Gottfried Wilhelm Leibniz Universität Hannover

Fakultät für Elektrotechnik und Informatik

# Evaluating SQuAD-based Question Answering for the Open Research Knowledge Graph Completion

*A thesis submitted in fulfillment of the requirements for the degree of*

**Bachelor of Science in Computer Science**

BY

**Moussab Hrou**

Matriculation number: 10014616

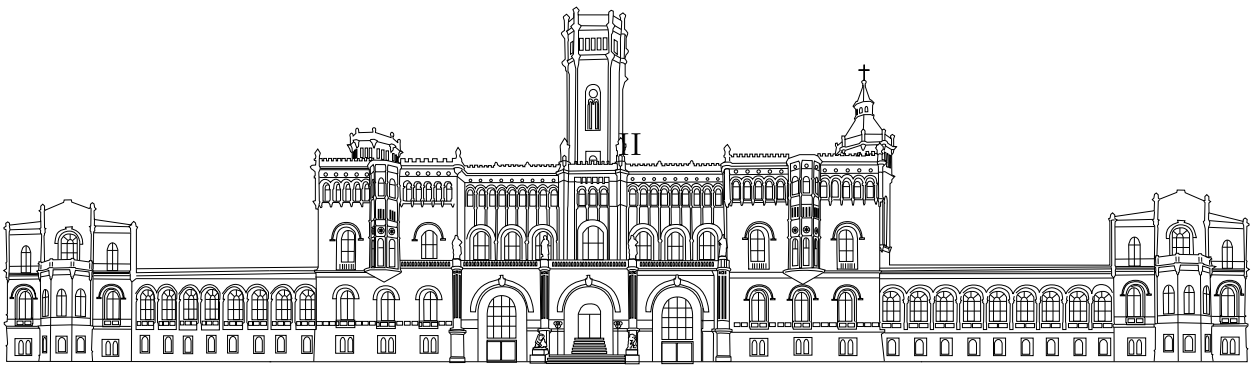
E-mail: moussab.hrou@stud.uni-hannover.de

First evaluator: Prof. Dr. Sören Auer

Second evaluator: Dr. Jennifer D'Souza

Supervisor: Omar Arab Oghli

23-September-2022



# Declaration of Authorship

I, Moussab Hrou, declare that this thesis titled, 'Evaluating SQuAD-based Question Answering for the Open Research Knowledge Graph Completion' and the work presented in it are my own. I confirm that:

- This work was done wholly in fulfillment of the requirements for the degree of Bachelor of Science in Computer Science at the Leibniz University Hannover.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Moussab Hrou

Signature: 

Date: 23-September-2022

# *Acknowledgements*

I would like to thank Dr. Jennifer D'Souza, my second examiner, for her constant support and guidance throughout this thesis project. Many thanks to o Prof. Dr. Sören Auer my first examiner for the opportunity to work on such an interesting project. I would also like to thank Omar Arab Oghli and Mohamad Yaser Jaradeh for supporting me on technical matters.

# *Abstract*

Every year, approximately around 2.5 million new scientific papers are published. With the rapidly growing publication trends, it is increasingly difficult to manually sort through and keep track of the relevant research – a problem that is only more acute in a multidisciplinary setting. The Open Research Knowledge Graph (ORKG) is a next-generation scholarly communication platform that aims to address this issue by making knowledge about scholarly contributions machine-actionable, thus enabling completely new ways of human-machine assistance in comprehending research progress.

As such, the ORKG is powered by a diverse spectrum of NLP services to assist the expert users in structuring scholarly contributions and searching for the most relevant contributions. For a prospective recommendation service, this thesis examines the task of automated ORKG completion as an object extraction task from a given paper Abstract for a query ORKG predicate. As a main contribution of this thesis, automated ORKG completion is formulated as an extractive Question Answering (QA) machine learning objective under an open world assumption. Specifically, the task attempted in this work is *fixed-prompt Language Model (LM) tuning* (LMT) for few-shot ORKG object prediction formulated as the well-known SQuAD extractive QA objective. Three variants of BERT-based transfomer LMs are evaluated. To support the novel LMT task, this thesis introduces a scholarly QA dataset akin in characteristics to the SQuAD QA dataset generated semi-automatically from the ORKG knowledge base. As a result, the BERT model variants when tested in vanilla setting versus after LMT, show a positive, significant performance uplift for automated ORKG completion as an object completion task. This thesis offers a strong empirical basis for future research aiming at a production-ready automated ORKG completion model.

