Incorporating Structured Knowledge into PAQ

Candidate Number: MJRJ2[1]

MSc Machine Learning

Dr. Pasquale Minervini

Submission date: 27 September 2021

[1]**Disclaimer:** This report is submitted as part requirement for the MSc Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report will be distributed to the internal and external examiners, but thereafter may not be copied or distributed except with permission from the author.

Abstract

This thesis is primarily centred around the Natural Language Processing task of Open-Domain Question Answering (ODQA). One system that stands out is PAQ-RePAQ – which combines a collection (PAQ) of 65 million automatically generated Question-Answer (QA) pairs based on Wikipedia with a fast QA-pair retrieval system (RePAQ). This system achieves competitive performance on ODQA benchmarks with inference speeds up to 1,000 times faster than the pre-dominant retrieve-reader systems. PAQ-RePAQ is clearly a powerful system that can be used for large scale production settings, however, it has knowledge gaps that could potentially be improved. The aim of this thesis is to find methods of improvement specifically through the incorporation of data from structured knowledge sources as PAQ currently only contains data from the unstructured knowledge source Wikipedia.

Before attempting to make any improvement we first perform a systematic analysis of RePAQ to better understand the reasons for mistakes. Through a manual error analysis on RePAQ's performance on Natural Questions (NQ), we find that a significant amount of mistakes (46%) are due to incorrect annotations and ambiguous questions in NQ as well as the strict nature of the Exact Match (EM) evaluation metric. To help separate the signal from the noise in our systematic analysis of RePAQ, we propose a new metric EM_{norm} that reduces the amount of incorrectly labelled mistakes by automatically normalizing names, dates, and amounts. We find that the remaining source of error in RePAQ is due to incomplete coverage of PAQ. By using question clustering, entity linking and knowledge base lookups we automatically identify a number of topic areas in PAQ where coverage is lacking the most: areas such as cast members of movies, performers of songs and release dates of episodes. Following so, we demonstrate that with a simple yet effective approach of templating, we can integrate structured knowledge from Wikidata, DBpedia and IMDb about these areas into PAQ.

Our findings indicate that this improves RePAQ's performance on two popular ODQA benchmarks, Natural Questions and TriviaQA, by 1.02 and 0.41 points EM-5, respectively. These findings, therefore, suggest that the performance gains from incorporating structured knowledge into PAQ for ODQA are possible but only marginally and dependent on the knowledge gap areas identified.

All source code and experimental results are available at `https://github.com/thymo/2paq`.

Contents

1 Introduction **3**
- 1.1 Motivation . . . 3
- 1.2 Research Aims and Structure . . . 4

2 Background **5**
- 2.1 Question Answering . . . 5
  - 2.1.1 Question-Answer Formats . . . 5
  - 2.1.2 Open-Domain Question Answering vs Machine Reading Comprehension . . . 6
  - 2.1.3 Types of Knowledge Sources . . . 6
  - 2.1.4 Open vs Closed-Book Question Answering . . . 7
  - 2.1.5 Simple Factoid vs Multi-Hop Question Answering . . . 8
- 2.2 PAQ and RePAQ . . . 8
  - 2.2.1 QA-Pair Generation . . . 9
  - 2.2.2 QA-Pair Retrieval . . . 10
  - 2.2.3 Coverage . . . 10
- 2.3 Datasets . . . 11
  - 2.3.1 Natural Questions . . . 11
  - 2.3.2 TriviaQA . . . 11
  - 2.3.3 Evaluation Metric . . . 11
- 2.4 Knowledge Bases . . . 12
  - 2.4.1 Wikidata . . . 12
  - 2.4.2 DBpedia . . . 12
- 2.5 Entity Linking . . . 13
  - 2.5.1 GENRE . . . 13

3 Analysing Knowledge Gaps in PAQ **14**
- 3.1 Manual Error Analysis . . . 14
  - 3.1.1 Top 1 vs Top 5 Retrieved QA-pairs . . . 14
  - 3.1.2 Mistakes of Top-5 Retrieved QA-pairs . . . 19
- 3.2 Normalising Answers During Evaluation . . . 23
  - 3.2.1 Name Normalization . . . 23
  - 3.2.2 Date Normalization . . . 24
  - 3.2.3 Amount Normalization . . . 25

3.2.4 Results . . . 25
3.3 Identifying Hard Question Templates . . . 25
3.3.1 Method . . . 26
3.3.2 Natural Questions . . . 26
3.3.3 TriviaQA . . . 27
3.4 Identifying Useful Knowledge Base Topic Areas . . . 28
3.4.1 Method . . . 28
3.4.2 Results on Natural Questions . . . 29
3.4.3 Results TriviaQA . . . 29

4 Incorporating Structured Knowledge **31**
4.1 QA-Pair Generation . . . 31
4.1.1 Wikidata . . . 31
4.1.2 DBpedia . . . 33
4.1.3 IMDb . . . 33
4.2 Experiments . . . 34
4.2.1 Method . . . 34
4.2.2 Results with Wikidata . . . 34
4.2.3 Results with DBpedia . . . 35
4.2.4 Results with IMDb . . . 37
4.3 Final Results . . . 38

5 Conclusion **39**
5.1 Summary . . . 39
5.2 Future Work . . . 40

A Mistake Patterns NQ **46**

B Mistake Patterns TriviaQA **48**

C KB Predicates Statistics Natural Questions **50**
C.1 Wikidata . . . 50
C.2 DBpedia . . . 51

D KB Predicates Statistics TriviaQA **54**
D.1 Wikidata . . . 54
D.2 DBpedia . . . 56

List of Tables

3.1 Reasons for difference EM top-1 and EM top-5 retrieved QA-pairs 15
3.2 Examples of RePAQ retrieval inaccuracies 15
3.3 Examples of ambiguous questions 16
3.4 Examples of correct predictions classified as incorrect because of a different granularity . . 17
3.5 Percentage of mistakes due to the difference in granularity per question type 17
3.6 Examples of correct predictions classified as incorrect because of a different form or spelling 18
3.7 Examples of incorrect gold labels 18
3.8 Examples of incorrect duplicates in PAQ 19
3.9 Overview of mistake patterns with frequencies 19
3.10 Examples questions where PAQ lacks coverage 20
3.11 Example ambiguous questions, where the answer depends on time asked/source 21
3.12 Example of questions with incorrect gold labels 22
3.13 Example of different form answers: names 22
3.14 Example of different form answers: numbers 22
3.15 Example of different granularity answers: dates 23
3.16 Examples of automatic normalization of names 24
3.17 The difference in EM and normalized EM scores across ODQA systems on Natural Questions 25
3.18 Example of questions where the answer is automatically found in Wikidata 28
3.19 Wikidata predicate statistics Natural Questions 29
3.20 DBpedia predicate statistics Natural Questions 30
3.21 Wikidata predicate statistics TriviaQA 30
3.22 DBpedia predicate statistics TriviaQA 30

4.1 Wikidata QA-pair templates 32
4.2 DBpedia predicate QA-pair templates 33
4.3 IMDb QA pair templates 34
4.4 Results including QA pairs from Wikidata 35
4.5 Results including filtered QA-pairs from Wikidata 36
4.6 Results including QA pairs from DBpedia 36
4.7 Examples of questions referring to song by slightly different name 37
4.8 Results including QA pairs from IMDb 37
4.9 Examples of questions referring to character by something else then their name 37
4.10 Results including selected structured data 38

Chapter 1

Introduction

1.1 Motivation

Open-Domain Question Answering (ODQA) is a Natural Language Processing (NLP) task that gained much attention in recent years due to its numerous practical applications. Most prominently, used on a large scale for modern search engines, personal assistants, and chatbots for answering simple fact-based questions. A more specific example of its use would be for example, answering the questions *"What is the highest mountain in the world?"* or *"Who plays Donna on the tv show Suits?"*. The open-domain nature of this task entails that the context of the answer is not provided and can come from any source.

The prevalent ODQA-systems are based on a retrieve-reader approach in which a retriever retrieves the top-k relevant text passages and a reader then extracts or generates the answer from these passages (Izacard and Grave, 2021; Oguz et al., 2020). Even though these models provide state-of-the-art results on leading benchmarks, they are not the most efficient at inference time, mostly due to the computationally expensive passage reader component.

A novel approach introduced by Lewis et al. (2021b) deviates from the conventional passage retrieve-reader models and instead uses a fast QA-pair retriever (RePAQ) in combination with a large corpus of 65 million automatically generated QA-pairs called 'Probably Asked Questions' or PAQ. This system is referred to as PAQ-RePAQ and won two tracks of the EfficientQA 2020 competition (Min et al., 2020) and proved to be competitive with existing retrieve models while being significantly faster at inference time. The faster inference time, along with high interpretability, and prediction confidence scores (i.e. it knows when it doesn't know) makes this model attractive for large-scale production use.

However, on current ODQA benchmarks, it was found that PAQ's coverage is still lacking; in 32% of the cases, the correct answers of test questions were not found in the top-50 retrieved PAQ QA-pairs. This lead to the suggestion that a potential area of improvement is increasing the coverage of PAQ. The 65 million QA-pairs in PAQ were generated based on text passages from Wikipedia. In this thesis, we investigate whether we can extend PAQ by leveraging information available in structured knowledge sources such as knowledge bases and databases. More specifically, we evaluate whether we can improve coverage and downstream performance of answering simple factoid questions by converting structured data into question-answer pairs and concatenating them to the existing PAQ question-answer pairs. We hypothesise that as long as structured data can be converted to textual question-answer pair representation, we are able to leverage the information available in these structured knowledge sources while still relying on the advantages of using a fast and confident-aware QA-retriever. Such a unified QA-retriever system could therefore leverage both free form text and structured data for a larger coverage of simple factoid questions.

1.2 Research Aims and Structure

The main aim of this thesis is to find out whether RePAQ's Open-Domain Questions Answering (ODQA) performance can be improved by enriching PAQ with QA-pairs generated from structured knowledge sources. This is because PAQ's current collection of QA-pairs has been based on only unstructured knowledge sources until now.

This thesis is structured as follows. Chapter 2 is dedicated to providing contextual background information about Question Answering, the PAQ-RePAQ architecture, ODQA datasets, public Knowledge Bases and entity linking. This is to provide a clearer understanding of key factors before proceeding to our analysis and experiments. Following suit, Chapter 3 will go on to a systematic analysis of RePAQ and PAQ to provide a better understanding of the system's current weaknesses and potential areas for improvement. First, Section 3.1 shows the findings of a manual error analysis and gives insight into why RePAQ's is making the mistakes it is making. Then, Section 3.2 introduces a new evaluation metric EM_{norm} that helps in focusing on analysing mistakes that are truly incorrect, not due to of a small mismatch in the answer form or granularity. Next, section 3.3 shows how question clustering can be used to automatically find common mistake areas of RePAQ. Finally, section 3.4 shows how another method, involving entity linking and knowledge bases, can give even more insight into the knowledge gaps of PAQ. Bringing us to Chapter 4, which is focused on closing these knowledge gaps by leveraging data available in structure knowledge sources. First, Section 4.1 explains how structured knowledge can be incorporated into PAQ. Then, Section 4.2 analyzes how RePAQ's performance on Natural Questions and TriviaQA changes after incorporating structured knowledge into PAQ. Finally bringing us to Chapter 5 which will present a summary of the work carried out as well as an overview of potential future work to improve the limitations of this thesis.

Chapter 2

Background

2.1 Question Answering

As described in the introduction, this thesis is primarily concerned with the question answering system of PAQ and RePAQ. The goal of a Question Answering system is to automatically answer any question asked by a user in natural language. These systems have wide-ranging applications in web search engines, customer service chatbots, and personal assistants like Siri, Alexa and Google Assistant. They are the systems that enable a user to quickly get an answer to their specific questions without browsing and reading through a large collection of pages. The Question Asking systems can be categorized into multiple formats depending on the desired setting: question-answer format, the availability of context, the type of knowledge source(s), access to knowledge source at inference time and complexity of supported questions. We give a brief overview of all these settings to better contextualize where the RePAQ and PAQ question answering system fits.

2.1.1 Question-Answer Formats

To evaluate the progress of Question Answering (QA) systems, there are a multitude of different benchmarks and data sets to test these systems on. These benchmarks test different question-answer formats and reasoning abilities. For example, Visual Question Answering datasets like CLEVR (Johnson et al., 2017b) in which a system has to answer question about a visual scene. On the other hand, Textual QA datasets like SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017) and Natural Questions (Kwiatkowski et al., 2019) focus on extractive question answering in which the answer is simply a span in a context text and the system simply has to extract the right answer. Whereas, abstractive QA datasets like NarrativeQA require the system to generate an answer based on the context. Here, the correct answer is not explicitly found in the context; it should be inferred. Finally, there are multiple-choice datasets like ARC (Clark et al., 2018) where the system is given a set of answer options. Even though many models are trained specifically for one question-answer format, a unified model using multi-task training on multiple data sets with different formats seems to outperform non-general models (Khashabi et al., 2020).

2.1.2 Open-Domain Question Answering vs Machine Reading Comprehension

Based on the availability of context when answering a question we can divide QA systems in two categories: Open-Domain Question Answering and Machine Reading Comprehension systems. Open-Domain Question Answering (ODQA) refers to the setting in which no specific context is given to the system (Prager, 2006). It should be able to answer all kinds of simple factoid questions about any topic. For example, it should be able to answer *"Who played James Bond in the 2008 movie Casino Royale?"* as well as *"When was the first manned flight?"*. These questions are given without any relevant text article or passage that contains the answers. If context, for example, an article about the Wright Brothers is given with the question about the first manned flight, this setting would be a Machine Reading Comprehension (MRC) task. MRC systems are trained to answer questions about a particular context, often a text passage. This mirrors the setting of language proficiency exams where the test taker is evaluated on his ability to understand and comprehend written text (Zhu et al., 2021). This, however, is not always a realistic case in real-world settings where the user seeks to obtain the answer for a question without giving any particular context document. The ODQA and MRC tasks do overlap and progress on MRC have led to subsequent improvement on ODQA benchmarks (Zhu et al., 2021). The predominant ODQA systems break up the problem into two stages, a retrieval stage to get the relevant context passages, and a reading stage to read the context passages and answer the question like the MRC task (Min et al., 2020). There are two main variants of these Retrieve-Reader systems depending on the type of reader: extractive models like DrQA (Chen et al., 2017) and ORQA (Lee et al., 2019), in that, select an answer span from one of the context passages, generative models like RAG (Lewis et al., 2020b) and FiD (Izacard and Grave, 2021) that condition on the context passages and use an autoregressive language model to generate the answer.

2.1.3 Types of Knowledge Sources

To answer any kind of question, human beings rely on a wide variety of reference knowledge sources. People are able to get information from Wikipedia, books, manuals, scientific journals, legal databases, explainer videos, excel spreadsheets, and many other sources. Human beings are versatile and able to leverage the right knowledge source for the task or question at hand, irrespective of the knowledge source type. In Open Domain Question Answering, systems often are designed for one specific knowledge source type. Recent works show that general models unifying different knowledge source types can result in state-of-the-art performance on modern ODQA datasets (Khashabi et al., 2020). Knowledge sources can be categorized based on the data type: unstructured, structured, semi-structured and structured.

Unstructured With unstructured knowledge sources, we refer to collections of knowledge where there is no predefined schema or structure in how the information is represented. The knowledge is found in free-form text that humans are able to comprehend by reading them. Examples of unstructured knowledge sources are Wikipedia, news articles, books and internal company knowledge bases. Due to progress in MRC and IR models, unstructured textual data can now be understood and leveraged by machines without resorting to inaccurate heuristic parsing techniques. Even though some ODQA systems use the entire web as their knowledge base (Talmor and Berant, 2018), most recent ODQA system follow the work of Chen et al. (2017) and use Wikipedia as their main knowledge source (Seo et al., 2019; Lee et al., 2019; Izacard and Grave, 2021; Lewis et al., 2020b). These models differ in how they split up and index the Wikipedia corpus: either full Wikipedia pages, (100 word) passages or individual phrases. Instead of considering the whole Wikipedia corpus some memory-efficient models designed for the EfficientQA competition prune the corpus to only consider informative pages and passages based on a binary classifier or popularity filter heuristic (Min et al., 2020). Most ODQA systems that use Wikipedia as the knowledge source, first pre-process an archived Wikipedia dump of all the pages to filter out any semi-structured data such as lists, tables and info-boxes (Chen et al., 2017; Karpukhin et al., 2020).

