. W. REALM: Retrieval-Augmented language model pre-training. *International Conference on Machine Learning*, PartF168147-6, p. 3887–3896, 2020.

HERZIG, J.; NOWAK, P. K.; MÜLLER, T.; PICCINNO, F.; EISENSCHLOS, J. TaPas: Weakly supervised table parsing via pre-training. In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. p. 4320–4333. Available at: ⟨https://aclanthology.org/2020.acl-main.398⟩.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 11 1997. ISSN 0899-7667. Available at: ⟨https://doi.org/10.1162/neco.1997.9.8.1735⟩.

HUANG, J.; ZHONG, W.; LIU, Q.; GONG, M.; JIANG, D.; DUAN, N. Mixed-modality representation learning and pre-training for joint table-and-text retrieval in OpenQA. In: *Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022. p. 4117–4129. Available at: ⟨https://aclanthology.org/2022.findings-emnlp.303⟩.

IZACARD, G.; GRAVE, E. Leveraging passage retrieval with generative models for open domain question answering. In: *Conference of the European Chapter of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2021. p. 874–880. Available at: ⟨https://aclanthology.org/2021.eacl-main.74⟩.

JIN, Q.; DHINGRA, B.; LIU, Z.; COHEN, W.; LU, X. PubMedQA: A dataset for biomedical research question answering. In: *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 2019. p. 2567–2577. Available at: ⟨https://aclanthology.org/D19-1259⟩.

JOSÉ, M. M.; JOSÉ, M. A.; MAUÁ, D. D.; COZMAN, F. G. Integrating question answering and text-to-SQL in portuguese. In: PINHEIRO, V.; GAMALLO, P.; AMARO, R.; SCARTON, C.; BATISTA, F.; SILVA, D.; MAGRO, C.; PINTO, H. (Ed.).

Computational Processing of the Portuguese Language. Cham: Springer International Publishing, 2022. p. 278–287. ISBN 978-3-030-98305-5.

KARPUKHIN, V.; OGUZ, B.; MIN, S.; LEWIS, P.; WU, L.; EDUNOV, S.; CHEN, D.; YIH, W.-t. Dense passage retrieval for open-domain question answering. In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020. p. 6769–6781. Available at: ⟨https://www.aclweb.org/anthology/2020.emnlp-main.550⟩.

KEDIA, A.; ZAIDI, M. A.; LEE, H. FiE: Building a global probability space by leveraging early fusion in encoder for open-domain question answering. In: *Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022. p. 4246–4260. Available at: ⟨https://aclanthology.org/2022.emnlp-main.285⟩.

KHASHABI, D.; MIN, S.; KHOT, T.; SABHARWAL, A.; TAFJORD, O.; CLARK, P.; HAJISHIRZI, H. UNIFIEDQA: Crossing format boundaries with a single QA system. In: COHN, T.; HE, Y.; LIU, Y. (Ed.). *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020. p. 1896–1907. Available at: ⟨https://aclanthology.org/2020.findings-emnlp.171⟩.

KHOT, T.; CLARK, P.; GUERQUIN, M.; JANSEN, P. A.; SABHARWAL, A. QASC: a dataset for question answering via sentence composition. In: *AAAI Conference on Artificial Intelligence*. [s.n.], 2019. Available at: ⟨https://api.semanticscholar.org/CorpusID:204915921⟩.

KOSTIĆ, B.; RISCH, J.; MÖLLER, T. Multi-modal retrieval of tables and texts using tri-encoder models. In: FISCH, A.; TALMOR, A.; CHEN, D.; CHOI, E.; SEO, M.; LEWIS, P.; JIA, R.; MIN, S. (Ed.). *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*. Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021. p. 82–91. Available at: ⟨https://aclanthology.org/2021.mrqa-1.8⟩.

KWIATKOWSKI, T.; PALOMAKI, J.; REDFIELD, O.; COLLINS, M.; PARIKH, A.; ALBERTI, C.; EPSTEIN, D.; POLOSUKHIN, I.; DEVLIN, J.; LEE, K.; TOUTANOVA, K.; JONES, L.; KELCEY, M.; CHANG, M.-W.; DAI, A. M.; USZKOREIT, J.; LE, Q.; PETROV, S. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, MIT Press, Cambridge, MA, v. 7, p. 452–466, 2019. Available at: ⟨https://aclanthology.org/Q19-1026⟩.