*Keywords: Question Answering, Prompt-based Learning, Open Research Knowledge Graph, Knowledge Graph Completion, Link Prediction*

# Contents

# List of Figures

# List of Tables

# Acronyms

**KG** Knowledge Graph

**ORKG** Open Research Knowledge Graph [5]

**RDF** Resource Description Framework [14]

**SKG** Scholarly Knowledge Graph

**SQuAD** Stanford Question Answering Dataset [44]

# Chapter 1

# Introduction

Traditional search models over scholarly communication are now changing toward knowledge graph models operating on structured fine-grained scholarly content offering enhanced contextual search results. Several initiatives exist to this end: Google Scholar, Scopus [6], Web of Science [8], Microsoft Academic Graph [59], Research Graph Foundation [3], OpenAIRE Research Graph [35], Open Research Knowledge Graph [4], Crossref [24], Semantic Scholar [23] to name just a few. These knowledge graphs differ in their content, their level of detail, etc., as they strive to capture the diverse aspects of scholarly communication.

*Knowledge graphs (KG).* Their well-known utility is in offering enhanced contextualized search as demonstrated successfully in industry by Facebook [37] and by Google [1]; and even in the open data community by Wikidata [57] serving information over many general domains. One could say inspired from such KG success stories in the general domain, they are now being realized over scholarly knowledge as well, evidenced by the plethora of the afore-cited scholarly KG initiatives. The eager adoption or discovery of KG-based scholarly information access technology is fostered by our present era of the publications deluge [30, 29] when scientists seek more intelligent computer assistance to keep pace with the research volume.

*Open Research Knowledge Graph (ORKG)* [5]. A knowledge graph-based infrastructure that acquires scholarly knowledge in human and machine actionable form, as

opposed to the traditional document-based form. This opens new ways to machine assistance, which would immensely help research with the tasks of Literature comparison, and finding relevant contributions to their fields. The ORKG is a free service hosted at `thttps://www.orkg.org/`, and its code is available under an open source license under `https://gitlab.com/TIBHannover/orkg`

This thesis tackles the challenging problem of Scholarly Knowledge Graph completion. Specifically, the Open Research Knowledge Graph completion. This is addressed via a Question Answering task formulation. That is given an ORKG subject and predicate tuple as a question and the paper abstract as context information, the underlying machine learning model is expected to find the object resource by extracting the pertinent answer phrase from the given context.

Particularly, this thesis aims to evaluate the performance of state-of-the-art transformer-based language models originally developed on the objective for commonsense question answering (specifically, the SQuAD objective), for the Open Research Knowledge Graph completion task. For this, the transformer-based language models are tested out-of-the-box and are tested when finetuned on a small corpus of structured scholarly knowledge.

Based on the mentioned aims the research questions of this thesis formally explores the following three research questions (**RQ**):

**RQ1**) what is the most intuitive formulation of the question answering objective in terms of a knowledge graph completion objective;

**RQ2**) how do we encode the Question Answering objective in a domain-specific corpus based on the Open Research Knowledge Graph structured scholarly contributions; and

**RQ3**) how do transformer-based models perform on the Knowledge Graph completion task as vanilla models versus fine-tuned models?

The remainder of this work is structured as follows. Chapter 2 reviews the related work. Chapter 3 presents some key features of the ORKG. In chapter 4 the approach used to tackle the task at hand is presented. In chapter 5, the implementation of

the system to train and evaluate the transformer-based models is presented. The results of the training and evaluation on the ORKG data is presented and discussed in chapter 6.

# Chapter 2

# Related Work

## 2.1 Scholarly Knowledge Graphs

Recently, the impetus for implementing knowledge graphs over scholarly data has garnered quite a bit of attention, resulting in many knowledge graphs for various goals. Herein are discussed a selected few of the scholarly knowledge graphs.

**PID Graph** [21]. Developed in the context of the EC-funded FREYA project, the PID Graph is a federated graph with metadata about persistently identified articles, datasets, people, organisations and their relations. The GraphQL-based interface thus provides harmonised access to metadata curated by numerous infrastructures, in particular DataCite, Crossref, ORCID.

**OpenAIRE** [25]. Aims to provide a central entry point to publications and datasets funded by the European Commission and National agencies. Services such as the OpenAIRE Research Graph are offered to populate, curate, and enrich an Information Space based on metadata about organizations, data sources, projects, funding programmes, persons, publications, and datasets.

**OpenResearch** [55]. Is a MediaWiki[1] based service addressing researchers who

[1] `https://www.mediawiki.org`

search for and publish information on scientific events, as well as universities, information infrastructure institutions, specialised societies, publishers and funding agencies.

**ResearchGraph** [58]. A distributed network of graphs connecting scholarly works including data from data repositories, academic and grey literature, grants and funders, and researchers and research organisation information.

**SciGraph**.[2] The Springer Nature linked data platform that links metadata from across the research landscape, i.e. metadata about documents, people, places and relations of importance to the science and scholarly domain. In SciGraph, data sources are aggregated from within Springer Nature and key partners from the scholarly domain. They leverage a semantic Web technology stack for scalable and expressive enterprise-level metadata management. Their citations data links vocabularies of article types, subjects, articles, journals, books, chapters, conferences, funders, other organizational stakeholders, persons, and grants.