Semi-Structured Semi-structured data formats such as lists, tables and info-boxes offer a concise and flexible way to present knowledge to a human reader. Wikipedia has over 3 million tables that contain many useful facts that are not covered in the textual passages, which could help answer long-tail questions about specific facts (Oguz et al., 2020). Information about release dates of specific TV show episodes, rankings of sports competitions, characters and their actors of movies are all found in Wikipedia lists and tables but often not found with high coverage in text passages. Instead of filtering them out and only considering plain the text passages like general textual ODQA models, ODQA models such as HYBRIDER (Chen et al., 2020), DTR+TAPAS (Herzig et al., 2021) and UniK-QA (Oguz et al., 2020) combine Wikipedia text passages, semi-structured tables and info-boxes as the knowledge source. As the schemas of the tables are not predefined and can be inconsistent across multiple pages, these systems cannot rely on a structured query language. Instead, these models first linearize the tables to text and use specially pre-trained MRC models to retrieve the answer.

Structured Knowledge sources that represent data using a predefined schema are referred to as structured knowledge sources. This includes relational databases as well as knowledge bases such as Wikidata (Vrandecic and Krötzsch, 2014) and DBPedia (Auer et al., 2007). Questing answering systems that use a knowledge base (KB) as the primary knowledge source are referred to as KBQA systems. A KB can be considered as a knowledge graph where Predicates between and properties of millions of real-world entities are stored. Most KBQA systems first start with an entity linking step, where they identify the main entity from the question, (Lan et al., 2021). For example for the question *"When does Fifty Shades of Grey come out?"* an entity linker identifies *"fifty shades of grey"* as the topic entity. Then most KBQA systems can be categorized under two types. Semantic parsing-based systems parse the natural language question and translate it into a structured query (e.g. using the SPARQL query language) that can be used to traverse the knowledge graph and retrieve the right answer (Berant et al., 2013). Information retrieval-based systems first extract a subgraph from the KB based on the neighbourhood of the topic entity and consider each neighbouring node as a potential answer (Bordes et al., 2014). These candidate answers are then ranked and the highest scoring candidate is selected as the output answer.

2.1.4 Open vs Closed-Book Question Answering

When students take exams there are generally two settings, an open-book setting, in which students have access to reference material while taking the exam or a closed-book setting where they are not allowed any access to external material and students have to solely rely on their internal memory and reasoning capabilities. In Open Domain Question Answering researchers make a similar distinction depending on whether the ODQA system has access to external knowledge or needs to know everything internally during test time (Roberts et al., 2020). In the open-book setting systems models can retrieve information from external knowledge sources such as Knowledge Bases, encyclopedias and databases as mentioned in Section 2.1.3.

An advantage of leveraging these external knowledge sources is that knowledge can easily be updated without an expensive retraining phase. Closed-book ODQA models store all knowledge in their parameters. State-of-the-art language models such as BERT, BART, and T5 have been shown to internalize real-world facts during pre-training which can be prompted to answer simple factoid questions (Petroni et al., 2019; Roberts et al., 2020). To provide competitive performance on ODQA benchmarks, these seq2seq language models are typically fine-tuned on question-answer pairs from ODQA training sets. Lewis et al. (2021a) show that these closed-book models are especially good at answering test questions that overlap with train questions. They suggest that the large capacity of these seq2seq language models (hundred million+ parameters) combined with a more powerful understanding of the question allow it to better recall training questions compared to other models. However, these closed book models perform significantly worse on completely novel questions (Lewis et al., 2021a; Liu et al., 2021). Besides a lack of generalization and lower interpretability, these parametric closed book models are also more expensive to update as they need to be retrained anytime new knowledge needs to be added or existing knowledge needs to be rectified.

2.1.5 Simple Factoid vs Multi-Hop Question Answering

The prevailing ODQA models are effective at answering primarily simple factoid questions where the correct answer can be found explicitly in a single text passage, table cell or KB triplet. For example, *"What is the capital of Zimbabwe?"* or *"Who sings the song Banana Pancakes?"*. As human beings, we are able to pose and answer more complex questions that require multiple steps of reasoning such as *"In what city was Barack Obama's wife born?"* and *"Who has a higher net worth Bill Gates or Jeff Bezos?"*. The answer to these questions can only be inferred by logical reasoning over multiple evidence documents. To challenge the ODQA community to build systems that are able to answer these types of complex questions, new datasets have recently been introduced to specifically test for multi-hop questions. HotpotQA, a multi-hop ODQA dataset introduced by Yang et al. (2018), focuses on questions for which the answer can only be inferred by combining the information of at least two Wikipedia passages. Multi-hop ODQA systems either follow an iterative retrieval process (Xiong et al., 2021) or use a question decomposition strategy to decompose a complex question into simpler sub-questions (Perez et al., 2020). These multi-hop systems tackling complex questions still share and leverage components from the simpler factoid ODQA systems.

2.2 PAQ and RePAQ

As mentioned, this thesis aims to analyse the ODQA system of PAQ and RePAQ introduced by Lewis et al. (2021b) and investigate whether we can leverage structured knowledge to improve coverage and subsequently improve down-stream performance. For greater context, the focus of this section is thus to introduce and explain how PAQ and RePAQ work. PAQ, which stands for *Probably Asked Questions*, is a collection of 65 million automatically generated Question-Answer (QA) pairs based on the Wikipedia corpus. RePAQ is the complementary QA-pair retriever that can be used to answer open-domain questions. When given a new question q_t, RePAQ retrieves the most similar question in PAQ and returns the corresponding answer.

QA-pair retrievers such as RePAQ are much faster at inference time and have higher interpretability compared to the prevalent retrieve-reader systems. Lewis et al. (2021a) showed that QA-pair retrievers are competitive when it comes to answering test questions that are similar to or paraphrases of training questions. In the past, QA-pair retrievers were not able to match the accuracy of state-of-the-art ODQA models, as the QA-pairs they retrieve from consisted only of a limited amount of QA-pairs collected from ODQA training datasets. PAQ was thus created to expand the coverage of QA-pairs and enable a simple QA-pair retriever to be competitive with retrieve-reader models which use the entire Wikipedia corpus as their knowledge source.

2.2.1 QA-Pair Generation

Knowing how PAQ's QA-pair generation processes worked helps in our analysis to understand why PAQ might still be lacking knowledge. The general idea of PAQ is to automatically generate a large collection of question answers pairs over the full Wikipedia corpus. These questions are similar to the questions in the training dataset but have a higher coverage. More formally, Lewis et al. (2021a) assume there is a distribution $P(q, a)$ of QA-pairs, defined over $Q \times A$, with Q as the collection of all possible questions and A as the collection of all possible answers. We do not have access to this distribution, but only have an empirical sample K drawn from P, an ODQA training dataset of QA-pairs. Based on this empirical sample Lewis et al. (2021b) implicitly model the distribution $P(q, a)$ and draw a large sample from it. To actually draw this sample (i.e. generate QA-pairs) Lewis et al. (2021b) uses a textual knowledge corpus C such as Wikipedia and a four-step process consisting of a Passage Selection, Answer Extraction, Question Generation and Global Filtering step.

Passage Selection The first step consists of learning a passage selection distribution $P_s(c)$ that models how likely a user will ask questions about the information contained in a text passage $c \in C$. For example, a specific passage at the end of a Wikipedia page about a niche topic will have a lower probability than for example the introductory passage about a famous actor. To model the distribution P_s, Lewis et al. (2021b) fine-tune a RoBERTa model (Liu et al., 2019) by using passages from Natural Questions (Kwiatkowski et al., 2019) as high probability passages and other random 100-word passages from Wikipedia as low probability passages. Following so, Lewis et al. (2021b) score all 21 million Wikipedia passages with this model and use the top 10 million for the next step.

Answer Extraction The second step consists of learning an answer distribution $P_a(a|c)$ that models how likely token-span a is an answer to a question given context passage c. Lewis et al. (2021b) models this by passing the context passage c through a pre-trained BERT encoder (Devlin et al., 2019), concatenating the embeddings of the first and last token of the answer a and uses that as the input for an MLP to calculate $P_a(a|c)$. At generation time, all possible answer spans up to 30 tokens within a context passage are evaluated, after which the highest probability answer candidates are extracted for the next step.

Question Generation At this point, we would have been able to extract the relevant context passages with highly likely answer candidates. The third step then consists of learning a question generation model $p_q(q|a, c)$ that, given a context passage and an answer candidate, generates a sensible question. For this question generator, Lewis et al. (2021b) fine-tunes the pre-trained BART-base model (Lewis et al., 2020a) on the training set questions from SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017) and Natural Questions (Kwiatkowski et al., 2019). The system then feeds all passages and answer candidates from the previous step to this question-generator and obtain new QA-pairs.

Filtering By the end of this process, Lewis et al. (2021b) were able to generate 279 million unique QA-pairs. However, it is important to note that many of these QA-pairs are not accurate or ambiguous. Therefore, a filter step needs to be applied to obtain a set of 65 million high-quality QA-pairs. This is done by feeding all QA-pairs to FiD (Izacard and Grave, 2021), a state-of-the-art reader-retrieve ODQA model, and removing pairs for which the generated answer does not overlap with the predicted answer.

2.2.2 QA-Pair Retrieval

Now that we have generated the 65 million QA-pairs, there is one component missing before they can can be used for the ODQA task. This brings us to QA-Pair Retrieval systems. How this works is, given test question q and a collections of N QA-pairs $K = \{(q_1, a_1)...(q_N, a_N)\}$, a QA-pair retrieval systems retrieves the most similar QA-pair $(q', a') \in K$ based on a similarity function $f_r(q, q')$ (Lewis et al., 2021b). It then simply returns a' as the predicted answer to test question q'. There are two types of similarity functions: standard Information Retrieval (IR) techniques such as TF-IDF and BM25, and dense IR techniques. TF-IDF and BM25 are extremely efficient but are not able to capture semantic similarities of words. Dense IR techniques, on the other hand, were introduced to solve this issue. They utilize pre-trained language models that are able to understand deep semantic meanings of text.

RePAQ RePAQ, the QA-pair retriever introduced by (Lewis et al., 2021b) follows the dense IR paradigm to retrieve the most relevant QA-pairs. It does this by using an embedding function g_q to embed both the test question q and QA-pair questions $\{q_1, ...q_N\}$. The similarity score function is then given by

$$f_r(q, q') = g_q(q)^T g_q(q')$$

the inner product between the embedding of test question q and QA-pair question q'. At inference time, RePAQ is then able to retrieve the top k QA-pairs (q', a') with the highest inner products. For the question embedding function, g_q Lewis et al. (2021b) use a pre-trained ALBERT model (Lan et al., 2020) and fine-tune it using a latent variable approach similar to RAG (Lewis et al., 2020b). All QA-pairs in PAQ are embedded and indexed before test time so that new test questions can be answered efficiently. Using the open-source Maximum Inner Product Search library FAISS (Johnson et al., 2017a) and the approximate Hierarchical Navigable Small World (HNSW) (Malkov and Yashunin, 2016) indexing technique, RePAQ can answer 100-1000s questions per second. This is magnitudes faster than the latest retrieve-reader ODQA systems.

2.2.3 Coverage

The PAQ and RePAQ combination achieves competitive performance on standard ODQA datasets. However, the system might still have room for improvement. Lewis et al. (2021b) observe that in 32% of the test question, the correct answer is not found in the top 50 QA-pairs retrieved by RePAQ. They suggest that a lack of coverage in PAQ might be the cause of this. In this thesis, we will investigate if a lack of coverage is indeed the reason for this and explore if we can improve the coverage of PAQ by leveraging data from structured knowledge sources.

2.3 Datasets

To evaluate and compare the progress of different Questions Answering systems, the NLP community proposed a number of standardized datasets over the years. These datasets differ in the way they were collected and the type of setting they are testing for (see Section 2.1. For our experiments, we use the popular ODQA datasets Natural Questions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017), the same datasets that Lewis et al. (2021b) used for the evaluation of PAQ and RePAQ.

2.3.1 Natural Questions

The Natural Questions (NQ) dataset comprises a collection of questions searched for on Google by real anonymized users, which have a corresponding answer in a Wikipedia article (Kwiatkowski et al., 2019). This was done as Google automatically-collected these questions from users and annotators then selected the right answer span in a Wikipedia article. These questions have thus been defined as "natural" as they are representative of actual information-seeking questions from online users. In our experiments, we use the ODQA version of the dataset. This version consists of only the question-answer pairs that have short answers (less than 6 tokens) and without any context information. We use the same splits as Lewis et al. (2021b) in our experiments, consisting of 79,168 train, 8,757 development and 3,610 test QA-pairs.

2.3.2 TriviaQA

The TriviaQA (TQA) dataset comprises a collection of question-answer pairs provided by trivia enthusiasts (Joshi et al., 2017). The QA-pairs are obtained by scraping 14 popular trivia and quiz-league websites. These trivia questions, which are organically created by humans to engage other humans are challenging and an interesting benchmark for evaluating QA systems. The answer to these questions can in 93% of the cases be found on related Wikipedia page. If a correct answer corresponds to a Wikipedia entry, all possibles aliases are given as accepted answers too. In our experiments, we use the same splits as Lewis et al. (2021b), consisting of 79,168 train, 8,837 development and 11,313 test QA-pairs.

2.3.3 Evaluation Metric

We follow the ODQA community and use Exact Match (EM) as our base evaluation metric. EM is an automatic evaluation metric that assigns a score of 1 if the predicted answer exactly matches one of the gold/reference labels and 0 otherwise. The reported EM score is the percentage of exact match answers. Most authors use a minor normalization function created by Chen et al. (2017) that transforms answers to lowercase and removes extra white space, punctuation and the articles "a, an, the". The EM metric is a very strict metric that will not capture semantically equivalent answers. For example, with gold label "15", and prediction "fifteen" under EM this would be evaluated as incorrect. Min et al. (2020) note on this harsh nature of the automatic evaluation of EM too and compare it to manual human evaluation. They observe that a substantial amount of answers would have been accepted as valid answers by humans compared to EM, though both human evaluation and EM are consistent in ranking the performance of different ODQA systems. Nevertheless, we find it can be useful to have an automatic evaluation metric closer to manual human evaluation and propose a more normalized EM metric in Section 3.2.

2.4 Knowledge Bases

In this thesis, we investigate whether we can extend the coverage of PAQ by leveraging structured data available in large-scale public Knowledge Bases (KB). These KB's contain hundreds of millions of factual knowledge statements about the real world represented as a large collection of subject-predicate-object triplets. Each triplet (s, p, o) denotes a relationship of type p between the subject s and the object o. For example, the triplets (*London*, capital_of, *UK*), (*Let It Be*, performer, *The Beatles*) and (*Barack Obama*, date_of_birth, *04/08/1961*). Most open Knowledge Bases use the Resource Description Framework (RDF) as the specific format for representing these triplets. We can retrieve data from these KB's using SPARQL (Prud'hommeaux and Seaborne, 2008), a standard query language for knowledge graphs represented as RDF triplet stores. It has similarities to SQL but is adapted specifically for querying data modelled as a graph. In our experiments, we use both the raw RDF dumps and the public SPARQL services to access the data. There are many public and commercial large-scale KB's to chose from. We focus on two of the most popular open-source KB's Wikidata and DBpedia as they are inherently linked to Wikipedia which is the unstructured knowledge source targeted by most ODQA datasets.

2.4.1 Wikidata

Wikidata is an open and collaborative knowledge base storing facts about the world as structured data (Vrandecic and Krötzsch, 2014). It can be thought of as the "Wikipedia for structured data" and was launched in October 2012 by the Wikimedia Foundation to provide structured data for Wikimedia projects such as Wikipedia. Wikidata is managed collectively by a large community of contributors (23,000 as of September 2021[1]) who add, change and remove structured knowledge. Wikidata became the leading open-source KB when, In 2014, Google decided to offer the content of its public KB Freebase, one of the largest KB's at the time, to the Wikidata community. Wikidata is organized under a standardized schema managed by the community that defines entity types and Predicates. It consists of more than 4 billion triplets over 90 million entities[1].