LEWIS, M.; LIU, Y.; GOYAL, N.; GHAZVININEJAD, M.; MOHAMED, A.; LEVY, O.; STOYANOV, V.; ZETTLEMOYER, L. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: JURAFSKY, D.; CHAI, J.; SCHLUTER, N.; TETREAULT, J. (Ed.). *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. p. 7871–7880. Available at: ⟨https://aclanthology.org/2020.acl-main.703⟩.

LEWIS, P.; PEREZ, E.; PIKTUS, A.; PETRONI, F.; KARPUKHIN, V.; GOYAL, N.; KüTTLER, H.; LEWIS, M.; YIH, W.-t.; ROCKTäSCHEL, T.; RIEDEL, S.; KIELA, D. Retrieval-augmented generation for knowledge-intensive nlp tasks. In: *Advances in Neural Information Processing Systems*. [S.l.]: Curran Associates, Inc., 2020. v. 33, p. 9459–9474.

LEWIS, P.; WU, Y.; LIU, L.; MINERVINI, P.; KÜTTLER, H.; PIKTUS, A.; STENETORP, P.; RIEDEL, S. PAQ: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, MIT Press, Cambridge, MA, v. 9, p. 1098–1115, 2021. Available at: ⟨https://aclanthology.org/2021.tacl-1.65⟩.

LI, A. H.; NG, P.; XU, P.; ZHU, H.; WANG, Z.; XIANG, B. Dual reader-parser on hybrid textual and tabular evidence for open domain question answering. In: *Annual Meeting of the Association for Computational Linguistics and the Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. [S.l.]: Association for Computational Linguistics, 2021. p. 4078–4088.

LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. Available at: ⟨http://arxiv.org/abs/1907.11692⟩.

MA, K.; CHENG, H.; LIU, X.; NYBERG, E.; GAO, J. Open-domain question answering via chain of reasoning over heterogeneous knowledge. In: *Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022. p. 5360–5374. Available at: ⟨https://aclanthology.org/2022.findings-emnlp.392⟩.

MA, K.; CHENG, H.; ZHANG, Y.; LIU, X.; NYBERG, E.; GAO, J. Chain-of-skills: A configurable model for open-domain question answering. In: ROGERS, A.; BOYD-GRABER, J.; OKAZAKI, N. (Ed.). *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, 2023. p. 1599–1618. Available at: ⟨https://aclanthology.org/2023.acl-long.89⟩.

MATOS, V.; GRAVA, R.; TAVARES, R.; JOSé, M.; PIROZELLI, P.; BRANDãO, A.; PERES, S.; COZMAN, F. Coordination within conversational agents with multiple sources. In: *Encontro Nacional de Inteligência Artificial e Computacional*. Porto Alegre, RS, Brasil: SBC, 2023. p. 939–953. ISSN 2763-9061. Available at: ⟨https://sol.sbc.org.br/index.php/eniac/article/view/25755⟩.

MNIH, V.; BADIA, A. P.; MIRZA, M.; GRAVES, A.; LILLICRAP, T.; HARLEY, T.; SILVER, D.; KAVUKCUOGLU, K. Asynchronous methods for deep reinforcement learning. In: *International Conference on Machine Learning*. New York, New York, USA: PMLR, 2016. (Proceedings of Machine Learning Research, v. 48), p. 1928–1937. Available at: ⟨https://proceedings.mlr.press/v48/mniha16.html⟩.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; GRAVES, A.; ANTONOGLOU, I.; WIERSTRA, D.; RIEDMILLER, M. A. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. Available at: ⟨http://arxiv.org/abs/1312.5602⟩.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G.; PETERSEN, S.; BEATTIE, C.; SADIK, A.; ANTONOGLOU, I.; KING, H.; KUMARAN, D.; WIERSTRA, D.; LEGG, S.; HASSABIS, D. Human-level control through deep reinforcement learning. *Nature*, Nature Publishing

Group, v. 518, n. 7540, p. 529–533, 2015. ISSN 14764687. Available at: ⟨http://dx.doi.org/10.1038/nature14236⟩.

NGUYEN, T.; ROSENBERG, M.; SONG, X.; GAO, J.; TIWARY, S.; MAJUMDER, R.; DENG, L. MS MARCO: A human generated MAchine reading COmprehension dataset. *CEUR Workshop Proc.*, v. 1773, p. 1–10, 2016. ISSN 16130073.