## 2.2 Automated Knowledge Graph Link Prediction

The pioneering research on knowledge graphs (KG) emerged over KGs in the general or commonsense domain. Some prominent examples of generic KGs are WordNet [20], YAGO [52], Freebase [9], NELL [7] and DBpedia [31] as well as commercial KGs such as Google's Knowledge Graph, Microsoft's Satori and Facebook's Open Graph. As such with the invention of these KGs, several machine learning objectives around mainly KG construction and KG completion were defined. Given the recent emergence of scholarly domain-specific KGs as elicited in the previous section, these machine learning objectives would need to be transferred over scholarly KGs. In this respect, since this thesis examines KG completion as Link prediction with an open world assumption, we review existing machine learning approaches. To elaborate, the organization of triples in the form of (head entity, relation, tail entity) comprises the construction of KGs. Following which, inferring new objects given the head entity and relation and some context is the problem addressed in this work.

[2] `https://www.springernature.com/scigraph`

### 2.2.1 Knowledge Graph Embedding (KGE) Models

KG embedding is the task of completing the KGs by probabilistically inferring the missing arcs from the existing graph structure. This spectrum of machine learning approaches involves representation learning of KGs by embedding both entities and relations into a low-dimensional vector space aiming to predict unknown triples based on previously visited triples. KGE differs from ordinary relation inference as the information in a KG is multi-relational and more complex to model and computationally expensive. There are many popular KGE models such as TransE [11], TransH [61], TransD [28], TransR [32], RESCAL [36], DistMult [63], ComplEx [54], and RotatE [53], which define different score functions to learn entity and relation embeddings. For instance, the translational-based (Trans) models try to find a low-dimensional vector representation of entities in relation to the translation of entities. However, KGE methods are largely used for predicting missing links between existing entity pairs in a KG. This work goes beyond this application domain by predicting new objects from a certain context given existing related subject and predicate pairs, thereby adopts an open world assumption. The use of KGEs in this application domain remains limited since these models are leveraged within the closed-world assumption [2]. Instead, language models fitted with a question answering objective, discussed next, prove particularly suited to our task.

### 2.2.2 Natural Language (NL) Question Answering (QA)

NL QA tasks are categorized as three main types: abstraction, extraction, and retrieval [13]. The first type, abstraction, which is also the most challenging form of QA, requires an answer to be generated in natural language free form without necessarily relying on given context. Extraction is a slightly less advanced task in that the answer needs to be extracted from a given context by identifying and selecting pertinent parts of it. The simplest form of QA is based on retrieval, where the goal is to select an answer to a given question by ranking a number of short text segments, usually passages. We review related work on extractive QA as this is the focus of this work. In particular, we consider the SQuAD dataset [44]. The Stanford Question Answering Dataset (SQuAD) was introduced as a reading comprehension dataset consisting of 100,000+ questions posed by crowdworkers on a set of Wikipedia articles, where the answer to each question is a segment of text from the corresponding reading passage. In fact, we formulate our task per the SQuAD objective as follows: the answer to each question (formulated as related subject and predicate pairs) is a segment of text from the corresponding scholarly article's Abstract. Furthermore the SQuAD dataset partitioned answers in the following 10 categories: date; other

numeric; person; location; other entity; common noun phrase; adjective phrase; verb phrase; clause; and other. This methodology was also adopted by us for our scholarly QA dataset and tailored to the suitable 12 categories that were observed in our dataset.

### Prompt-based Learning Paradigm

The prompt-based learning paradigm [33] is increasingly adopted in the NLP community these days for obtaining fine-tuned versions of powerful, transformer-based QA language models such as GPT-3 [22] or BERT [18] model variants on domains with relatively less training data.[3] The method of prompting the large language models [62] was introduced to fine-tune them in scenarios where large amounts of training data is unavailable. Thus this method reduces the amount of data needed to fine-tune the large language models [45] and are used instead. While prompts benefit the overall performance, their design does not follow a specific rule. Thus in the SQuAD QA scenario, a prompt is constructed in the format similar to the training dataset. A set of tokens represent the question and answer pair, and a short text representing the context are given as input to the model. Such a task is also called 'prompt engineering.' QA tasks have been shown to be improved by few-example prompts [12]. This thesis constructs a small QA dataset and leveraged the effective prompt-based method [46] of finetuning BERT model variants for scholarly knowledge graph completion formulated as a QA task predicated on the predicate labels as the question, object labels as the answer, and the corresponding paper abstract as the context.

[3] https://blog.paperspace.com/prompt-based-learning-in-natural-language-processing/

# Chapter 3

# Background

## 3.1 The Open Research Knowledge Graph

Since the Open Research Knowledge Graph completion is the objective of this work, we explore some of the key features of the ORKG in this chapter. We also present some information regrading The Stanford Question Answering Dataset.