2.4.2 DBpedia

DBpedia is another open knowledge base that uses Wikipedia as its source (Auer et al., 2007). The data in DBpedia is collected automatically by extracting structured information from Wikipedia pages. The KB was started in 2007 by researchers from Leipzig University and the Free University of Berlin. They start building the DBpedia Information Extraction Framework (DIEF), a system to automatically extract structured data from the info-boxes on Wikipedia pages. So while Wikidata is used as a data source for Wikipedia, DBpedia in turn uses the info-boxes on Wikipedia as a data source. The public release of DBpedia is called Tiny Diamond and consists of structured knowledge extracted from the English Wikipedia. It comprises of more than 900 million triples as of 2021[2]. Entity properties are published under two schemas, DBpedia Predicates (DBR) and DBpedia Ontology (DBO). DBR properties are extracted using a general raw info-box extraction method. They directly transform properties from Wikipedia page info-boxes into knowledge triplets. This method results in high coverage of facts but is of lower quality as info-boxes do not always use consistent naming. To solve these issues DBpedia also publishes properties under the DBO

[1] https://www.wikidata.org/wiki/Wikidata:Statistics
[2] https://www.dbpedia.org/resources/latest-core

schema. Here they first map info-box properties to a general ontology of 2,795 properties defined by the DBpedia community. For example, they map the info-box properties *birthplace* and *placeofbirth* to the same property *dbo:birthPlace*. This collection of properties is therefore of higher quality but does not have the same coverage as the raw info-box extracted properties. We make use of both DBR and DBO properties in our experiments.

2.5 Entity Linking

As part of our analysis on PAQ, we make extensively use of Entity Linking (EL). EL is the task of correctly identifying entities mentioned in textual inputs and mapping them to entries in a knowledge base such as Wikipedia. This is challenging, as mentions in text do not always overlap exactly with the full entity name, and can sometimes only be inferred from context. For example, in the question *"In which year did Djokovic win Roland Garros?"* we intuitively know we are talking about the tennis player *Novak Djokovic*. EL systems allow machines to do the same and map the textual mention of *Djokovic* to the Wikipdia entry of *Novak Djokovic*. More formally, given a collection of entities $\mathcal{E}$ and text input s, we want to first detect entity mentions in s and then find the entity $e \in \mathcal{E}$ that is the most relevant with respect to these mentions. There are some variants of the pure EL task: Entity Disambiguation and Document Retrieval. In Entity Disambiguation (ED), we are given the boundaries of the mention in s so we only need to find the most relevant entity to this mention. In Document Retrieval (DR) we are also not interested in detecting the exact span of the mention, as we only want to find the most related entity to the text input.

2.5.1 GENRE

Recent EL systems mostly model the EL task as a multi-class classification and follow a bi-encoder approach (similar to the QA-pair retrieval and re-ranking approach of RePAQ) to retrieve the most relevant entities (Wu et al., 2020). The encoder computes embeddings of all Wikipedia entities using the entities' titles and short descriptions. In our work, we use the GENRE entity linker which uses a novel approach that currently achieves state-of-the-art results on standard EL benchmarks. GENRE introduced by Cao et al. (2021) retrieves the relevant entity by generating its name using an autoregressive language model. The relevance score of an entity e, mentioned in sentence s is given by the probability that its name is generated. More formally

$$p(e|s) = \prod_{i=1}^{N} p_\theta(w_i|w < i, s)$$

with w_i as the individual tokens of the entity name with length N, and θ the parameters of the autoregressive language model. Cao et al. (2021) use the pre-trained BART model and fine-tune it on different EL datasets. At inference time, entity names are generated using Constrained Beam Search (CBS) (Sutskever et al., 2014). CBS is similar to regular Beam Search except that it constrains the selection of new tokens so that the output sequence is part of a predefined set of valid sequences. This ensures, for example, that the generated entity names are always valid Wikipedia page titles.

Chapter 3

Analysing Knowledge Gaps in PAQ

To fundamentally improve any system, it is vital to first identify its current shortcomings and bottlenecks and understand their root causes. Only then will we be able to find avenues to implement improvements. Henceforth, this chapter will focus on detailing a thorough error analysis on PAQ/RePAQ and following so, analyse whether there are structural patterns within these errors. More specifically, investigating if there are any knowledge gaps in PAQ and whether these can be improved or fixed through the incorporation of structured data.

3.1 Manual Error Analysis

The first step in our analysis is to conduct a manual error inspection to gain a deeper and more intuitive understanding of the current areas of error or mistake. This will involve manually inspecting the subset of questions which RePAQ answered incorrectly. This is done to better understand if a lack of coverage in PAQ is the main cause of RePAQ's mistakes as suggested by Lewis et al. (2021b) or whether there are other factors present. Incorrect questions in this context refers to the questions for which the top-1 retrieved QA-pair by RePAQ does not match exactly with one of the the gold labels. Hence, these questions lower the aggregate Exact Match (EM) score, the evaluation metric used to assess and compare ODQA models. It is important to note that one of the advantages of the PAQ-RePAQ architecture for ODQA, is that it can return an interpretable list of the top-k answer candidates (e.g. the top-k retrieved QA-pairs). We can, therefore, evaluate the EM performance of RePAQ if we consider all answers from the top-k retrieved QA-pairs. As a point of reference, we will now refer to this metric as EM-k. We observe that there is a relatively large gap (9-10 points) between the EM and EM-5 performance of RePAQ that is not always resolved by using RePAQ's re-ranker. Hence, our first point of investigation is looking into the reasons behind this gap.

3.1.1 Top 1 vs Top 5 Retrieved QA-pairs

To do this, we take the subset of questions that have an EM-1 score of 0 and EM-5 score of 1 on the development set of Natural Questions (NQ). This results in a set of 969 questions explaining a 11.07 percentage point ($969/8757$) difference in overall EM score. Through the process of manually going through a sample of 100 of these questions, there were specific patterns that were observed.

Reason	Frequency (%)
Inaccurate Retriever Ranking	39
Ambiguous Question	22
Answer Granularity	14
Answer Form	12
Incorrect Gold Label	9
Confusing Duplicates in PAQ	4

Table 3.1: Reasons for difference EM top-1 and EM top-5 retrieved QA-pairs

Inaccurate Retriever Ranking As expected based on the analysis of Lewis et al. (2021b), the biggest reason for the difference between the EM-1 and EM-5 scores is the inexactness of RePAQ's fast but approximate dense QA-pair retrieval architecture. Small differences in the phrasing of the question can affect the specific ranking of the top-5 retrieved QA-pairs. This affects about 39% of the questions.

Question	Retrieval Score	Answer
Input how many times peyton been to super bowl		four
how many times has peyton manning *been to* the pro bowl	11.13	14
how many times did peyton manning go to the super bowl	11.77	four
how many times did peyton manning go to the superbowl	12.58	four
how many times did peyton manning play in the super bowl	12.83	two
how many times did peyton manning go to the pro bowl	12.89	14
Input who plays the voice of darth vader in star wars		James Earl Jones
who plays darth vader in star wars	9.35	David Prowse
who plays darth vader in star wars and science fiction	9.65	David Prowse
who does the voice of darth vader in star wars	9.80	James Earl Jones
who plays darth vader in the star wars movies	9.87	David Prowse
who was the voice of darth vader in star wars	9.99	James Earl Jones
Input who started the first news paper in india		James Augustus Hicky
which is the india's first *news paper*	12.79	Udant Martand
who founded the first newspaper in india	14.11	James Augustus Hicky
who started news from indian country newspaper	14.60	Paul DeMain
who was responsible for the first newspaper published in india	14.71	James Augustus Hicky
who was the editor of india's first newspaper	14.84	James Augustus Hicky

Table 3.2: Examples of RePAQ retrieval inaccuracies

In Table 3.2 we are able to see a sample of examples where RePAQ ranks a similar but incorrect QA-pair higher because it matches the general phrasing of the question better. For the question *"how many times peyton been to super bowl"*, RePAQ ranks a question about the "Pro Bowl" higher than one about the "Super Bowl", most likely because the question contains more general wording that overlap with the test question. The "Pro Bowl" question *"how many times has peyton manning been to the pro bowl"* contains the phrase "been to" similar to the test question, compared to the correct "Super Bowl" question *"how many times did peyton manning go to the super bowl"* that contains the less similar phrase "go to". Likewise, for the question *"who plays the voice of darth vader in star wars"*, RePAQ favors a QA-pair with a similar phrasing "who plays" over the correct QA-pair with phrasing "who does the voice". This is indicative that even a small difference in the spelling of one word is able to negatively impact the ranking. This is also seen in the input question *"who started the first news paper in India"*. Here RePAQ prefers an incorrect question that contains the word “news paper” over a more relevant question that contains the “newspaper” spelled as one word. We have observed that applying a more powerful language model that better understands the semantic meaning of the questions seems to be more helpful in ranking the correct answer as the top one. We are able to discover that after enabling RePAQ’s slower but more accurate re-ranker about 80% $(20/25)$ of the questions that we classified with this error type are correct now.

Confusing and Ambiguous Question The second biggest pattern we observe that explains 19% of the cases is ambiguous questions. These are questions with multiple possible interpretations and also multiple possible correct answers which are not included in the dataset’s accepted gold labels. In these cases, PAQ had enough coverage to retrieve the QA-pair that corresponds to the interpretation of the gold label, though the ambiguity of the question hindered RePAQ in ranking it as the top-1 QA-pair.

Question	**Retrieval Score**	**Answer**
Input when was walk on the wild side written		1972
when was walk on the wild side written	11.42	1956
when was a walk on the wild side written	12.87	1956
when was walk on the wild side of life written	13.58	1956
when was walk on the wild side by lou reed written	13.80	1972
when was the song walk on the wild side written	13.88	1962
Input what country’s flag has a moon and star		Turkey
which country has the yellow star and moon flag	16.52	Chile
which country has a flag with a star and crescent	16.67	Algeria
the star and crescent is the national flag of which country	17.39	Turkey
which country has a red flag with a star and a crescent moon	17.42	Algeria
which country has a star and crescent flag	17.75	Turkey
Input what is the visa on arrival fee in thailand		2,000 baht
what is the visa fee in thailand 2017	14.12	1,000 baht
what is the visa fee in thailand	14.20	1,000 baht
how much is the visa fee in thailand	15.87	1,000 baht
how much is visa fee in thailand 2017	15.97	1,000 baht
what is the cost of visa in thailand	16.04	2,000 baht

Table 3.3: Examples of ambiguous questions

We highlight a number of examples of confusing and ambiguous questions in Table 3.3. In the test question *“when was walk on the wild written”*, we are able to observe that RePAQ retrieves five QA-pairs with very similar-looking questions. For context, it should be noted that there exists a song called *“walk on the wild side”* as well as a book and movie with the exact same name. In this instance, it is unclear from the input question that it is referring to the *song*, not the *book*. We observe this pattern repeatedly, where the question could refer to multiple entities.

This does not just occur in questions about entities with the same name, but also when questions lack enough specificity and therefore have multiple possible answers. For example, the question *“what country’s flag has a moon and star”* is not specific enough. This would be considered a vague or not adequately specific question as Turkey is not the only country with a flag that has a moon and star in it. Many Islamic countries such as Tunisia, Algeria and Pakistan have these symbols and would be alternative answers to this question. It also should be noted that PAQ’s collections of QA-pairs are not always factually correct. For example, the QA-pair *“which country has the yellow star and moon flag”* has as answer *Chile* which is not correct and should be either *Malaysia* or *Mauritania*.

This pattern is also observed in questions with answers that are dependent on the time at which the question is asked. For example, the question *"what is the visa on arrival fee in thailand"* is time-dependent as the visa on arrival fee can change. Another question such as *"when did the new macbook air come out"* would also depend on when the question is asked.

We found that enabling RePAQ's re-ranker did not help at all for these ambiguous questions. This is not surprising, as humans would not have been able to provide a single correct answer either. These ambiguous questions are badly formulated questions and should be considered as noise in the data set. This shows that ODQA datasets that are considered high-quality still contain a considerable amount of noise, harming the evaluation and potentially training of ODQA models.

Granularity of the Answer The third most frequent pattern we see is that the predicted top-1 answer has a different granularity than the gold label. At least one of the answers in the remaining top-5 retrieved QA-pairs does share the same granularity as the gold label and is therefore counted as correct under EM-5. For example, in Table 3.4 we see that *Paris* is predicted but the gold label expects the more specific answer *Paris, France*. For the prediction *11 August 1823* it is not accepted as it is more specific than the gold label of *1823*.

Question	Retrieval Score	Answer
Input where are the 2024 summer games going to be		Paris, France
where are the 2024 summer olympic games going to be held	11.14	Paris
where are the 2024 summer olympics going to be held	11.96	Paris, France
Input where was the first motte and bailey castle built		northern Europe
where was the first motte and bailey castle built	10.30	Normandy
where was the motte and bailey castle built	12.14	northern Europe
Input when was the second edition of frankenstein published		1823
when was the second edition of frankenstein published	11.64	11 August 1823
when was the second edition of frankenstein written	13.30	1823

Table 3.4: Examples of correct predictions classified as incorrect because of a different granularity

We see this pattern of granularity mismatch mostly with *where* and *when* questions. After manually going over all *when* where questions in the top-1 vs top-5 mistake subset, we find that 23.50% of *when* questions and 33.33% of *when* questions. In Section 3.2 we propose an alternative automatic evaluation metric that tackles the majority of granularity mistakes with the date and year answers.

Question Type	Total #	Granularity Mistakes #	%
When ___	234	55	23.50
Where ___	72	24	33.33

Table 3.5: Percentage of mistakes due to the difference in granularity per question type

Answer Form Another common pattern we observed is that the top-1 retrieved QA-pair is the correct answer but has a different spelling or is an alternative name referring to the same entity (person, group, concept). As PAQ contains many duplicate QA-pairs with slightly different phrasing, it was able to retrieve alternative QA-pairs where the answer was in the exact same form as the provided gold labels but ranked it lower. In Table 3.6 we can see, for example, the predicted answer *six* which should be accepted as the gold label is *6*, but it is not as it is syntactically different. The same goes for the prediction *Canadian rock band Glass Tiger* which a human evaluator would classify as correct as it refers to the same entity as the gold label *Glass Tiger*. We note that RePAQ's re-ranker is not helpful here either as it can not infer in which exact form the answer is accepted. As mentioned above, we propose an alternative evaluation metric in Section 3.2 which is also able to tackle this particular issue.

Question	**Retrieval Score**	**Answer**
Input how many episodes of peaky blinders season 3		6
how many episodes in season one peaky blinders	14.59	six
how many episodes in a series of peaky blinders	15.54	6
Input who sang don't forget me when i'm gone		Glass Tiger
who sings don't forget me when i'm gone	12.31	Canadian rock band Glass Tiger
who sang dont forget me when im gone	13.71	Glass Tiger
Input what does the zip in zip code stand for Zone Improvement Plan		
what is the name of the zip code in the us	15.60	A ZIP Code
what does the zip code mean in the us	15.95	Zone Improvement Plan

Table 3.6: Examples of correct predictions classified as incorrect because of a different form or spelling

Incorrect Gold Label We observe that in some cases PAQ-RePAQ is smarter than the human annotators from the Natural Questions dataset. In about 10% of the cases, we find that the provided gold label is *not* correct while RePAQ returns the correct answer. These mistakes are very nuanced and can be hard to spot. Some examples of incorrect labels are highlighted in Table 3.10. For the question *when did the social security number become mandatory* the gold label gives *November 1935* which is incorrect as this was the date social security number were introduced in the US. It only became mandatory in *1986*. The same year that, according to the annotators of Natural Questions, *phantom of the opera* started on broadway. This again is not completely correct, as the musical did start in *1986* in London's West End but only moved to New York's Broadway in *1988*.

Question	**Retrieval Score**	**Answer**
Input when did it become mandatory to have a social security number		November 1935
when did the social security number become mandatory	12.90	1986
when was the social security number first issued	15.27	November 1935
Input when did phantom of the opera start on broadway		1986
when did phantom of the opera debut on broadway	10.62	1988
when did the phantom of the opera start	12.99	1986
Input where is the tallest tower in the world located		Tokyo Skytree
where is the tallest tower in the world located	12.28	Tokyo
what is the name of the tallest tower in the world	13.63	Tokyo Skytree

Table 3.7: Examples of incorrect gold labels

Question	Retrieval Score	Answer
Input when is the world population expected to reach 8 billion		2024
when will the worlds population reach 8 billion	10.08	2050
when will the world population reach 8 billion	10.10	2024
Input who played in the super bowl in 2013		San Francisco 49ers, Baltimore Ravens
who played in the 2013 nfl super bowl	7.73	Seattle Seahawks
who played in the super bowl in 2013	7.78	The Baltimore Ravens
Input when is the last episode of the originals going to air		August 1, 2018
when did the last episode of the originals air	13.95	June 23, 2017
when does the season finale of the originals air	15.16	August 1, 2018

Table 3.8: Examples of incorrect duplicates in PAQ

Incorrect Duplicates in PAQ In only 4% of the cases, we observe that RePAQ's retrieved the right question but answered it incorrectly because the answer PAQ generated is factually incorrect. In these cases, a similar QA-pair with a question with almost the exact same phrases did have the correct answer. For example, for the question *"when is the world population expected to reach 8 billion"* PAQ generates both the incorrect answer *2050* and the correct answer *2024*. Likewise, for the question *"who played in the super bowl in 2013"* PAQ contains two QA-pairs with very similar question phrasing but two distinct answers: the incorrect answer *Seattle Seahawks* which played in the *2014* super bowl and one of the correct answers *The Baltimore Ravens*. Also for the question *saywhen is the last episode of the originals going to air* RePAQ returns a QA-pair with the right question but an invalid answer *June 23, 2017*.