OUYANG, L.; WU, J.; JIANG, X.; ALMEIDA, D.; WAINWRIGHT, C.; MISHKIN, P.; ZHANG, C.; AGARWAL, S.; SLAMA, K.; RAY, A.; SCHULMAN, J.; HILTON, J.; KELTON, F.; MILLER, L.; SIMENS, M.; ASKELL, A.; WELINDER, P.; CHRISTIANO, P. F.; LEIKE, J.; LOWE, R. Training language models to follow instructions with human feedback. In: KOYEJO, S.; MOHAMED, S.; AGARWAL, A.; BELGRAVE, D.; CHO, K.; OH, A. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2022. v. 35, p. 27730–27744. Available at: ⟨https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf⟩.

PARTALIDOU, E.; CHRISTOU, D.; TSOUMAKAS, G. Improving zero-shot entity retrieval through effective dense representations. In: *Hellenic Conference on Artificial Intelligence*. New York, NY, USA: Association for Computing Machinery, 2022. (SETN '22). ISBN 9781450395977. Available at: ⟨https://doi.org/10.1145/3549737.3549771⟩.

PASCHOAL, A. F. A.; PIROZELLI, P.; FREIRE, V.; DELGADO, K. V.; PERES, S. M.; JOSé, M. M.; NAKASATO, F.; OLIVEIRA, A. S.; aO, A. A. F. B.; COSTA, A. H. R.; COZMAN, F. G. Pirá: A bilingual portuguese-english dataset for question-answering about the ocean. In: *ACM International Conference on Information & Knowledge Management*. New York, NY, USA: Association for Computing Machinery, 2021. p. 4544–4553. ISBN 9781450384469. Available at: ⟨https://doi.org/10.1145/3459637.3482012⟩.

PIROZELLI, P.; CASTRO, A. B. R.; OLIVEIRA, A. L. C. de; OLIVEIRA, A. S.; CAçãO, F. N.; SILVEIRA, I. C.; CAMPOS, J. G. M.; MOTHEO, L. C.; FIGUEIREDO, L. F.; PELLICER, L. F. A. O.; JOSé, M. A.; JOSé, M. M.; LIGABUE, P. de M.; GRAVA, R. S.; TAVARES, R. M.; MATOS, V. B.; SYM, Y. V.; COSTA, A. H. R.; BRANDãO, A. A. F.; MAUá, D. D.; COZMAN, F. G.; PERES, S. M. The blue amazon brain (blab): A modular architecture of services about the brazilian maritime territory. In: *AI: Modeling Ocean and Climate Change Workshop @ International Joint Conference on Artificial Intelligence*. [s.n.], 2022. p. 1–11. Available at: ⟨https://arxiv.org/pdf/2209.07928.pdf⟩.

PIROZELLI, P.; JOSé, M. M.; SILVEIRA, I.; NAKASATO, F.; PERES, S. M.; BRANDãO, A. A. F.; COSTA, A. H. R.; COZMAN, F. G. *Benchmarks for Pirá 2.0, a Reading Comprehension Dataset about the Ocean, the Brazilian Coast, and Climate Change*. 2023.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. et al. Improving language understanding by generative pre-training. OpenAI, 2018.

RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D.; SUTSKEVER, I. Language models are unsupervised multitask learners. In: . [S.l.: s.n.], 2019.

RAFFEL, C.; SHAZEER, N.; ROBERTS, A.; LEE, K.; NARANG, S.; MATENA, M.; ZHOU, Y.; LI, W.; LIU, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, v. 21, n. 140, p. 1–67, 2020. Available at: ⟨http://jmlr.org/papers/v21/20-074.html⟩.

RAFFIN, A.; HILL, A.; GLEAVE, A.; KANERVISTO, A.; ERNESTUS, M.; DORMANN, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, v. 22, n. 268, p. 1–8, 2021. Available at: ⟨http://jmlr.org/papers/v22/20-1364.html⟩.

RAJPURKAR, P.; ZHANG, J.; LOPYREV, K.; LIANG, P. SQuAD: 100,000+ questions for machine comprehension of text. *Conference on Empirical Methods in Natural Language Processing, Proceedings*, n. ii, p. 2383–2392, 2016.

RAMAMURTHY, R.; SIFA, R.; BAUCKHAGE, C. *NLPGym – A toolkit for evaluating RL agents on Natural Language Processing Tasks*. 2020.

ROBERTSON, S.; ZARAGOZA, H. *The probabilistic relevance framework: BM25 and beyond*. [S.l.: s.n.], 2009. v. 3. 333–389 p. ISSN 15540669. ISBN 1500000019.