**Paper and contributions**. A researcher can create an ORKG paper, which will be a representation of a scholarly article. First, metadata about the article is entered either manually or by looking up this data using the paper DOI. The second option also has the advantage of checking whether a paper with the same DOI already exists in the ORKG. Most importantly, information regarding the content of the scholarly article such as research problem, materials, methods and results is organized within paper contributions using contribution editor in the ORKG as shown in Figure 3.1.

**Comparison**. Is the core feature and contribution of ORKG project. Using this feature, Contribution related to the same research problems can be organized in a tabular format. The property fields of the contribution can be rearranged to have the most import ones, for example at the top. Furthermore, it is possible to filter the contributions displayed based on the values of the properties using a very simple user interface. An ORKG comparison example is shown in Figure 3.2 and the filtering feature in the same comparison in Figure 3.3.

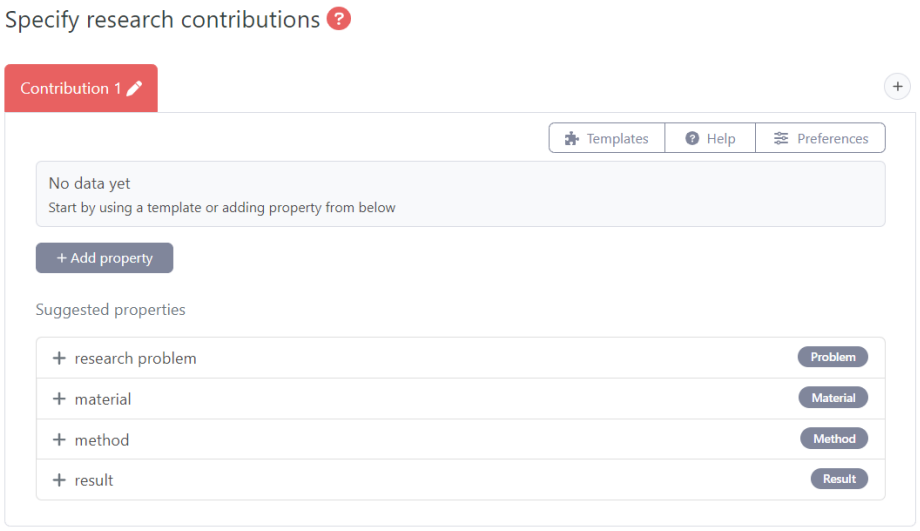



Figure 3.1: The ORKG contribution editor interface

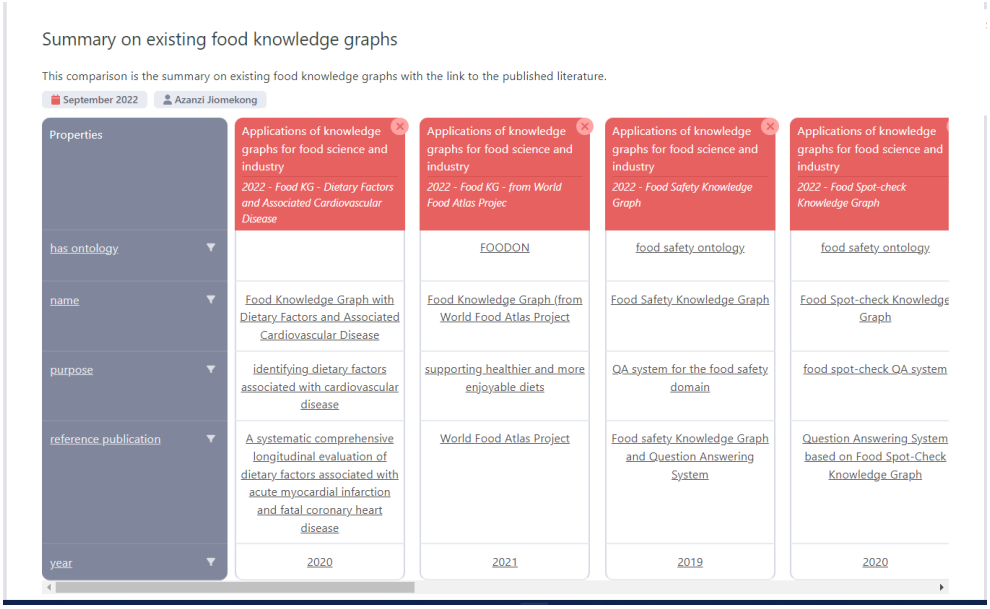



Figure 3.2: An example of a comparison in the ORKG

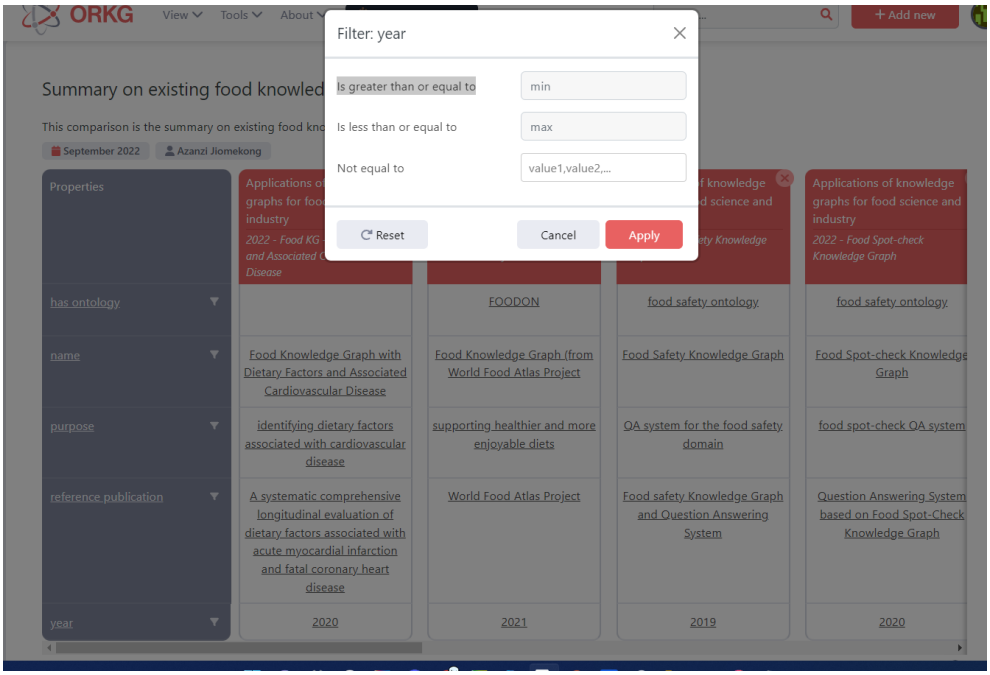



Figure 3.3: An overview of the filtering feature in the ORKG comparisons

The ORKG has other features such as Views, Lists, visualizations and other. These are not of interest for us here and so will not be presented. Further information is available in `https://orkg.org/about/19/Templates`.

Some general statistics about the ORKG data as of 23-09-2022 are presented in Table 3.1.

| **metric** | **value** |
|---|---|
| number of papers | 11860 |
| number of contributions | 17853 |
| number of research problems | 4847 |
| number of research fields | 709 |
| number of comparisons | 942 |

Table 3.1: General stats about the ORKG data.

## 3.2 The Stanford Question Answering Dataset

The Stanford Question Answering Dataset (SQuAD) is a collection of question-answer pairs derived from Wikipedia articles. The reading comprehension dataset is first introduced in the paper "SQuAD: 100,000+ Questions for Machine Comprehension of Text" [43] in 2016. The datasets consist of more than 100000 questions formulated by crowd-workers on a set of Wikipedia articles.