3.1.2 Mistakes of Top-5 Retrieved QA-pairs

Now we take a look at the subset of questions with a EM-5 of 0 (i.e. question for which the correct answer was not found in the top-5 retrieved QA-pairs by RePAQ). This regards 4267 questions, 48.73% of the total 8756 questions in the NQ development set. We again take a sample of 100 questions of this subset and manually go through them. From this, there were similar patterns to Section 3.1.1 and these are reported in Table 3.9. In addition, besides manually looking into the error types, we also try to find if the questions can be answered using data available in a KB. This is done by looking up the topic of interest mentioned in the question in Wikidata and seeing if the right answer can be found in one of the triplets.

Mistake Type	Frequency $\%$
PAQ Incomplete Coverage (Unstructured data)	27
PAQ Incomplete Coverage (Structured data)	14
PAQ Incorrect	12
Ambiguous Question	14
Incorrect Gold Label	12
Different Form	11
Different Granularity	9

Table 3.9: Overview of mistake patterns with frequencies

Incomplete Coverage The largest number of mistakes are now due to incomplete coverage in PAQ. In these instances, PAQ does not contain any question in its collection of 65 million QA-pairs that comes close to the test question. We note that in many of these cases PAQ does contain QA-pairs about other facts related to the subject in the question. For example, for the input question *"who played mr. o'hara in gone with the wind"* RePAQ retrieves QA-pairs with questions about who played *miss* o'hara not *mr.* o'hara. Similarly for the question *"what is the name of the 2017 doctor who christmas special"* PAQ contain questions about the name of *doctor who christmas specials* of other years (2005, 2006, 2010, 2012, 2013, and 2016) but not of *2017*. In some cases, it can be observed that PAQ does not contain any QA-pairs about the subject/entity of the question but does contain questions asking about the same concept of other entities. For example, for the question *"when were air brakes first used on trucks"* RePAQ retrieves questions about the first time air brakes were used on *aircraft* and *trains* but not on *trucks*.

Question	**Answer**
Input who played mr. o'hara in gone with the wind	Thomas Mitchell
who played scarlette o'hara in gone with the wind	Vivien Leigh
who played scarlett o'hara in gone with the wind	Vivien Leigh
who played scarlett o'hara in gone with the wind	Vivien Leigh
who was scarlett o'hara's husband in gone with the wind	Charles Hamilton
whoopi goldberg played in gone with the wind	Hattie McDaniel
what is the name of the 2017 doctor who christmas special	Twice Upon a Time
Input what is the name of the 2017 doctor who christmas special	Twice Upon a Time
what is the title of the 2011 christmas special for doctor who	The Doctor, the Widow and the Wardrobe
what is the title of the 2011 christmas special of doctor who	The Doctor, the Widow and the Wardrobe
what is the name of the 2016 christmas special for doctor who	The Return of Doctor Mysterio
when does the doctor who christmas special come out 2017	25 December 2017
what is the name of the 2016 christmas special of doctor who	The Return of Doctor Mysterio
Input when were air brakes first used on trucks	the early 20th century
when was the first air brake used in aircraft	1931
when were air brakes first used in aeroplanes	1931
when were compressed air brakes first used on trains	1869
when were compressed air brakes first used in trains	1869
when was the first air brake put on an airplane	1931

Table 3.10: Examples questions where PAQ lacks coverage

To get an initial estimation of if we can increase the coverage of PAQ in these cases using structure knowledge sources, we manually check whether we can find the answer to these questions in Wikidata. For example, for the question *"what is the running time of the last jedi"* we lookup the Wikidata entry of *Star Wars: Episode VIII – The Last Jedi* and find the correct answer *152 minutes* under the Wikidata statement of *duration*. By doing so, it is shown that for about 34% of all *Incomplete Coverage* mistakes, the correct answer can be found in Wikidata (14% of total EM-5 mistakes). This seems to be a promising observation as it suggests incorporating structured knowledge in PAQ might help increase its coverage and downstream performance.

PAQ Incorrect We also observe that about 12% of the mistake questions are wrong because PAQ contained an incorrect QA-pair. In these instances, RePAQ is able to retrieve a QA-pair with the right question but this QA-pair contains the wrong answer to the question. These issues can therefore be attributed to PAQ's QA-pair generation process. Hence, even though PAQ filters all automatically generated QA-pairs using FiD, incorrect QA-pairs still end up in the final corpus. This confirms the observations done by Lewis et al. (2021b).

Ambiguous Questions Furthermore, we observe again that ambiguous questions in the dataset cause predictions to be classified as incorrect. In our sample of 100 mistake questions, this accounts for 14% of the cases. Ambiguous questions as mentioned above are questions where the question is not specific enough, could refer to multiple entities or contexts, has multiple valid answers depending on the time the question is asked or has no clear-cut answer. Such questions would even be difficult for a human being to answer due to their lack of specificity. For example, in the question *"where is the new years eve concert held"*, there are multiple correct answers depending on which city the question is referring to. There are new years eve concerts in the Musikverein, Vienna; Westminster Central Hall, London and Times Square, New York. Any of these would be considered a correct answer to the question. Similarly, for the question *"who played jill abbott on the young and the restless"* the dataset gives as gold label *"Bond Gideon (1980)"* which is not the only actress who portrayed Jill Abbott on the Young and Restless: the character has been portrayed by six different actresses over the years. We see this more often where the answer to the question depends on the time it is asked. In Table 3.11, we highlight a number of questions from the NQ dataset where the correct answers depend on the time the question was asked or which source document was used. We believe these questions should either not be part of an ODQA dataset or be rephrased to include a reference date.

Question
how many billionaires are there in the united states of america
how many starbucks are there around the world
how many american citizens live in the united states
how many colleges or universities are in the united states
what is the latest version of chrome browser
how many mlb players are in the hall of fame

Table 3.11: Example ambiguous questions, where the answer depends on time asked/source

Incorrect Gold Label Besides ambiguous questions, we observe that in 12% of the cases RePAQ is actually correct but is penalized because the gold labels in the dataset are not correct. Some of these incorrect gold labels are easy to spot when to answer is in a different format than the question is asking for. For example, for question *"how many times france take the world cup"* the provided gold label is *"France"*. Other times the incorrect labels are harder to spot and are sometimes only wrong because of small nuances. For example, for question *"who wins season 2 of americas next top model"* the provided gold label is "*Saleisha Stowers*" who is not the winner of season *2* but season *9*. RePAQ does answer the correct season 9 winner *Yoanna House*.

Question	Incorrect Gold Label	Correct Prediction by RePAQ
how many times france take the world cup	France	15
who wins season 2 of americas next top model	Saleisha Stowers	Yoanna House
once upon a time season 7 episode 11 release date	March 9, 2018	March 2, 2018

Table 3.12: Example of questions with incorrect gold labels

Different Form About 11% of the questions are labelled as mistakes only because the form of the predicted answer is (slightly) different from the gold label answer. A human evaluator would understand in these cases the predicted answer is actually correct, even though it does not match the gold label letter-by-letter. We see this mostly occur with questions asking about people, organisations or amounts.

Gold Label	Prediction
Isaac Newton	Sir Isaac Newton
Soviet	Soviet Union
UNHCR	United Nations High Commissioner for Refugees
Ludwig van Beethoven	Beethoven
European Union (EU)	European Union
American rock band REO Speedwagon	REO Speedwagon
Empire of Japan	Japan
Mayweather	Floyd Mayweather
Peter Reckell	Peter Paul Reckell
Abraham Harold Maslow	Abraham Maslow
David J. Fielding	David Fielding
Keith Hampshire (1973)	Keith Hampshire

Table 3.13: Example of different form answers: names

In Table 3.13, we highlight a number of examples where the gold label and RePAQ's prediction both refer to the same person or group but have a slightly different form. Sometimes, RePAQ answers with the full name of an entity while the correct answer is an abbreviation, other times RePAQ returns an alias of the person or returns a more specific name containing a person's initials.

Gold label	Prediction
18 chapters	18
607 islands & islets	607
8	eight
fifth title	5
2,700	about 2,700
1,800	1,800 acres
over 6 million	over six million
five times	5
24th	24
18 chapters	18
16 episodes	16
2 titles	two
5,100	more than 5,100

Table 3.14: Example of different form answers: numbers

Also with questions where the answer type is an amount, we can see there can exist slight differences between the form of the gold label and a prediction that should be considered correct. As seen in Table 3.14, this can sometimes be caused by numbers being written out instead of given as digits, or where the gold label contains (unnecessary) extra wording while RePAQ's prediction only contains the amount.

Different Granularity The last pattern we observe in 9% of the mistakes is that of different answer granularity. In these cases, it cannot be inferred from the question how granular the expected answer should be. The gold label is either more specific than the predicted answer or the other way around. This happens mostly with questions about locations or moments in time. In Table 3.15 we highlight som examples of this.

Gold label	**Prediction**
November 5, 2019	2019
1970	April 1970
26 August 1968	August 1968
January	January 27, 2018
1967 model year	1967
May 31, 2009	31 May 2009
10,000 years ago	about 10,000 years ago
sometime in 2018	in 2018
the mid-19th century	1861

Table 3.15: Example of different granularity answers: dates

All in all, through this thorough manual error analysis we were able to observe that a substantial amount of mistakes (46% in total) are not just due to a lack of coverage but are more nuanced and can be explained by noise in the data sets and the harsh nature of the automatic EM evaluation metric. Many predictions which are currently classified as incorrect would have been accepted as correct by a human evaluator.

3.2 Normalising Answers During Evaluation

Section 3.1 was able to highlight that many predictions are incorrectly labeled as mistakes due to the harsh nature of the EM metric. In order to do any accurate automatic analysis of PAQ we realize we first need to remove these false negatives. This links us to this section which will focus on finding a better alternative to the standard EM metric. We propose a new metric EM_{norm} that is closer to real human evaluation by automatically normalizing answers which are names, dates, and amounts. We first explain how this normalization process works. Then we evaluate the new metric on RePAQ and other recent ODQA systems.

3.2.1 Name Normalization

In the name normalization step, the aim is to accept all possible names of a particular person and organisation. To do this, we leverage both Wikidata and Wikipedia to find all aliases of persons and organisations. The process is as follows. We begin by using an entity linker over the QA-pair to get the exact entity name of the answer. Following so, this entity is looked up in Wikidata and Wikipedia and all alias names will be added as correct answers. If the prediction is the same as the original gold label or any of these aliases, we label the prediction as correct.

Entity Linking For entity linking we use the pre-trained Entity Disambiguation model from GENRE (Cao et al., 2021). This model is trained to disambiguate entities depending on their context. The possible entities it can return are all the English Wikipedia pages. The input to the GENRE model is the question concatenated with the answer with surrounding start and end tokens. For example, "who plays spiderman in the new spider-man homecoming? [START_ENT] Tom Holland [END_ENT]". We empirically find that adding a question mark at the end of the question (which is not in the dataset), helps the entity linker better disambiguate the entity. We take the top-1 most probable entity generated by GENRE and only accept it if the log probability is higher than a threshold. Following so, we found a threshold of -1.5 to be a good hyper-parameter for our use case, balancing precision and recall of the retrieved entity. And so if no entity with a log probability higher than -1.5 is found, we employ a backup entity linking strategy that involves using the Wikipedia search API to find a Wikipedia page about the answer.

Wikidata Lookup Once the entity is identified we use the SPARQL interface of Wikidata to find all the alias names of the entity. We check that the entity found in the previous step is either a person or organisation by looking a the entities entity type. This sanity check reduces the likelihood that incorrect aliases are added. We then obtain the alias by querying for the English *skos:altLabel*.

Wikipedia Lookup We find in some cases that Wikidata did does contain all possible aliases of a person; it is missing the person's full name containing all middle names. This full name is always found in the first sentence of a person's Wikipedia page. For example, *"Charles John Huffam Dickens (7 February 1812 – 9 June 1870) was an English writer and social critic"*. We leverage this consistent pattern in Wikipedia and automatically extract the full name by using a simple Regex pattern: `(.*?)(?:,| \(| is | was )`.

Original Label	**Alternative Label(s)**
Tom Holland	Tom Holland (actor), Thomas Stanley Holland
Magnus Carlsen	Sven Magnus Øen Carlsen
Kim Il-Sung	Kim Il-sung, Kim Sŏng-ju, Kim Il Sung
Christine Michelle Metz	Chrissy Metz, Christine Metz
Mathew Fraser	Mat Fraser (athlete), Mat Fraser
Charles Dickens	Dickens, C.Dickens, Charles John Huffam Dickens, Boz
Roger Federer	Federer
American girl group Fifth Harmony	5H, Fifth Harmony
American singer Sam Hunt	Sam Hunt, Sam Lowry Hunt

Table 3.16: Examples of automatic normalization of names

3.2.2 Date Normalization

The date normalization step is applied to questions starting with "when". It ensures that predictions with dates that are the same or more specific than the gold label date reference are accepted. Three scenarios are supported depending on the granularity of the gold label. If the gold label is a year (e.g. 2021), only predictions with dates in the same year are accepted. If the gold label is a month and a year (e.g. August, 1998), only predictions with dates sharing that month and year will be accepted. Lastly, if the gold label is a date with a day number, only predictions that match the exact day, month and year are accepted. This date normalisation step reduces the amount of incorrectly labelled mistakes due to a mismatch of granularity as shown in table 3.15. However, it should be noted that it does not solve all of these cases, as predictions that are less granular than the gold label are still not classified as correct.

3.2.3 Amount Normalization

The third normalization step is applied to questions starting with "how many" or "how much". It ensures that predictions with amounts written as either digits or written full-out are both accepted. In addition to this, extra wordings are ignored and only the amount in the answer is considered. This is done by parsing the answer with the Python *number_parser* library that transforms any number written full-out to its corresponding digit representation (e.g. six becomes 6). Following so, we proceed to remove any non-digit character from the answer such as "episodes" in "16 episodes". This is seen in the examples in Table 3.14 presented through this normalization step which are now considered equal in the sense that they will all be accepted in the same way.

3.2.4 Results

Finally, we re-evaluate 6 recent ODQA with this new metric. The change in EM scores is thus highlighted in Table 3.17. As seen from the table, the scores of the systems increase by 0.8-3.9 points. The ranking of the systems stays consistent although the scores of some systems seem to increase more than others. It is evident however that the name normalization step seems to indicate the largest change in points, from 0.5 to 2.9 points increase. This indicates that ODQA systems often answer with different alias names compared to the gold labels.

Model	**EM**	**Name Norm**	**Date Norm**	**Amount Norm**	**Total**	**EM_{norm}**
PAQ retrieval	41.3	+1.7	+0.6	+0.3	+2.6	43.9
PAQ reranker	47.3	+1.6	+0.5	+0.2	+2.5	49.8
FiD	53.1	+2.9	+ 0.8	+0.3	+3.9	57.0
RAG	44.5	+1.0	+0.2	+0.3	+1.5	46.0
T5	36.6	+2.3	+1.0	+0.2	+3.5	40.1
BART	26.5	+0.5	+0.2	+0.1	+0.8	27.2

Table 3.17: The difference in EM and normalized EM scores across ODQA systems on Natural Questions

3.3 Identifying Hard Question Templates

In this section, we aim to investigate if it is possible to find clusters of similar questions that RePAQ answers incorrectly, and if so, what the topics of these clusters are. This information could be useful as it could give insight into the areas where PAQ is lacking knowledge. We begin by explaining how we can find these mistake clusters automatically through a simple algorithm. Following so, we show the clusters of RePAQ's for both Natural Questions and TriviaQA.