SANH, V.; WEBSON, A.; RAFFEL, C.; BACH, S.; SUTAWIKA, L.; ALYAFEAI, Z.; CHAFFIN, A.; STIEGLER, A.; RAJA, A.; DEY, M.; BARI, M. S.; XU, C.; THAKKER, U.; SHARMA, S. S.; SZCZECHLA, E.; KIM, T.; CHHABLANI, G.; NAYAK, N.; DATTA, D.; CHANG, J.; JIANG, M. T.-J.; WANG, H.; MANICA, M.; SHEN, S.; YONG, Z. X.; PANDEY, H.; BAWDEN, R.; WANG, T.; NEERAJ, T.; ROZEN, J.; SHARMA, A.; SANTILLI, A.; FEVRY, T.; FRIES, J. A.; TEEHAN, R.; SCAO, T. L.; BIDERMAN, S.; GAO, L.; WOLF, T.; RUSH, A. M. Multitask prompted training enables zero-shot task generalization. In: *International Conference on Learning Representations*. [s.n.], 2022. Available at: ⟨https://openreview.net/forum?id=9Vrb9D0WI4⟩.

SCHULMAN, J.; LEVINE, S.; ABBEEL, P.; JORDAN, M.; MORITZ, P. Trust region policy optimization. In: *International Conference on Machine Learning*. Lille, France: PMLR, 2015. (Machine Learning Research, v. 37), p. 1889–1897. Available at: ⟨https://proceedings.mlr.press/v37/schulman15.html⟩.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. Available at: ⟨http://arxiv.org/abs/1707.06347⟩.

SONG, K.; TAN, X.; QIN, T.; LU, J.; LIU, T.-Y. MPNet: masked and permuted pre-training for language understanding. In: *International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2020. ISBN 9781713829546.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. BERTimbau: pretrained BERT models for Brazilian Portuguese. In: *Brazilian Conference on Intelligent Systems*. [S.l.: s.n.], 2020.

SPINDOLA, S.; JOSÉ, M. M.; OLIVEIRA, A. S.; CAÇÃO, F. N.; COZMAN, F. G. Interpretability of attention mechanisms in a portuguese-based question answering system about the blue amazon. In: *Encontro Nacional de Inteliência Artificial e Computacional*. Porto Alegre, RS, Brasil: SBC, 2021. p. 775–786. ISSN 0000-0000. Available at: ⟨https://sol.sbc.org.br/index.php/eniac/article/view/18302⟩.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction.* Cambridge, MA, USA: A Bradford Book, 2018. ISBN 0262039249.

SYM, Y.; CAMPOS, J.; JOSé, M.; COZMAN, F. Comparing computational architectures for automated journalism. In: *Encontro Nacional de Inteligência Artificial e Computacional.* Porto Alegre, RS, Brasil: SBC, 2022. p. 377–388. ISSN 2763-9061. Available at: ⟨https://sol.sbc.org.br/index.php/eniac/article/view/22797⟩.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. *Advances in Neural Information Processing Systems*, v. 2017-Decem, p. 5999–6009, 2017. ISSN 10495258.

WANG, P.; SHI, T.; REDDY, C. K. Text-to-SQL generation for question answering on electronic medical records. In: *Web Conference.* New York, NY, USA: Association for Computing Machinery, 2020. p. 350–361. ISBN 9781450370233. Available at: ⟨https://doi.org/10.1145/3366423.3380120⟩.

WANG, S.; YU, M.; GUO, X.; WANG, Z.; KLINGER, T.; ZHANG, W.; CHANG, S.; TESAURO, G.; ZHOU, B.; JIANG, J. R3: reinforced ranker-reader for open-domain question answering. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence.* [S.l.]: AAAI Press, 2018. ISBN 978-1-57735-800-8.

WANG, Y.; JIN, H. A deep reinforcement learning based multi-step coarse to fine question answering (MSCQA) system. *AAAI Conference on Artificial Intelligence*, v. 33, n. 01, p. 7224–7232, 2019. Available at: ⟨https://ojs.aaai.org/index.php/AAAI/article/view/4707⟩.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. *Machine Learning*, v. 8, n. 3, p. 279–292, maio 1992. ISSN 1573-0565. Available at: ⟨https://doi.org/10.1007/BF00992698⟩.

WILLIAMS, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, Springer, v. 8, p. 229–256, 1992.

XUE, L.; CONSTANT, N.; ROBERTS, A.; KALE, M.; AL-RFOU, R.; SIDDHANT, A.; BARUA, A.; RAFFEL, C. mT5: A massively multilingual pre-trained text-to-text transformer. In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* [S.l.]: Association for Computational Linguistics, 2021. p. 483–498.