The SQUaD1.1 is created for the task of machine reading comprehension, which refer to the task of reading text and answering questions about it. The datasets that existed before the SQUaD dataset were either too small and of high quality, or large while being semi-synthetic, which different characteristics of manually created datasets. The dataset was created to solve these issues, by created a large dataset with questions manually posed by humans. To ensure this, only the top 10000 articles of English Wikipedia. From those passages are extracted. Crowd-workers were then tasked with reading the passages and posing questions, and then highlighting the answers in the texts. Additionally, these answers are quite diverse as shown in the Figure 3.4

| Answer type | Percentage | Example |
|---|---|---|
| Date | 8.9% | 19 October 1512 |
| Other Numeric | 10.9% | 12 |
| Person | 12.9% | Thomas Coke |
| Location | 4.4% | Germany |
| Other Entity | 15.3% | ABC Sports |
| Common Noun Phrase | 31.8% | property damage |
| Adjective Phrase | 3.9% | second-largest |
| Verb Phrase | 5.5% | returned to Earth |
| Clause | 3.7% | to avoid trivialization |
| Other | 2.7% | quietly |

Figure 3.4: The distribution of the answer categories of the SQUaD1.1 dataset [43]

The SQUaD2.0 [42] was created by combining the SQUaD1.1 dataset with additional 50,000 unanswerable questions written also by crowd-workers. This is done in order

to account for the fact that sometimes answers don't exist in a text. And so in order to do well on the SQUaD2.0 set, models should not only be able to locate the answers, but also determine whether an answer exists or not, and if not abstain from answering.

The SQUaD datasets can be downloaded, and explored in `https://rajpurkar.github.io/SQuAD-explorer/`

# Chapter 4

# Approach

In this chapter, we formulate the question answering objective in terms of a knowledge graph completion objective, and then we discuss various steps needed for the ORKG dataset creation.

## 4.1 Task Formulation

From the Background chapter, we can see that most if not all the information pertaining to the content of a research paper lie within the contribution sections of said paper. Because of this, the contributions are the base of key features in the ORKG, such as the comparison feature. It follows then that for the knowledge graph completion, we focus mainly on the completion of the data within the contributions in the ORKG.