3.3.1 Method

As mentioned, we propose a simple but effective way to find mistake clusters. We first use a Named Entity Recognize model[1] to identify the entities in the question and then mask out these entities by replacing them with a mask token. Next, we embed the masked questions using a standard language model encoder. For this, we use the pre-trained DistilBERT model (Alhamzeh et al., 2021), specifically the *elastic/distilbert-base-uncased-finetuned-conll03-english* model from the *sentence_transformers* library, for a good trade-off between speed and accuracy. Finally, we cluster the embeddings using a fast community clustering algorithm and obtain a number of interesting question groups. We consider question pairs with a cosine-similarity larger than 0.9 as similar.

3.3.2 Natural Questions

After applying this clustering procedure on RePAQ's mistake questions of Natural Questions we find a number of clusters with more than 5 questions in them. We highlight some of the largest clusters below; all other clusters can be found in Appendix A. We can see that RePAQ consistently makes mistakes on questions about the same topics. Topics such as actors of characters, songwriters and release dates of episodes. The question clusters look relatively consistent as questions in each cluster follow a similar template. This indicates that PAQ is structurally missing coverage in the topic areas of these clusters.

Actor of character
who played ___ in ___
who played ___ in ___
who played ___ in ___
who played ___ in ___
who played ___ in ___
who played the ___ in ___
who played ___ in ___ on ___
who played ___ in ___ and ___
who played ___ on ___
who played ___ on ___
who played ___ on ___
who played ___ on ___
who played ___ in the original ___
who played the ___ on ___
who played the ___ in the ___
who played ___ in ___'s world ___
who played ___ in the ___ film ___
who played ___ in the ___ movie
who did ___ play in ___
who played ___ in the movie ___
who played ___ in the movie ___
who played ___ in the movie ___
who played ___ on the original ___
who played ___ on the ___ show
who played ___ on the ___ show
who played ___ in the ___ series
who played ___ on the show ___
who plays ___ on ___
who plays ___ on ___
who plays ___ on ___
who plays ___ on ___
who plays ___ in ___
who plays ___ in ___
who plays ___ in ___
who plays ___ in ___
who did ___ play for in the ___
who plays ___ in the ___
who plays ___ in the ___
who plays ___ in the ___
who plays ___ in the ___
who played the ___ in the movies ___
who plays ___ on the ___
who plays ___ on the ___
who is playing ___ in the ___
who played the ___ man in ___
who played ___ in the movie that was made in ___'s
who played ___ on the fresh ___
who played the role of ___ in ___
who played the role of ___ in ___
who played the role of ___ in ___
who plays ___ on our ___
who plays ___ in the ___ movie
who plays ___ in the movie ___
who plays ___ in the movie ___
who did ___ play in ___ 2
who plays ___ in ___ and ___

Release date episode
when does ___ episode ___ come out
when does episode ___ of ___ come out
when does episode ___ of the ___ come out
when will episode ___ of ___ be released
what episode does ___ come on ___'s
when is ___ season 2 episode ___ coming out
what episode does ___ appear in the ___
when is episode ___ of ___ piece coming out
what episode of ___ does ___ come in
when does a new episode of ___ come out
when was ___ episode ___ re released
when does the new episode of ___ come out
when does ___ super episode ___ come out
once upon a time season ___ episode ___
when does season 3 episode ___ of attack on ___ come out
when did season ___ of ___ come out on ___
what episode of ___ is ___ in
when does season ___ of ___ come out
when is ___ season ___ coming out
when is season ___ of ___ coming out

Last name origin
where did the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where did the name ___ come from
where did the name ___ come from
where did the name ___ come from

Number of episodes
how many episodes in the ___ series ___
how many episodes in ___ season ___
how many episodes of ___ and ___ are there
how many episodes in season ___ of ___
how many episodes in season ___ of ___
how many episodes does ___ season ___ have
how many episodes are there in season ___ of ___
how many episode in ___ season ___
how many episodes of ___ and ___ will there be
___ how many episodes are in season ___
how many episodes are there of ___
how many episodes of ___ are there
how many episodes in series ___ of taken
how many episodes in season ___

Establishment date
when was ___ established
when was the ___ established
what year was the ___ originally established
when was the ___ founded
when was ___ first made
when was the new ___ built
when was the first ___ made
when was the ___ in ___ built
when was the ___ enacted
who founded ___ in what is now ___
what year did ___ do the ___

[1]https://spacy.io

3.3.3 TriviaQA

The clusters found on RePAQ’s mistake questions of TriviaQA are about different topics than the topics of the clusters found on Natural Questions, except for the *actor of character* topic. For TriviaQA questions PAQ seems to lack coverage about topics such as cities, countries, presidents, and other topics listed in Appendix B. Also, we note that questions within each cluster are more diverse and do not always follow the exact same template.

Character in movie
who played ’___’ in the ___ film ’___’
who played ___ in the film version of ___
who in the ___ film ___ played ___?
in the ___ film ’___’, who plays the character ’___’
who played the part of ___ in ‘___’
who played ___ in the tv series ’___’
who played the role of ___ in the ___ television series ___’
who played a character based on ___ in a ___ film
___ played ‘___’ in which ___ film
who did ___ portray in the film ___
who played the role of ___ in ___
what character was played first by ___ in a ___ film, and by ___ in a ___ remake
who played the title roles in the ___ film ‘___ and ___’
in the ___, name either actor who played ___ or ___ in the episode the ___
who has played ___ and lord ___ in films by ___
in the film ’___’, who played ___’s husband
what was the name of the character played by ___ in ___
which ___ starred in ___ film ’___’
what was the subject of the ___ film ’___’ which starred ___ and ___
___ played ___ in ___. who was his actor brother.
who starred alongside ___ in ’___’

Cities
which ___ city was known to the ___ as
which ___ city was known by the ___ as ___?
what city has been called the ___ the ___?
in which city did the original ___ in ___
in which city was the ___ founded in ___
___ was the ___ name for which town
which ___ city is said, like ___, to be built on ___
in which town was the ’___’ unveiled in ___
in which city is the ___
in which city is the ___
in which city was the ___ version of ___ assembled
which city was the ___ capital during ___
the ___ is the name given to the ___ from ___ to ___, it continues from there as the ___, to which city
like ___, which ___ city is said to be built on ___
which ___ town was granted city status as part of the ___ celebrations of ___
at which ___, city or ___, did ___ take part
___ are residents of which ___ city
the ___ runs from ___ to which city
the ___ is located in which city

Countries
the ___ and the ___ are in which ___ country
the ___ are which country’s equivalent of the ___
___ is based in which ___ country
___ is in which country
in which country is the ___
___ ruled which country ___
the ___ countries which border the ___ are the ___ republic, the ___, and...
name one of the ___ countries in ___ which are members of the ___
___ shared between ___ countries in the ___. name either
___, ___ and ___ are all sections of the ___ track in which country
which ___ countries make up the ___ countries
___, ___ and which other country were the main opponents of ___ in the ___
for a point each, name the ___ countries, along with the disputed territory of ___, that surround the ___
which country, which is surrounded by ___, gained its independence from ___ in ___
what are the only ___ countries that border both the ___ and the ___
___ is bordered by which ___ countries

Presidents
who was the president of ___, ___
who was ___ president at ___
who preceded ___ as president of the ___
who preceded ___ as president of ___
who was the first president of the ___ in the ___
which ___ president was nicknamed ’the ___’
who replaced ___ as president of ___ in ___
who became president of the ___ after the assassination of president ___
___ was the first ___ president to receive a ___, who was the second
name either president of the ___ whose term coincided with the reign of ___ in the ___
who became the leader of the ___ in ___
by what name was ___ president ___ affectionately known
what was the name of the president of ___ who was executed during the ___ coup
in ___, ___ was re-elected as president of which ___ country
who was king of ___ between ___ and ___

Rivers
___ in ___ stands on which river
on which ___ river does ___ stand
___ stands on which river
___ stands on which river
which river separates ___ from ___
the ___ is another name for which ___ river
___ and ___ are both on which ___ river
upon which river does ___ stand
on which river is ___
the ___ city of ___ is on which river
the ___ city of ___ is on which river

Names
what is ___’s real name
what was ___’s real name
what is ___’s full name
what is ___’s real first name
what is ___’s real first name
what is ___’s first name
what’s another name for ___
___ is better known by what name
what is ’___’s’ first name

Author
who wrote the novel ___, published in ___
who wrote the ___ novel ‘___ ___’
who wrote the ___ opera ’___’
who wrote the musical ___, which was based on ___ novel ___
who wrote ’___: the official biography’ published in ___
who wrote the ___ novel a ___ for a knave?
who wrote the tales of ___ and ___ in ___
who has written ___ novels featuring the ___ drifter ___
which story writer and cartoonist wrote ’the secret ___ of ___’

3.4 Identifying Useful Knowledge Base Topic Areas

In this section, we introduce another method to automatically find knowledge gaps in PAQ. This informs us of potential areas where incorporating structured knowledge would be most helpful. In addition, it can also give us an insight into how ODQA datasets differ in their distribution of question categories.

3.4.1 Method

During the manual error analysis described in Section 3.1, we identified that for a number of mistake questions the correct answer can be found in Wikidata. This was done by manually going through a sample of mistake questions and looking up the Wikidata page of the question's subject entity. We then checked if we could find a knowledge statement (one of the entities' properties) on this page that did provide a correct answer to the question at hand. This was an effective way to get an initial estimate if incorporating information from structured knowledge sources could help increase PAQ's coverage in places where it makes the most mistakes. However, this method does not scale well as it is a time-consuming process to manually check each question. We, therefore, propose an automatic procedure that achieves the same goal; it automatically identifies all topic areas with incomplete coverage in PAQ but complete coverage in Wikidata or DBpedia.

Question	**Entity Found**	**Predicate**	**Cardinality (avg)**	**Answer**
when was the orleans hotel in las vegas built	The Orleans	inception	1	1996
when was national technical institute for the deaf established	National Technical Institute for the Deaf	inception	1	1965
who played larry in the spy next doorname	The Spy Next Door	cast member	10	Lucas Till
who plays the hairdresser in she's the man	She's the Man	cast member	10	Paula Jones
who sang i don't want to live without your love	I Don't Wanna Live Without Your Love	performer	1	Chicago
who sings you are just too good to be true	Can't Take My Eyes Off You	performer	1	Frankie Valli
when did fast and furious 7 come out	Furious 7	publication date	4	April 1, 2015
when did the first dark tower book come out	The Dark Tower (series)	publication date	1	1982

Table 3.18: Example of questions where the answer is automatically found in Wikidata

The procedure works as follows: for each question, we first apply the GENRE entity linker to automatically identify the question's main subject entity (e.g. the person, movie or song the question is asking about). We use the pre-trained Document Retieval model provided by Cao et al. (2021) to retrieve the most relevant entity in Wikipedia for each question. We accept an entity if the log probability given by GENRE is higher than 1.75, to ensure we only consider highly related entities. Then, we query the public SPARQL services of both Wikidata and DBpedia and retrieve a collection of all knowledge triplets (subject, predicate, object) with the identified entity as the subject. Next, for each triplet, we compare the English label of the object with the question-answer provided by the dataset. If they match according to our $\mathrm{EM}_{\mathrm{norm}}$ metric, we record the corresponding predicate. For example, in Table 3.18 we can see that for the question "*when was the* *orleans hotel in las vegas built*", this procedure identifies the entity *The Orleans* and, after iterating through all Wikidata statements about *The Orleans*, finds that the correct answer is found in a triplet with predicate *inception*.

In addition to the predicate's name, we also record the predicate's cardinality - meaning the number of other knowledge statements about the same entity and predicate. For example, for the entity *The Spy Next Door* it was found that there are 10 statements in total with the predicate *cast member*. This makes the predicate less useful if we want to use it to generate unique QA-pairs from it to include into PAQ. In the above example, we would get 10 QA-pairs all with the same question "*who was a cast member of the spy next door*". As most questions in ODQA datasets target a single correct answer, adding these QA-pairs with duplicate questions would not be helpful.

3.4.2 Results on Natural Questions

After executing the above-described procedure on the development set of Natural Questions (NQ) we obtain a list of topics where RePAQ is consistently making mistakes. In Table 3.19 and 3.20 we highlight the top-10 mistake topics found using Wikidata and DBpedia respectively. The full list of results can be found in Appendix C.1. We can observe that RePAQ is making most mistakes on questions about cast members and song performers but also questions about release and publication dates account for a significant amount of mistakes. This agrees with the topics areas identified in Section 3.3. For *cast members* there are not just many mistakes because NQ asks many questions about also because PAQ's is consistently lacking coverage in this area; the EM is only 65.89 for these questions. We can also observe that many of the top-identified predicates in Wikidata and DBpedia are overlapping, though Wikidata seems to have a slightly larger coverage for most areas.

Predicate	**Total #**	**Mistakes #**	**EM**	**Cardinality (avg)**
cast member	299	102	65.89	32.7
performer	306	45	85.29	2.4
has part	65	31	52.31	8.4
publication date	143	21	85.31	2.3
inception	62	18	70.97	1.1
lyrics by	120	18	85.00	1.4
composer	108	16	85.19	1.4
voice actor	62	15	75.81	17.4
start time	68	10	85.29	1.0
country	39	9	76.92	1.6

Table 3.19: Wikidata predicate statistics Natural Questions

3.4.3 Results TriviaQA

In order to assess the generalization of our findings, we are also interested in identifying the mistake areas of RePAQ on TriviaQA. We execute the same procedure as described before but now on the development set of TriviaQA. We highlight the top-10 mistake areas per knowledge base in Table 3.21 and 3.22 and report the full list of results in Appendix D. We observe that *cast member* and *starring* questions still account for most mistakes but less so than for NQ. In general, the number of mistakes seems to be lower for each predicate though we can find a more diverse set of mistake predicates. This indicates that the distribution

Predicate	Total #	Mistakes #	EM	Cardinality (avg)
starring	182	49	73.08	6.5
artist	195	31	84.10	1.5
performer	185	27	85.41	1.2
writer	135	20	85.19	1.6
producer	60	14	76.67	2.1
episodes	34	10	70.59	2.6
release date	70	10	85.71	1.4
title	22	8	63.64	3.0
author	30	7	76.67	1.1
released	63	7	88.89	1.6

Table 3.20: DBpedia predicate statistics Natural Questions

of questions in TriviaQA is more diverse. It also indicates that improving the performance of RePAQ on TriviaQA will not be as straightforward as increasing PAQ's coverage in a single area but requires closing knowledge gaps in multiple areas.

Predicate	Total #	Mistakes #	EM	Cardinality (avg)
country	126	25	80.16	1.9
cast member	91	23	74.73	19.4
has part	65	23	64.62	9.8
notable work	84	23	72.62	17.3
country of citizenship	29	17	41.38	1.3
sport	27	17	37.04	1.1
subclass of	23	16	30.43	1.4
located in the territorial entity	93	15	83.87	1.9
occupation	22	12	45.45	3.6
part of	34	11	67.65	1.2

Table 3.21: Wikidata predicate statistics TriviaQA

Predicate	Total #	Mistakes #	EM	Cardinality (avg)
starring	54	17	68.52	5.7
country	63	12	80.95	1.1
Location	31	11	64.52	1.4
artist	71	9	87.32	1.3
location	28	9	67.86	1.2
performer	37	9	75.68	1.2
author	84	6	92.86	1.3
nationality	9	6	33.33	1.1
notableworks	19	5	73.68	6.0
subdivision name	18	5	72.22	1.6

Table 3.22: DBpedia predicate statistics TriviaQA

Chapter 4

Incorporating Structured Knowledge

Through our investigations and analyses so far, it has been identified in the previous chapter that PAQ does indeed contain certain knowledge gaps. It is now vital that we focus on trying to close these gaps and improve the issues at hand. This chapter will be dedicated to evaluating whether we are able to leverage structured knowledge sources to enrich PAQ in areas where coverage is currently lacking.

The 65 million Question-Answer (QA) pairs in PAQ are currently only generated from unstructured knowledge sources. We hypothesize that a hybrid approach that allows PAQ to incorporate knowledge from both structured and unstructured knowledge sources could help RePAQ's performance for Open-Domain-Question Answering (ODQA). To show this, we will first explain how we incorporated structured knowledge into PAQ and then go on to report and analyse the difference in performance of RePAQ on Natural Questions and TriviaQA after using the extended PAQ collection.

4.1 QA-Pair Generation

In this first section, we will explain how we incorporate structured knowledge in PAQ. It should be noted that structured data which is stored according to a pre-defined schema needs to be transformed in the semi-structured format of PAQ: natural language question-answer pairs. To do this, we resort to a simple templating approach. So, for each data category, we create a general template that allows us to transform structured data stored in KB's and relational databases to QA-pairs. After observing that we can identify a number of coherent areas where PAQ lacks coverage, we believe a general template for each data category to be sufficient. As KB's can be large, containing up to a billion triplets, we only select a handful of "probably asked" categories to extend PAQ with. The structured knowledge sources we use are Wikidata, DBpedia, and IMDb.