YANG, Z.; DAI, Z.; YANG, Y.; CARBONELL, J.; SALAKHUTDINOV, R. R.; LE, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In: *Advances in Neural Information Processing Systems.* Curran Associates, Inc., 2019. v. 32. Available at: ⟨https://proceedings.neurips.cc/paper_files/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf⟩.

YANG, Z.; QI, P.; ZHANG, S.; BENGIO, Y.; COHEN, W.; SALAKHUTDINOV, R.; MANNING, C. D. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In: *Conference on Empirical Methods in Natural Language Processing.* Brussels, Belgium: Association for Computational Linguistics, 2018. p. 2369–2380. Available at: ⟨https://aclanthology.org/D18-1259⟩.

ZHANG, Z.; YANG, J.; ZHAO, H. Retrospective reader for machine reading comprehension. In: *AAAI Conference on Artificial Intelligence*. [s.n.], 2020. Available at: ⟨https://api.semanticscholar.org/CorpusID:210920624⟩.

ZHONG, W.; HUANG, J.; LIU, Q.; ZHOU, M.; WANG, J.; YIN, J.; DUAN, N. Reasoning over hybrid chain for table-and-text open domain question answering. In: *International Joint Conference on Artificial Intelligence*. [s.n.], 2022. p. 4531–4537. Main Track. Available at: ⟨https://doi.org/10.24963/ijcai.2022/629⟩.

APPENDIX A – PUBLISHED PAPERS

During the master's program, contributions to six additional papers were made, supplementing those outlined in Section 1.3. These endeavors provided valuable insights into QA datasets, transformer neural networks, and DRL:

- The author contributed to the development of Pirá, one of the first bilingual Question Answering datasets in Portuguese and English. This dataset focuses on the Blue Amazon or the Brazilian maritime coast, containing curated question-answer pairs extracted from scientific article abstracts. Our research outcomes were showcased at the resources track of the "ACM International Conference on Information and Knowledge Management" (CIKM 2021) (PASCHOAL et al., 2021).
  Participating in this project yielded a valuable lessons about Question Answering dataset generation and the critical distinctions between various types. This knowledge is instrumental in shaping the design of suitable architectures and constructing efficient models.

- During the research, the author worked on the control of a robotic manipulator using DQN and Episodic REINFORCE. This work was published in the journal "Revista Brasileira de Computação Aplicada" (RBCA) in November 2021 (COTRIM; JOSÉ; CABRAL, 2021).
  Although this application is not directly linked to QA, it provided essential insights into model-free DRL theory and the practical aspects of building an environment and training a suitable agent.

- The author participated in a paper published at "Encontro Nacional de Inteligência Artificial e Computacional" (ENIAC 2021) (SPINDOLA et al., 2021), centered around creating a small QA dataset about the Blue Amazon and interpretting transformers models. We applied an off-the-shelf QA system based on a fine-tuned BERTimbau Model (SOUZA; NOGUEIRA; LOTUFO, 2020) and explored how visualizing attention weights can aid in interpreting the system's responses.

The development of this work brought important knowledge about transformer neural networks and their layers of attention in a QA applications. The ability to interpret such models is essential for identifying areas of improvement and comprehending potential pitfalls, thereby contributing to the overall refinement of QA systems

- The author participated in an article outlining the initial stages of the "BLue Amazon Brain" (BLAB), an artificial agent designed for conversation about the Brazilian maritime territory. It encompasses a range of research on the topic, including QA Systems and conversational agents. This work was presented at the "AI: Modeling Oceans and Climate Change (IJCAI-ECAI), 2022" (PIROZELLI et al., 2022).
- The author contributed to the development to a project focused on creating a journalist robot for reporting on the Blue Amazon using three of the most important Natural Language Generation techniques: template-based, pipeline-based, and neural end-to-end. Results were published at "Encontro Nacional de Inteligência Artificial e Computacional" (ENIAC 22) (SYM et al., 2022).
- The author participated in a research paper titled "Coordination within Conversational Agents with Multiple Sources" which describes how a Large Language Model can effectively coordinate various sources and systems to develop a conversational agent. This work was presented in the proceedings of "XIX Encontro Nacional de Inteligência Artificial e Computacional" (ENIAC 23) (MATOS et al., 2023).

There is also an approved paper that is yet to be published:

- As a continuation of the construction of the dataset Pirá mentioned above, the author participated in a paper focused on improving it and creating benchmarks and tests for several tasks, including closed generative question answering, machine reading comprehension, information retrieval, open question answering, answer triggering, and multiple choice question answering. This paper has been accepted by the journal "Data Intelligence" and is currently available on ArXiV (PIROZELLI et al., 2023).