A contribution consists of properties. A property has a property key and a property value. There are predefined properties in the ORKG such as "has method", "research field", "has approach" and so on. The users are encouraged to use these predefined property keys, but they also have the option to define additional ones if needed. Entering the property values are the responsibility of the users alone. With this, we define the question answering objective as follows:

Given a question answering model **M**, a paper abstract as a context **C**, a property

key from a contribution of that paper as a question **Q**.

$$M(C, Q) = A \tag{1}$$

How good is the answer **A** (the property value) predicted by the model given the C and Q as inputs.

For the prediction, we choose transform models pre-trained on the question answering datasets SQuAD1.1 and SQuAD2.0. Additionally, we further fine-tune these models using data extracted from the ORKG and processed to fit the structure of a question answering dataset.

In the next section, we discuss the various steps required to create and process the ORKG dataset. We also present key stats about it. Finally, we present some key information regarding the script used to create this dataset.

## 4.2 Corpus

The Data in the ORKG is structured as a knowledge graph, that consists of nodes and links that represent the relations between the nodes. The nodes can be subjects or objects, in this way the graph can be viewed as a directed graph. It is worth nothing that a subject node can also be an object node in another subject object relation, it is simply the matter of which link we consider. In the ORKG system, the links are called predicates. The ORKG graph can therefore also be represented as a set of subject predicate object triples. A triple is called a statement in the ORKG system.

For instance, the example graph shown in Figure 4.1 can be serialized into the following statements:

- **subject**: Paper 1, **predicate**: has contribution, **object**: contribution 1
- **subject**: Paper 1, **predicate**: has contribution, **object**: contribution 2
- **subject**: contribution 1, **predicate**: has research problem, **object**: research problem 1

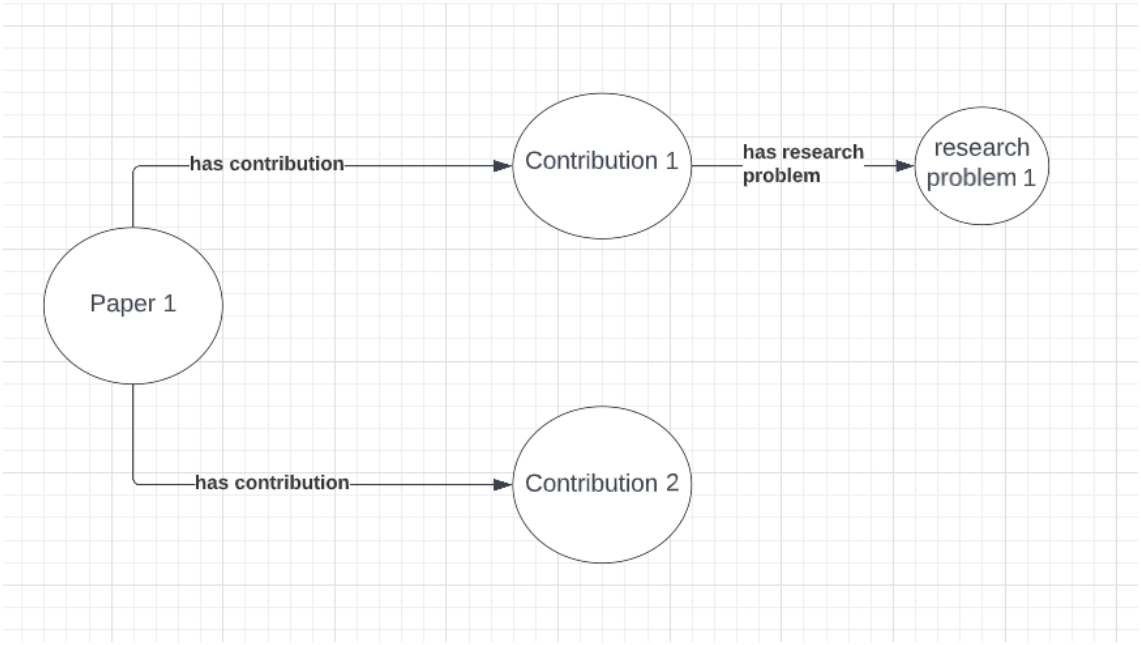



Figure 4.1: A simple example a part of a knowledge graph

The statement format (triples) is very important, since it is the only format the ORKG API [38] returns data in. Because of this, additional work might be necessary to restore the relationships between different parts of the graph. Consider the example discussed above, in order to restore the link "Paper 1" to "research problem 1" shown in the graph in Figure 4.1 one must first query for the statements where the subjects are papers and then filter the results by the object "Contribution 1" and then query again for the statements where type contribution is the subject, and then merge the data. The issue is more obvious for the more complex cases that we will tackle in the following subsections.

As mentioned in the introduction of this section, the objective at hand is the ORKG completion, and more precisely the completion of the data in the contribution section of each paper in the ORKG. Each paper in the ORKG can have one or more contributions in which the author enters keys information pertaining to the topics and research problems that the paper contributes to. Examples of such information are date and location of an experiment, the results, the research problem, sample sizes and such details. An actual graph representation of a contribution from the ORKG is depicted in Figure 4.2.