4.1.1 Wikidata

To generate QA-pairs from Wikidata, we use the raw Wikipedia dump[1]. Specifically, we use the *latest-truthy.nt.gz* dump from June 1, 2021. This (compressed) file of 47.71GB contains all direct *truthy* statements (no meta-data) available in Wikidata as N-Triples. N-Triples is a plain text serialisation format for knowledge graph data. Each line contains one (subject, predicate, object) triplet. The subject, predicate,

[1] https://dumps.wikimedia.org/wikidatawiki/entities/

objects are referred to by unique identifiers such as "⟨http://www.wikidata.org/entity/Q24⟩". The first step is then to obtain a mapping from all identifiers to the corresponding English label. Following so, with each predicate that we want to include into PAQ, we collect all triplets corresponding to that predicate and transform it into a question-answer pair using a question template we pre-define. For example, for the predicate "author" the question becomes "who wrote ⟨*subject*⟩" with ⟨*subject*⟩ replaced by the subject of the triplet. The answer for this generated QA-pair is the *object* of the triplet.

Predicate	**Triplets #**	**Triplets filtered #**	**Template**
author	20,228,568	1,331	who wrote ⟨*subject*⟩
composer	141,250	3,338	who wrote ⟨*subject*⟩
country	9,324,685	11,935	in which country is ⟨*subject*⟩
country of citizenship	3,891,399	5,241	in which country is ⟨*subject*⟩ born
dissolved date	140,963		when did ⟨*subject*⟩ stop to exist
end time	491,030	1,930	when did ⟨*subject*⟩ end
inception	1,640,087	6,087	when was ⟨*subject*⟩ created
located in territorial entity	7,048,084	4,757	where is ⟨*subject*⟩
location	1,534,627	3,614	where is ⟨*subject*⟩
lyrics by	28,190	881	who wrote the song ⟨*subject*⟩
notable work	67,116	6,836	what is ⟨*subject*⟩ known for
number of episodes	44,908	1,544	how many episodes of ⟨*subject*⟩ are there
number of seasons	29,102	1,300	how many seasons of ⟨*subject*⟩ are there
occupation	8,177,497	21,444	what does ⟨*subject*⟩ do
performer	382,281	4,723	who sings ⟨*subject*⟩
point in time	685,165	1,308	when was ⟨*subject*⟩
publication date	38,029,301	7,050	when was ⟨*subject*⟩ released
sport	1,603,004	3,112	what sport does ⟨*subject*⟩ do
spouse	680,506	3,774	who is the spouse of ⟨*subject*⟩
start time	548,723	2,735	when did ⟨*subject*⟩ start
winner	189,119	2,373	who won ⟨*subject*⟩

Table 4.1: Wikidata QA-pair templates

As Wikidata contains more than 4 billion triplets with more than $10,000$ different predicate types, it is not helpful to parse and include all of Wikidata's triplets into PAQ. Instead, we only focus on triplets related to the knowledge gap areas we found in PAQ. More specifically, we create QA-pairs for a subset of the Wikidata predicates found during our systematic analysis in section **??**. We initially consider all predicates reported in Table 3.19 and 3.21 for which RePAQ makes at least 5 mistakes on Natural Questions or TriviaQA. This ensures that the predicate is related to an area where PAQ's coverage is currently lacking. Next, we filter out all predicates for which the average cardinality is higher than 5. This is done to prevent a high number of irrelevant duplicate questions with multiple answers. For instance, if we would include the predicate *cast member*, we would end up with more than 50 QA-pairs with the same question "who was a cast member in the movie casino royale" for the subject entity *Casino Royale*.

Hence, the final list of predicates we select from Wikidata is shown in Table 4.1. It can be observed that for some predicates, the number of triplets is extremely large. For example, there are more than 38 million statements about publication dates available in PAQ. So, instead of including all these in PAQ, we consider taking a filtered subset of publication dates. This is done by filtering out any publication date statement about entities not mentioned in any of the development and test set questions. To do this, we are able to reuse our entity linking results from section 3.4.

4.1.2 DBpedia

In addition to Wikidata, we also consider adding structured knowledge from DBpedia. It is relevant to consider knowledge from DBpedia as the coverage of Wikidata and DBpedia differ; sometimes facts are available in Wikidata but not in DBpedia and sometimes the other way around. Hence, by considering both KB's we can increase PAQ's coverage of structured data even more.

To include triplets from DBpedia, we query its public SPARQL service. We paginate each request and have a retry-on-try-out mechanism to cope with the rate limitations of the public service. We use the same method of selecting relevant predicates and converting triplets to QA-pairs. The selected DBpedia predicates along the templates are listed in Table 4.2.

Predicate	**Triplets #**	**Template**
dbo:artist	80,438	who sings ⟨*subject*⟩
dbo:author	75,109	who is the author of ⟨*subject*⟩
dbo:country	772,120	in which country is ⟨*subject*⟩
dbo:formationDate	15,767	when was ⟨*subject*⟩ created
dbo:numberOfEpisodes	42,288	how many episodes of ⟨*subject*⟩ are there
dbo:releaseDate	147,267	when did ⟨*subject*⟩ come out
dbo:writer	251,566	who wrote ⟨*subject*⟩
dbp:artist	285,052	who sings ⟨*subject*⟩
dbp:firstAired	50,656	when did ⟨*subject*⟩ first air
dbp:founded	83,769	when was ⟨*subject*⟩ founded
dbp:numEpisodes	43,779	how many episodes of ⟨*subject*⟩ are there
dbp:numSeasons	21,723	how many seasons of ⟨*subject*⟩ are there

Table 4.2: DBpedia predicate QA-pair templates

4.1.3 IMDb

Finally, in addition to data from knowledge graph KB's, we can also consider incorporating data from relational databases. After observing that RePAQ makes many mistakes on questions asking about characters in movies and series, especially in Natural Questions, we looked into data sources with information about these questions.

The *cast member* and *starring* predicates in Wikidata and DBpedia were deemed unhelpful as those are too generic. Most questions ask about the actor of a specific character in a movie so the fact that an actor just starred in that movie is not specific enough. Whereas, we find the IMDb database has the best coverage for this type of question. IMDb, which stands for Internet Movie Database is an online database of all types of information related to movies and tv series, including specific information about who played a certain character in a movie or series. The data is made available either as a dump in TSV format or online through IMDb's public web interface[2].

However, it should be noted that we find the coverage of the TSV dump is not high enough as only information about a limited number of characters per movie is included. The online web interface has a much higher coverage. We, therefore, use the data available from the web interface. We do this using the *IMDbPY*[3] Python library that scrapes IMDb's web interface.

[2]https://www.imdb.com/interfaces/
[3]https://github.com/alberanid/imdbpy

Predicate	Items #	Template
plays character in (train)	168,401	who plays ⟨*character*⟩ in ⟨*movie*⟩
plays character in (dev)	58,861	who plays ⟨*character*⟩ in ⟨*movie*⟩
plays character in (train+dev)	189,761	who plays ⟨*character*⟩ in ⟨*movie*⟩

Table 4.3: IMDb QA pair templates

Instead of including information about characters of all movies that exist, we only include character information about movies which are named in either the train or development set. We extract this list of movies using the following regex pattern:

```
who (?:play|starred)(?:.*?) (?:in|on)(?: the)?(?: (?:movies?|show|
    film|series|original))? (.+)
```

. For each extracted movie we fetch all characters and corresponding actors. Then we generate a QA-pair for each using the question template "who plays ⟨*character*⟩ in ⟨*movie*⟩" and using the *actor* as the answer. After generating all QA-pairs we remove any duplicates.

4.2 Experiments

In this section, we will investigate if enriching PAQ with QA-pairs generated from structured knowledge sources is able to improve downstream performance in effect. To do this, we begin by performing a number of experiments to find out how adding QA-pairs from each predicate in isolation affects RePAQ's performance. Following so, we proceed to then select the best predicates per knowledge source to construct our final extension of PAQ and evaluate RePAQ's performance.

4.2.1 Method

For the initial experiments, we proceed as follows: for each identified predicate in Section 4.1 we collect the generated QA-pairs and concatenate them to the original 65 million QA-pairs in PAQ. We then use RePAQ over this new collection of QA-pairs and make predictions for the questions in the development set of Natural Questions and TriviaQA. Once this is done, we proceed by evaluating the predictions with the EM-5_{norm} metric and compare the difference in points with the baseline (RePAQ's predictions when retrieving over the original PAQ QA-pairs). It should be noted that we do not perform any retraining of RePAQ, but use the same pre-trained *retriever_multi _base_256* model for all experiments. We only create a new HNSW index after embedding all new QA-pairs and concatenating them to the ones from PAQ. We report the difference in EM-5_{norm} for each predicate in Table 4.4, 4.6 and 4.8.

4.2.2 Results with Wikidata

In Table 4.4, we can see that adding QA-pairs for certain predicates of Wikidata can help improve RePAQ's performance. This is evidenced through the fact that some questions that were answered incorrectly before due to a lack of coverage in PAQ, are answered correctly now after adding new QA-pairs. We can see that after including QA-pairs about performers, songwriters, and number of episodes RePAQ's EM-5_{norm} score improves by 0.13, 0.07, and 0.07 respectively on NQ.

Predicate	Number of QA-pairs	Difference EM on NQ	Difference EM on TQA
composer	141,250	0.00	+0.02
country of citizenship	3,891,399	-0.23	-0.18
dissolved date	140,963	+0.01	0.00
end time	491,030	+0.03	-0.04
inception	1,640,087	+0.02	0.00
located in the territorial entity	7,048,084	-0.10	-0.10
location	1,534,627	-0.01	-0.05
lyrics by	28,190	+0.07	-0.01
notable work	67,116	+0.02	-0.04
number of episodes	44,908	+0.07	-0.04
number of seasons	29,102	+0.04	-0.01
occupation	8,177,497	-0.32	-0.63
performer	382,281	+0.13	-0.04
point in time	685,165	+0.01	-0.03
sport	1,603,004	0.00	-0.05
spouse	680,506	-0.01	+0.02
start time	548,723	-0.02	+0.01
winner	189,119	+0.02	-0.01

Table 4.4: Results including QA pairs from Wikidata

However, it can be seen in the table that for most predicates, the improvement in EM-5_{norm} is relatively small and for some even negative. We observe that this occurs mostly when the amount of added QA-pairs is extremely large. For instance, after adding more than 8 million QA-pairs about people's occupation we find that the EM-5_{norm} on TriviaQA drops by 0.64 points. We observe that 65 questions that were answered correctly before are now answered incorrectly. In these cases, RePAQ retrieves QA-pairs about occupation while the question is not asking anything about someone's occupation. We suspect this happens because the QA-pair distribution substantially changes after including 8 million questions about the same topic and RePAQ's retriever is not fine-tuned to this.

When we only include the filtered set of QA-pairs for each predicate we can indeed see in Table 4.5 that the adverse effect of adding a large number of QA-pairs becomes less. For example, adding the full 3.9 million QA-pairs on *country of citizenship* results in a -0.18 change in EM-5_{norm} on TQA, while after filtering out QA-pairs about irrelevant entities it becomes $+0.04$. From these results, it can be concluded that although adding QA-pairs for certain predicates of Wikidata can improve RePAQ's performance, it can also leave it stagnant or even make it worse.

4.2.3 Results with DBpedia

In Table 4.6, we can see that adding QA-pairs for certain predicates of DBpedia also helps improve RePAQ's performance on both Natural Questions and TriviaQA. The predicates that increase the EM-5_{norm} are similar to the Wikidata predicates that increased the score the most. The *dbp:artist*, *dbo:writer* and *dbo:releaseDate* predicates increase the EM-5_{norm} on Natural Questions the most with 0.12, 0.09 and 0.07 points respectively. For *dbp:artist*, this corresponds to a net reduction of (only) 11 mistakes. When we analyse the difference in mistakes between the *dbp:artist* and baseline predictions, we find that RePAQ answers 18 questions about artists correct now which were incorrect before but also 7 questions incorrect which were correct before. We wonder why this happens and find that this occurs because, in these questions which are about the artists of songs, there are multiple artists who performed the same song. Due to

Predicate	Number of QA-pairs	Difference EM on NQ	Difference EM on TQA
author	1,331	+0.00	+0.02
composer	3,338	+0.02	+0.00
country	11,935	+0.03	+0.04
country of citizenship	5,241	+0.00	+0.04
end time	1,930	+0.04	-0.04
inception	6,087	+0.03	-0.01
located in the territorial entity	4,757	+0.03	-0.01
location	3,614	+0.01	-0.04
lyrics by	881	+0.06	-0.04
notable work	6,836	+0.01	-0.01
number of episodes	1,544	+0.02	-0.01
number of seasons	1,300	+0.07	-0.04
occupation	21,444	+0.02	-0.03
performer	4,723	+0.09	-0.01
point in time	1,308	+0.02	-0.01
publication date	7,050	+0.07	-0.04
sport	3,112	+0.02	+0.02
spouse	3,774	+0.00	+0.06
start time	2,735	+0.02	-0.01
winner	2,373	+0.00	+0.01

Table 4.5: Results including filtered QA-pairs from Wikidata

the improved coverage about song artists, RePAQ now answers with one of the other artists that performed a particular song, which is not included in the accepted gold labels (but should be). This shows that adding QA-pairs can make questions that were once answered correctly become answered incorrectly. However, this, does not always mean the new answer is completely wrong as we see the above example it is often more nuanced and due to ambiguous questions or incomplete gold labels in the data set. This patterns does not just occur when adding QA-pairs from DBpedia, this applies to Wikidata and IMDb too.

Predicate	Number of QA-pairs	Difference EM on NQ	Difference EM on TQA
dbo:artist	80,438	0.11	0.02
dbo:author	75,109	0.01	0.03
dbo:country	772,120	0.04	0.01
dbo:formationDate	15,767	0.02	0.01
dbo:numberOfEpisodes	42,288	0.02	-0.01
dbo:releaseDate	147,267	0.07	-0.01
dbo:writer	251,566	0.09	0.01
dbp:artist	285,052	0.12	0.00
dbp:firstAired	50,656	0.00	-0.01
dbp:founded	83,769	-0.01	-0.01
dbp:numEpisodes	43,779	0.04	-0.04
dbp:numSeasons	21,723	0.02	0.01

Table 4.6: Results including QA pairs from DBpedia

We also wonder why the improvements of EM-5_{norm} for each predicate is still relatively low. We would expect more questions to be answered correctly after including QA-pairs about them. In our analysis in Section 3.4 we found, for instance, that there are 31 questions about artists where RePAQ make mistakes but the correct answer could be found in DBpedia. After including all artist QA-pairs generated from DBpedia, we find RePAQ is still making mistakes on 21 of those questions. After manually inspecting these questions we observe that there is a once again a more nuanced reason behind this. The questions asking about the performer of a certain song refer to the song with a slightly different name or by a line in the lyrics of the song as shown in Table 4.9. RePAQ is therefore not able to retrieve the right QA-pair. However, GENRE, the entity linker, is more powerful and is able to retrieve the correct song even though it is not exactly referred to it be the official song title. In conclusion, the experiments done on DBpedia, similar to Wikidata were able to show that.... adding QA-pairs has the potential to improve performance in Re

Question	Song Referred To
who sang you didn't have to love me like you did	Love Me like You Do
who sings should have seen it in color	In Color
who sang are we human or are we dancer	Human
who sings oh sit down next to me	Sit Down
who sings the song it's getting hot in here	Hot in Herre

Table 4.7: Examples of questions referring to song by slightly different name

4.2.4 Results with IMDb

As can be seen in Table 4.8, including cast information from IMDb does not make a difference for RePAQ's performance on TriviaQA but does make a significant difference for RePAQ performance on Natural Questions. This confirms our observation from Section 3.4where we identified that the character-actor questions are the biggest area RePAQ makes mistakes on for Natural Questions due to insufficient coverage in PAQ.

Predicate	Number of QA-pairs	Difference EM on NQ	Difference EM on TQA
plays character in (train)	168,401	0.60	0.01
plays character in (dev)	58,861	0.95	-0.01
plays character in (train+dev)	189,761	1.00	0.00

Table 4.8: Results including QA pairs from IMDb

After including the *plays character in* QA-pairs in PAQ, RePAQ is still, however, not able to answer all questions about this topic correct. We find that out of the 102 mistakes questions about cast members identified in section 3.4, 61 are still answered incorrectly. The reason for this is that these questions refer to the character not by their name but by a general description like *ben's brother* or *the little girl*. IMDb only contains the character names, so the generated QA-pairs will not match the description of the test question. So instead of "*who plays eric's sister in that 70 show*" we can only generate "*who plays laurie forman in that 70 show*".