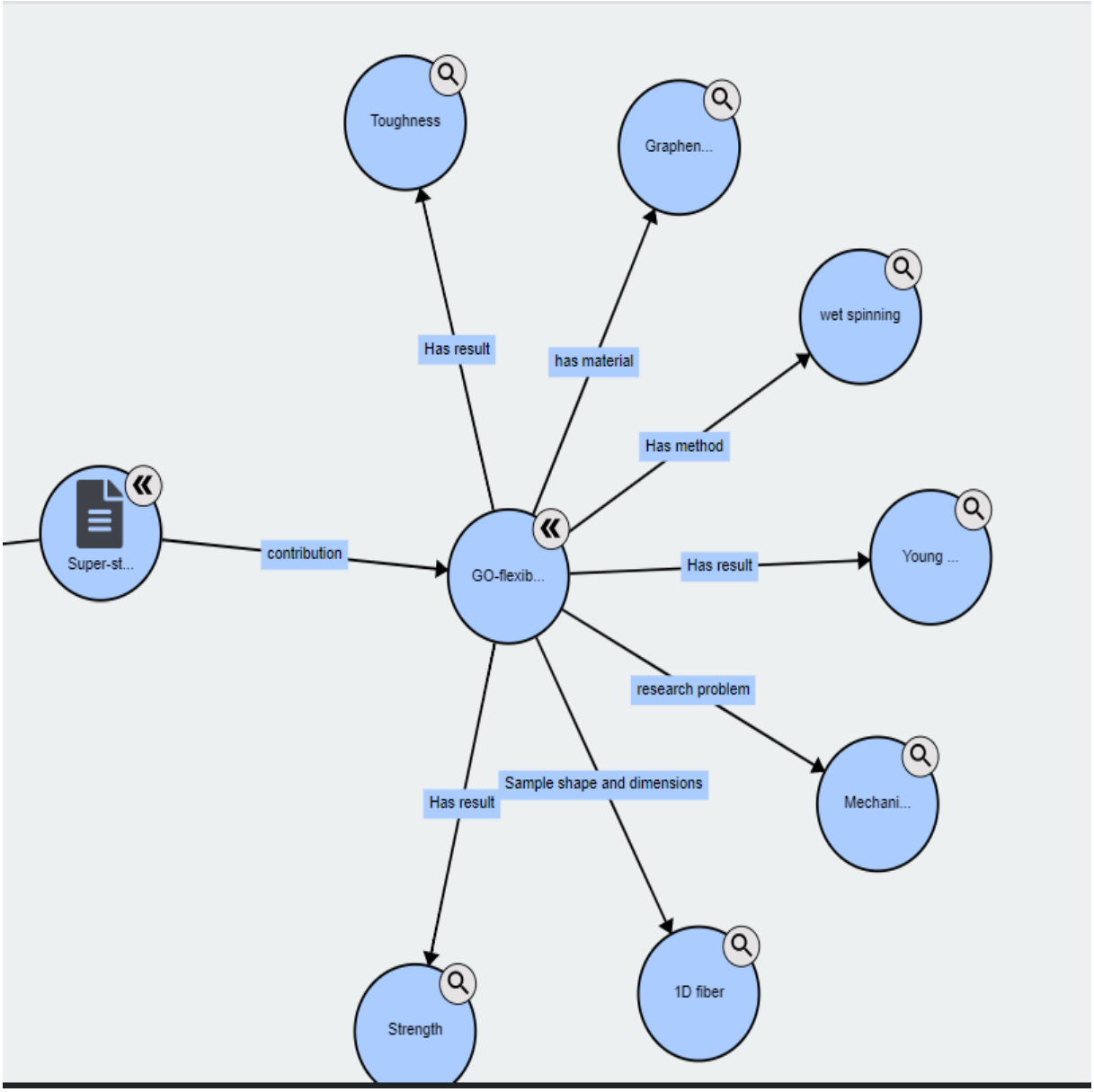



Figure 4.2: The graph view of a paper contribution in the Open Research Knowledge Graph, Taken using the "view graph" feature of the ORKG.

Looking at the statements where the contribution is the subject, the contribution property keys are the predicate labels in those statements, and the contribution property values - which would be the target of the predictions - are the object labels. The following fields will therefore be required for each data point we extract from the ORKG in order to fine-tune the transformer models.

- Paper title: will be mainly used to fetch the paper abstract later on, since the abstracts are not stored in the ORKG or any other related service.

- Paper abstract (not extracted from the ORKG): will serve as the context in which the models would search for answers. This presuppose that the answers are contained within the abstract itself.

- Predicate label: in the statements where the contribution is the subject. the predicate label will be processed to serve as the question.

- Object label: in the statements where the contribution is the subject. The object label will serve as the expected answer.