Question	Character referred to
who is ben's dad in descendants 2	Beast
who plays the little girl in cat in the hat	Sally
who plays the hairdresser in she's the man	Paul
who plays the lion in chronicles of narnia	Aslan
who played eric's sister on that 70 show	Laurie Forman

Table 4.9: Examples of questions referring to character by something else then their name

4.3 Final Results

Now that we have seen the difference in performance for each predicate individually, we can select the most helpful ones and aggregate them all together to extend PAQ. For each knowledge source, Wikidata, DBpedia and IMDb we select the predicates for which the EM-5_{norm} on either NatualQuestions or TriviaQa increased with at least 0.02 points. We evaluate the final performance of RePAQ using the newly extended PAQ on the tests sets of NatualQuestions and TriviaQA.

	NQ		**TQA**	
System	**EM-5**	**EM-5$_{\text{norm}}$**	**EM-5**	**EM-5$_{\text{norm}}$**
PAQ	51.83	53.85	47.91	48.22
PAQ + Wikidata	51.99	54.18	48.25	48.55
PAQ + DBpedia	52.35	54.49	48.01	48.32
PAQ + IMDb	52.33	54.34	47.96	48.27
PAQ + Wikidata + DBpedia + IMDb	**52.85**	**55.10**	**48.32**	**48.63**

Table 4.10: Results including selected structured data

We find a total of 1.02 point EM-5 improvement on NQ and 0.41 points on TQA after including QA-pairs generated from Wikidata, DBpedia and IMDb into PAQ. For NQ most of this improvement comes from IMDb while for TQA most improvement comes from DBpedia and Wikidata. All in all, we can conclude that adding structured data focused on areas where PAQ is currently lacking coverage can help improve RePAQ's performance on ODQA benchmarks but not substantially.

Chapter 5

Conclusion

5.1 Summary

The aim of this thesis was to find methods of improvement for PAQ-RePAQ specifically through the incorporation of data from structured knowledge sources. More specifically, assessing whether the incorporation of this data would be able to improve coverage in PAQ and RePAQ's performance for Open-Domain Question Answering (ODQA). As seen above, we have first carried out a systematic analysis on RePAQ's current performance to better understand its shortcomings and potential areas of improvement. Through a manual error analysis on Natural Questions we have found that over 46% of mistakes are not due to limitations of PAQ and RePAQ but due to ambiguous questions and incorrect gold in the Natural Questions dataset (26%) and the strict nature of the EM evaluation metric (20%). We have proposed a new evaluation metric EM_{norm} that reduces the amount of incorrectly labeled mistakes by normalising dates, names and numbers. This allowed us to better focus on the real improvement area of PAQ and RePAQ: knowledge gaps. We have proposed two methods to automatically identify topic areas where PAQ's coverage is systematically lacking, one involving question clustering and one involving entity linking and knowledge base lookups. Through this, it was identified that PAQ is lacking knowledge in areas such as cast members of movies, performers of songs, publication dates of books, and release dates of episodes. Our experimental findings then established that enriching PAQ with QA-pairs generated from structured knowledge sources about these areas can increase RePAQ's performance for ODQA. Through a simple yet effective approach of templating as described, we were able to demonstrate the ability to integrate structured knowledge into PAQ. This, improved RePAQ's performance on two popular ODQA benchmarks, Natural Questions and TriviaQA, by 1.02 and 0.41 points (EM-5), respectively. These findings ultimately highlighted that although there was evidence of an improvement in performance through the incorporation of structured knowledge, they were still limited and depended on the knowledge gap areas identified.

5.2 Future Work

The main limitation of our work is that in our experiments the RePAQ retriever was not retrained after adding the newly generated QA-pairs into PAQ. We believe that fine tuning the retriever on this new corpus of QA-pairs would lead to an ever higher performance of RePAQ and see this as the first step for further research. A second avenue for future research is looking into not only integrating structure knowledge but also semi-structured knowledge into PAQ. Through our manual error analysis we found that for a handful of questions the correct answer can only be found in a Wikipedia table or list. We therefore believe that adding QA-pairs generated based on these semi-structured knowledge sources into PAQ can help improve RePAQ's performance even more. This believe is strengthened by the results of (Oguz et al., 2020) who observe an substantial increase in performance after considering semi-structured data in their unified ODQA model. Lastly, given the knowledge and insights gained from the systematic analysis of RePAQ and PAQ, we believe a third avenue for future research would be further exploring the adverse effects of noise within ODQA datasets and finding ways to systematically remove this noise. By noise, we are referring to factors such as the ambiguous test questions and incorrect gold labels which hinder researchers in evaluating the true performance of their ODQA models and finding the real bottle-necks.

Bibliography

Alaa Alhamzeh, Mohamed Bouhaouel, Elöd Egyed-Zsigmond, and Jelena Mitrovic. 2021. Distilbert-based argumentation retrieval for answering comparative questions. In *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*, volume 2936 of *CEUR Workshop Proceedings*, pages 2319–2330. CEUR-WS.org.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 615–620. ACL.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879. Association for Computational Linguistics.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1026–1036. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 512–519. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 874–880. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017a. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2017b. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1988–1997. IEEE Computer Society.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single QA system. In *EMNLP (Findings)*, volume EMNLP 2020 of *Findings of ACL*, pages 1896–1907. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4483–4491. ijcai.org.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6086–6096. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Patrick S. H. Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021a. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 1000–1008. Association for Computational Linguistics.

Patrick S. H. Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021b. PAQ: 65 million probably-asked questions and what you can do with them. *CoRR*, abs/2102.07033.

Linqing Liu, Patrick Lewis, Sebastian Riedel, and Pontus Stenetorp. 2021. Challenges in generalization in open domain question answering.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yury A. Malkov and Dmitry A. Yashunin. 2016. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *CoRR*, abs/1603.09320.

Sewon Min, Jordan L. Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, Colin Raffel, Adam Roberts, Tom Kwiatkowski, Patrick S. H. Lewis, Yuxiang Wu, Heinrich Küttler, Linqing Liu, Pasquale Minervini, Pontus Stenetorp, Sebastian Riedel, Sohee Yang, Minjoon Seo, Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Edouard Grave, Ikuya Yamada, Sonse Shimaoka, Masatoshi Suzuki, Shumpei Miyawaki, Shun Sato, Ryo Takahashi, Jun Suzuki, Martin Fajcik, Martin Docekal, Karel Ondrej, Pavel

Smrz, Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, Jianfeng Gao, Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Sejr Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Wen-tau Yih. 2020. Neurips 2020 efficientqa competition: Systems, analyses and lessons learned. In *NeurIPS 2020 Competition and Demonstration Track, 6-12 December 2020, Virtual Event / Vancouver, BC, Canada*, volume 133 of *Proceedings of Machine Learning Research*, pages 86–111. PMLR.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Sejr Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2020. Unified open-domain question answering with structured and unstructured knowledge. *CoRR*, abs/2012.14610.

Ethan Perez, Patrick S. H. Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8864–8880. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2463–2473. Association for Computational Linguistics.

John M. Prager. 2006. Open-domain question-answering. *Found. Trends Inf. Retr.*, 1(2):91–231.

Eric Prud'hommeaux and Andy Seaborne. 2008. SPARQL Query Language for RDF.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5418–5426. Association for Computational Linguistics.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4430–4441, Florence, Italy. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 641–651. Association for Computational Linguistics.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6397–6407. Association for Computational Linguistics.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick S. H. Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *CoRR*, abs/2101.00774.

Appendix A

Mistake Patterns NQ

Actor of character
who played ___ in ___
who played ___ in ___
who played ___ in ___
who played ___ in ___
who played ___ in ___
who played the ___ in ___
who played ___ in ___ on ___
who played ___ in ___ and ___
who played ___ on ___
who played ___ on ___
who played ___ on ___
who played ___ on ___
who played ___ in the original ___
who played the ___ on ___
who played the ___ in the ___
who played ___ in ___'s world ___
who played ___ in the ___ film ___
who played ___ in the ___ movie
who did ___ play in ___
who played ___ in the movie ___
who played ___ in the movie ___
who played ___ in the movie ___
who played ___ on the original ___
who played ___ on the ___ show
who played ___ on the ___ show
who played ___ in the ___ series
who played ___ on the show ___
who plays ___ on ___
who plays ___ on ___
who plays ___ on ___
who plays ___ on ___
who plays ___ in ___
who plays ___ in ___
who plays ___ in ___
who plays ___ in ___
who did ___ play for in the ___
who plays ___ in the ___
who plays ___ in the ___
who plays ___ in the ___
who plays ___ in the ___
who played the ___ in the movies ___
who plays ___ on the ___
who plays ___ on the ___
who is playing ___ in the ___
who played the ___ man in ___
who played ___ in the movie that was made in ___'s
who played ___ on the fresh ___
who played the role of ___ in ___
who played the role of ___ in ___
who played the role of ___ in ___
who plays ___ on our ___
who plays ___ in the ___ movie
who plays ___ in the movie ___
who plays ___ in the movie ___
who did ___ play in ___ 2
who plays ___ in ___ and ___

Release date episode
when does ___ episode ___ come out
when does episode ___ of ___ come out
when does episode ___ of the ___ come out
when will episode ___ of ___ be released
what episode does ___ come on ___'s
when is ___ season 2 episode ___ coming out
what episode does ___ appear in the ___
when is episode ___ of ___ piece coming out
what episode of ___ does ___ come in
when does a new episode of ___ come out
when was ___ episode ___ re released
when does the new episode of ___ come out
when does ___ super episode ___ come out
once upon a time season ___ episode ___
when does season 3 episode ___ of attack on ___ come out
when did season ___ of ___ come out on ___
what episode of ___ is ___ in
when does season ___ of ___ come out
when is ___ season ___ coming out
when is season ___ of ___ coming out

Last name origin
where did the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where does the last name ___ come from
where did the name ___ come from
where did the name ___ come from
where did the name ___ come from

Meaning of
what does the name ___ mean in ___
what does the name ___ mean in ___
what is the meaning of name ___ in ___
what is the name of ___ in ___
by what name is ___ known in ___ and ___
what is ___'s real name in ___
what is the meaning of the name ___
what is the meaning of the name ___
real name of ___ in ___
what is the meaning of ___ in ___
what is the meaning of ___ in ___
what does ___ mean in ___ from ___
what does it mean if the ___ is ___
during ___ the term ___ was initially the name for ___ ___ real name
what do the letters ___ in ___ stand for
what is ___ in ___
what does ___ stand for ___
what do the ___ and ___ stand for in ___
nickname of ___ no ___
the name of the ___
where does the name ___ come from
where does the name ___ come from
what is a ___ in the ___
real name of ___ in ek ___ar ___
where does ___ come from in ___
where is ___ in ___
what is the real name for ___
what part of ___ is ___ in

Number of episodes
how many episodes in the ___ series ___
how many episodes in ___ season ___
how many episodes of ___ and ___ are there
how many episodes in season ___ of ___
how many episodes in season ___ of ___
how many episodes does ___ season ___ have
how many episodes are there in season ___ of ___
how many episode in ___ season ___
how many episodes of ___ and ___ will there be
___ how many episodes are in season ___
how many episodes are there of ___
how many episodes of ___ are there
how many episodes in series ___ of taken
how many episodes in season ___

Establishment date
when was ___ established
when was the ___ established
what year was the ___ originally established
when was the ___ founded
when was ___ first made
when was the new ___ built
when was the first ___ made
when was the ___ in ___ built
when was the ___ enacted
who founded ___ in what is now ___
what year did ___ do the ___

Song writer
who wrote the song ___ by ___
who wrote the song ___ sung by ___
who wrote the song ___

who wrote the song call ___ by ___
what songs did ___ wrote for ___
who sang ___ for ___
who wrote ___
who wrote ___
who is the song ___ written about
who wrote the ___
when did the song ___ by the ___ come out

President
who was the president of ___ in ___
who was the president of ___ during ___1
who is the president of ___ in ___
who is the president of ___
who is the president of ___
who was the president during ___
who was leader of ___ during ___
who was the governor of ___ during ___
who was the governor general of ___ during ___

Film location
where was the movie ___ filmed
where was the movie ___ filmed
where was the movie ___ filmed
where was the movie ___ filmed
where was the movie once upon a time in ___ filmed
where was the television show ___ filmed
where did they film the ___ show
___ and ___ go to ___ where was it filmed
where was ___ ___ filmed in ___

Release date movie
when did the ___ movie come out
when did the movie ___ come out
when did the first ___ movie come out
when did the first ___ movie come out
when did the first ___ movie come out
when did the last ___ movie come out
when was the first ___ movie released
when did the ___ come out
when did the ___ come out

Who plays who
who plays ___'s mother on ___
who plays ___'s mother in ___
who played ___'s mother on ___ and ___
who played ___'s mother in the movie ___
who played the mother on fresh ___ of ___
who plays ___'s wife on ___
who played ___ daughter in ___
who played ___'s sister on ___
who plays ___'s mom on the tv series

Winner competition in year
who won the ___ from ___
who won the ___ in ___
who won the ___ in ___
who won the ___ in ___
who won the ___ in ___
who won ___ ___
who won the ___ in ___ in the ___
who won ___ in ___
who won in the ___ w___
who won ___ v___
who won ___

Population
what is the population of ___
what is the population of ___
what is the population of ___
what was the population of ___ in ___
what is the population size of ___
population of the ___ at its peak

Winner recurring show
who wins season ___ of the ___
who won season ___ of the ___
who won season ___ of ___
who wins season ___ of total drama ___
who does ___ in season ___ of the ___
who wins season ___ of ___ next top model

College
where did ___ and ___ go to college
what college did ___ and ___ go to
where did ___ and ___ go to school
where does ___ go to college in ___3

Appendix B

Mistake Patterns TriviaQA

Character in movie
who played '___' in the ___ film '___'
who played ___ in the film version of ___
who in the ___ film ___ played ___?
in the ___ film '___', who plays the character '___'
who played the part of ___ in '___'
who played ___ in the tv series '___'
who played the role of ___ in the ___ television series ___'
who played a character based on ___ in a ___ film
___ played '___' in which ___ film
who did ___ portray in the film ___
who played the role of ___ in ___
what character was played first by ___ in a ___ film, and by ___ in a ___ remake
who played the title roles in the ___ film '___ and ___'
in the ___, name either actor who played ___ or ___ in the episode the ___
who has played ___ and lord ___ in films by ___
in the film '___', who played ___'s husband
what was the name of the character played by ___ in ___
which ___ starred in ___ film '___'
what was the subject of the ___ film '___' which starred ___ and ___
___ played ___ in ___. who was his actor brother.
who starred alongside ___ in '___'

Cities
which ___ city was known to the ___ as
which ___ city was known by the ___ as ___?
what city has been called the ___ the ___?
in which city did the original ___ in ___
in which city was the ___ founded in ___
___ was the ___ name for which town
which ___ city is said, like ___, to be built on ___
in which town was the '___' unveiled in ___
in which city is the ___
in which city is the ___
in which city was the ___ version of ___ assembled
which city was the ___ capital during ___
the ___ is the name given to the ___ from ___ to ___, it continues from there as the ___, to which city
like ___, which ___ city is said to be built on ___
which ___ town was granted city status as part of the ___ celebrations of ___
at which ___, city or ___, did ___ take part
___ are residents of which ___ city
the ___ runs from ___ to which city
the ___ is located in which city

Countries
the ___ and the ___ are in which ___ country
the ___ are which country's equivalent of the ___
___ is based in which ___ country
___ is in which country
in which country is the ___
___ ruled which country ___
the ___ countries which border the ___ are the ___ republic, the ___, and...
name one of the ___ countries in ___ which are members of the ___
___ shared between ___ countries in the ___. name either
___, ___ and ___ are all sections of the ___ track in which country
which ___ countries make up the ___ countries
___, ___ and which other country were the main opponents of ___ in the ___
for a point each, name the ___ countries, along with the disputed territory of ___, that surround the ___
which country, which is surrounded by ___, gained its independence from ___ in ___
what are the only ___ countries that border both the ___ and the ___
___ is bordered by which ___ countries

Presidents
who was the president of ___, ___
who was ___ president at ___
who preceded ___ as president of the ___
who preceded ___ as president of ___
who was the first president of the ___ in the ___
which ___ president was nicknamed 'the ___'
who replaced ___ as president of ___ in ___
who became president of the ___ after the assassination of president ___
___ was the first ___ president to receive a ___, who was the second
name either president of the ___ whose term coincided with the reign of ___ in the ___
who became the leader of the ___ in ___
by what name was ___ president ___ affectionately known
what was the name of the president of ___ who was executed during the ___ coup
in ___, ___ was re-elected as president of which ___ country
who was king of ___ between ___ and ___

Rivers
___ in ___ stands on which river
on which ___ river does ___ stand
___ stands on which river
___ stands on which river
which river separates ___ from ___
the ___ is another name for which ___ river
___ and ___ are both on which ___ river
upon which river does ___ stand
on which river is ___
the ___ city of ___ is on which river
the ___ city of ___ is on which river

Names
what is ___'s real name
what was ___'s real name
what is ___'s full name
what is ___'s real first name
what is ___'s real first name
what is ___'s first name
what's another name for ___
___ is better known by what name
what is '___'s' first name

Author
who wrote the novel ___, published in ___
who wrote the ___ novel '___ ___'
who wrote the ___ opera '___'
who wrote the musical ___, which was based on ___ novel ___
who wrote '___: the official biography' published in ___
who wrote the ___ novel a ___ for a knave?
who wrote the tales of ___ and ___ in ___
who has written ___ novels featuring the ___ drifter ___
which story writer and cartoonist wrote 'the secret ___ of ___'

Animal types
the ___ is what type of animal
a ___ is what type of animal
what type of animal is a ___
a ___ is what type of creature
an ___ is what type of creature
'___'s' and '___'s' are ___ of the ___ speciesof which animal
what type of creature is a ___
what type of creature is an ___
what creatures are colloquially known as '___' in parts of ___

Prime ministers
who became prime minister of ___ in ___
___ became prime minister of which ___ country in ___
___ became prime minister of which ___ country in ___
who was the prime minister of ___
___ was prime minister of which country from ___ and ___
___ was the first ___ prime minister of the ___. who was the second
which ___ is named after a prime minister, who was mp for ___ from ___
___ was the first ___ name of which ___ prime minister, who was born during ___, in ___

Appendix C

KB Predicates Statistics Natural Questions

C.1 Wikidata

Predicate	Total #	Mistakes #	EM	Cardinality (avg)
cast member	299	102	65.89	32.7
performer	306	45	85.29	2.4
has part	65	31	52.31	8.4
publication date	143	21	85.31	2.3
inception	62	18	70.97	1.1
lyrics by	120	18	85.00	1.4
composer	108	16	85.19	1.4
voice actor	62	15	75.81	17.4
start time	68	10	85.29	1.0
country	39	9	76.92	1.6
named after	19	9	52.63	1.7
has part→series ordinal	29	7	75.86	13.8
point in time	30	7	76.67	1.0
part of	25	6	76.00	1.2
end time	27	5	81.48	1.0
located in the administrative territorial entity	18	5	72.22	1.5
main subject	6	5	16.67	1.5
member of sports team	10	5	50.00	4.4
number of seasons	8	5	37.50	1.0
participant	23	5	78.26	11.4
subclass of	7	5	28.57	1.6
winner	55	5	90.91	4.2
Commons category	7	4	42.86	1.0
author	34	4	88.24	1.1
award received→winner	22	4	81.82	7.0
cast member→character role	10	4	60.00	17.0
characters	21	4	80.95	17.8
different from	8	4	50.00	1.6
dissolved, abolished or demolished date	10	4	60.00	1.0

filming location	20	4	80.00	2.4
location	17	4	76.47	2.5
number of episodes	24	4	83.33	1.0
original broadcaster→start time	17	4	76.47	1.2
participating team	29	4	86.21	6.1
producer	36	4	88.89	1.4
spouse	10	4	60.00	1.2
award received→point in time	18	3	83.33	13.9
creator	19	3	84.21	1.2
mouth of the watercourse	5	3	40.00	1.0
population	8	3	62.50	7.2
present in work→performer	14	3	78.57	3.1
ranking→point in time	7	3	57.14	279.1
screenwriter	12	3	75.00	2.7
shares border with	5	3	40.00	9.4
signatory→point in time	5	3	40.00	8.2
significant event→point in time	21	3	85.71	2.7
Twitter username→start time	6	2	66.67	1.5
chairperson	8	2	75.00	3.5
date of official opening	11	2	81.82	1.0
founded by	10	2	80.00	1.5
nominated for→nominee	21	2	90.48	9.0
officeholder	17	2	88.24	2.8
work period (start)	5	2	60.00	1.0
YouTube video ID→winner	8	1	87.50	6.2
candidate	8	1	87.50	4.9
capital	7	1	85.71	1.7
country of origin	7	1	85.71	1.1
date of death	7	1	85.71	1.0
depicts	5	1	80.00	4.2
director	6	1	83.33	1.0
instance of→start time	7	1	85.71	1.6
narrative location	22	1	95.45	2.1
presenter	8	1	87.50	3.8
voice actor→character role	5	1	80.00	28.4
discoverer or inventor	7	0	100.00	1.1
exploitation visa number→start time	10	0	100.00	1.0
located in or next to body of water	5	0	100.00	2.2
member of sports team→start time	5	0	100.00	3.6
nominated for→point in time	6	0	100.00	6.5

C.2 DBpedia

Predicate	Total #	Mistakes #	EM	Cardinality (avg)
starring	182	49	73.08	6.5
artist	195	31	84.10	1.5
performer	185	27	85.41	1.2
writer	135	20	85.19	1.6
producer	60	14	76.67	2.1
episodes	34	10	70.59	2.6
release date	70	10	85.71	1.4

title	22	8	63.64	3.0
author	30	7	76.67	1.1
released	63	7	88.89	1.6
start	36	7	80.56	4.7
country	29	6	79.31	1.3
date	33	6	81.82	4.5
first aired	64	6	90.62	1.0
link	7	6	14.29	5.6
Location	16	5	68.75	1.2
formation date	7	5	28.57	1.0
last aired	15	5	66.67	1.0
num episodes	34	5	85.29	1.1
number of seasons	7	5	28.57	1.0
Home	9	4	55.56	5.2
after election	17	4	76.47	1.0
chronology	17	4	76.47	1.1
label	10	4	60.00	3.1
nominee	11	4	63.64	1.1
num seasons	6	4	33.33	1.0
number of episodes	33	4	87.88	1.0
years	18	4	77.78	3.5
Road	12	3	75.00	3.2
candidate	12	3	75.00	4.5
composer	19	3	84.21	1.3
director	13	3	76.92	1.2
established	11	3	72.73	1.0
film director	10	3	70.00	1.0
first air date	35	3	91.43	1.0
formed	5	3	40.00	1.4
origin	5	3	40.00	1.0
place	6	3	50.00	1.0
portrayer	31	3	90.32	2.7
signeddate	5	3	40.00	1.0
Link from a Wikipage to another Wikipage	5	2	60.00	7.0
Team	10	2	80.00	1.9
active years start year	6	2	66.67	1.0
alt	5	2	60.00	3.0
end	6	2	66.67	4.3
founded by	6	2	66.67	1.0
leader	7	2	71.43	1.1
location	10	2	80.00	1.1
music	12	2	83.33	2.2
opened	7	2	71.43	1.9
owner	5	2	60.00	2.0
presenter	10	2	80.00	3.0
screenplay	8	2	75.00	3.0
spouse	5	2	60.00	3.0
voice	15	2	86.67	3.3
airdate	6	1	83.33	1.2
album	5	1	80.00	1.0
birth name	15	1	93.33	1.1
caption	8	1	87.50	1.2
champion	8	1	87.50	1.0
date ratified	5	1	80.00	1.0
death date	7	1	85.71	1.0
executive producer	11	1	90.91	6.7
first	8	1	87.50	1.8

founding date	7	1	85.71	1.1
music composer	7	1	85.71	1.1
next year	6	1	83.33	1.5
opening date	5	1	80.00	1.8
prev year	6	1	83.33	1.5
recorded	7	1	85.71	1.6
releasedate	6	1	83.33	5.7
season number	5	1	80.00	1.0
team	6	1	83.33	1.3
year start	5	1	80.00	1.0
AdmittanceDate	5	0	100.00	1.0
Division Champs	5	0	100.00	5.0
HomeAbr	5	0	100.00	2.0
RoadAbr	5	0	100.00	2.0
before election	6	0	100.00	1.0
beginning date	5	0	100.00	1.0
city	5	0	100.00	1.6
completion date	8	0	100.00	1.0
creator	11	0	100.00	1.7
creator (agent)	9	0	100.00	1.0
developer	6	0	100.00	1.7
established date	9	0	100.00	3.0
incumbent	11	0	100.00	1.0
lyricist	6	0	100.00	1.0
narrated	7	0	100.00	1.6
narrator	7	0	100.00	1.0
no league champs	5	0	100.00	1.0
opening theme	6	0	100.00	1.0
population total	5	0	100.00	1.0
published	5	0	100.00	1.0
regular season	5	0	100.00	1.0
theme music composer	7	0	100.00	1.9

Appendix D

KB Predicates Statistics TriviaQA

D.1 Wikidata

Predicate	Total #	Mistakes #	EM	Cardinality (avg)
country	126	25	80.16	1.9
cast member	91	23	74.73	19.4
has part	65	23	64.62	9.8
notable work	84	23	72.62	17.3
country of citizenship	29	17	41.38	1.3
sport	27	17	37.04	1.1
subclass of	23	16	30.43	1.4
located in the administrative territorial entity	93	15	83.87	1.9
occupation	22	12	45.45	3.6
given name	16	11	31.25	1.8
part of	34	11	67.65	1.2
performer	79	11	86.08	1.7
spouse	34	10	70.59	1.9
different from	22	9	59.09	1.5
located in or next to body of water	23	8	65.22	3.7
location	40	8	80.00	1.5
present in work	30	8	73.33	3.2
cast member→character role	17	7	58.82	10.2
contains administrative territorial entity	24	7	70.83	25.7
diplomatic relation	18	7	61.11	46.6
family name	9	7	22.22	1.6
instance of	14	7	50.00	2.4
named after	20	7	65.00	1.3
shares border with	18	7	61.11	5.8
author	85	6	92.94	1.0
winner	19	6	68.42	18.6
Freebase ID	8	5	37.50	1.0
award received→for work	22	5	77.27	4.8
capital of	26	5	80.77	5.5
characters	22	5	77.27	10.8
creator	38	5	86.84	1.0
headquarters location	17	5	70.59	1.4
member of sports team	8	5	37.50	6.2

Predicate	Total #	Mistakes #	EM	Cardinality (avg)
narrative location	14	5	64.29	1.7
nominated for→for work	24	5	79.17	9.3
topic’s main category	7	5	28.57	1.0
Commons category	7	4	42.86	1.0
Encyclopædia Britannica Online ID	7	4	42.86	1.0
JSTOR topic ID	5	4	20.00	1.2
Quora topic ID	6	4	33.33	1.0
child	14	4	71.43	9.4
country of origin	16	4	75.00	1.0
made from material	11	4	63.64	2.5
member of	17	4	76.47	3.0
position held→replaced by	16	4	75.00	5.2
sibling	8	4	50.00	2.1
Commons category	14	3	78.57	1.0
French Vikidia ID	5	3	40.00	1.0
Great Russian Encyclopedia Online ID	5	3	40.00	1.0
KBpedia ID	5	3	40.00	1.0
WordNet 3.1 Synset ID	5	3	40.00	1.0
applies to jurisdiction	7	3	57.14	1.1
instance of→of	13	3	76.92	1.2
lyrics by	18	3	83.33	1.1
mother	9	3	66.67	1.1
owner of	9	3	66.67	15.3
position held→replaces	12	3	75.00	5.9
said to be the same as	8	3	62.50	2.8
subclass of→of	10	3	70.00	1.2
award received→winner	17	2	88.24	5.8
basin country	7	2	71.43	5.4
color	5	2	60.00	2.2
composer	31	2	93.55	1.1
day in year for periodic occurrence	8	2	75.00	1.1
depicts	5	2	60.00	3.2
derivative work	9	2	77.78	2.3
end time	9	2	77.78	1.0
followed by	5	2	60.00	1.0
follows	6	2	66.67	1.2
founded by	7	2	71.43	1.6
head of state	8	2	75.00	6.6
instrument	10	2	80.00	2.1
owned by	14	2	85.71	1.1
place of birth	12	2	83.33	1.1
place of death	8	2	75.00	1.1
studies	5	2	60.00	1.2
taxon common name	8	2	75.00	34.8
work location	6	2	66.67	2.8
after a work by	6	1	83.33	1.0
based on	8	1	87.50	1.0
candidate	9	1	88.89	5.9
capital	42	1	97.62	1.3
conversion to standard unit	6	1	83.33	13.2
date of death	7	1	85.71	1.0
director	19	1	94.74	1.1
discoverer or inventor	8	1	87.50	1.1
father	6	1	83.33	1.0

Predicate	Total #	Mistakes #	EM	Cardinality (avg)
located in/on physical feature	11	1	90.91	1.3
media franchise	7	1	85.71	1.0
model item	12	1	91.67	9.7
nominated for→nominee	23	1	95.65	9.0
point in time	7	1	85.71	1.0
position held→end time	5	1	80.00	7.0
presenter	10	1	90.00	4.2
producer	14	1	92.86	1.6
screenwriter	13	1	92.31	1.6
start time	11	1	90.91	1.0
title	7	1	85.71	1.0
historic county	19	0	100.00	1.0
inception	9	0	100.00	1.0
movement	5	0	100.00	2.8
occupant	7	0	100.00	1.3
place served by transport hub	7	0	100.00	1.9
successful candidate	5	0	100.00	1.0

D.2 DBpedia

Predicate	Total #	Mistakes #	EM	Cardinality (avg)
starring	54	17	68.52	5.7
country	63	12	80.95	1.1
Location	31	11	64.52	1.4
artist	71	9	87.32	1.3
location	28	9	67.86	1.2
performer	37	9	75.68	1.2
Link from a Wikipage to another Wikipage	17	8	52.94	5.5
author	84	6	92.86	1.3
nationality	9	6	33.33	1.1
notableworks	19	5	73.68	6.0
subdivision name	18	5	72.22	1.6
known for	10	4	60.00	3.0
nickname	5	4	20.00	1.6
after	8	3	62.50	1.1
associated acts	8	3	62.50	3.6
common name	10	3	70.00	1.0
conventional long name	10	3	70.00	1.2
label	6	3	50.00	5.0
pushpin map	11	3	72.73	1.0
series	18	3	83.33	1.5
title	21	3	85.71	1.7
type	5	3	40.00	1.0
writer	31	3	90.32	2.0
President	5	2	60.00	1.0
Relates an entity to the populated place in which it is located.	7	2	71.43	1.3
associated band	7	2	71.43	2.9
associated musical artist	7	2	71.43	2.9
capital	18	2	88.89	1.7
creator	19	2	89.47	1.0

Predicate	Total #	Mistakes #	EM	Cardinality (avg)
creator (agent)	18	2	88.89	1.0
date	8	2	75.00	1.1
designer	5	2	60.00	1.2
first	11	2	81.82	1.0
headquarter	5	2	60.00	1.2
ingredient	5	2	60.00	2.0
notable works	5	2	60.00	1.6
operator	5	2	60.00	1.0
predecessor	7	2	71.43	1.4
producer	16	2	87.50	1.8
spouse	10	2	80.00	1.5
subdivision	10	2	80.00	2.4
vicepresident	8	2	75.00	2.8
chronology	6	1	83.33	1.2
city	8	1	87.50	2.6
composer	13	1	92.31	1.0
death date	5	1	80.00	1.0
director	17	1	94.12	1.0
film director	17	1	94.12	1.0
first appearance	5	1	80.00	1.0
leader	6	1	83.33	2.0
map type	8	1	87.50	1.0
mouth mountain	5	1	80.00	1.2
mouth place	5	1	80.00	1.2
music	6	1	83.33	1.7
notable work	5	1	80.00	2.4
portal	5	1	80.00	2.8
presenter	6	1	83.33	4.0
tenant	8	1	87.50	1.6
tenants	7	1	85.71	1.7
admin hq	7	0	100.00	1.0
birth place	5	0	100.00	1.8
largest city	7	0	100.00	1.4
owner	6	0	100.00	1.3
place	5	0	100.00	1.0
post town	6	0	100.00	1.0
seat	9	0	100.00	1.0
successor	5	0	100.00	1.8