These fields will be expanded with an additional field "object label category" which will serve to categorize object labels and to draw more information regarding the performance of the models during the evaluation phase.

Question answering models can be trained on questions that have answers in the corresponding context, but also on questions without answers. The models are therefore trained to not only extract the answers, but also decide whether an answer exists or not. In our case, since the object is the completion of the ORKG we only fine-tune the models on questions where the answers exist in the paper abstracts. As a consequence, The performance of the fine-tuned models in a production environment depends heavily on how much information about the paper is contained within the abstract. This means that in the case where there are no answers to a question (predicate label) within an abstract, the models would still return answers, which might decrease the usability of these models.

In the next subsections, we discuss the various steps followed to construct and process the ORKG dataset. At the end, an overview of the script used to create the dataset

is presented.

### 4.2.1 Querying And Mapping ORKG Data

For the Querying of the ORKG data, the python ORKG client [39] was used. The client helps to interact with the ORKG API [38]. First a connection to the API needs to be established using the ORKG login info (email and password), then the API can be queried. The API client provides a pagination feature, that allows to receive results in pages of configurable number of items, for example 1000 items per page. This feature is quite helpful, since the API doesn't provide the option for bulk queries. Because of this, querying all statements using the pagination feature and then filtering them is much faster than querying multiple times for specific statements.

The first step in the corpus creation is constructing the Research fields to contributions mapping. This is done in order to gain some insight into the distribution of the contributions over all the research fields in the ORKG. This is achieved first by querying for all statements with subjects of type "Paper" and then filtering those to only keep the statements with predicate label "research field" which has the ID "P30". With this data, we can create a research field to papers mapping. Then for each paper we query for statements where the predicate label is "contribution" which has the ID"P31". The object labels are then the contributions that we use to create a paper to contributions mapping. We finally merge the two mappings based on the paper ID. the distribution of the contributions is shown in the Figure 4.3. It is clear that the distribution as seen in the graph follows Zipf's law, which indicate the number of contributions is inversely proportional to the rank of the research field. For instance, the "Bioinformatics" research field in the first position has 2257 contributions while "Ecology and Evolutionary Biology" in the second position has 1251 and "Information Science" is in third position with 619.

At the time of the research (22-09-2022), there were around 710 research fields and around 9379 papers in the ORKG.

The next step is getting the predicate and object labels for each contribution. For this, we query for each contribution the statements where that contribution is the subject. The existing paper to contribution mapping is then expanded with the predicate and object label extracted from these statements. At this stage, each row

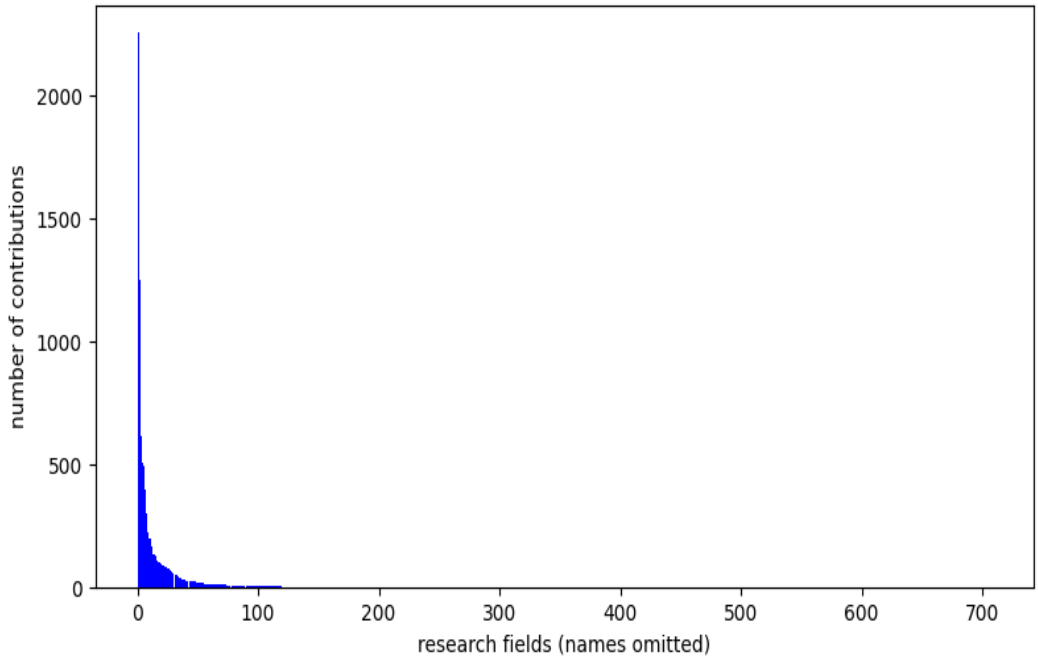



Figure 4.3: The distribution of the ORKG contributions per ORKG research field as of 22-05-2022

in the dataset consists of paper ID, paper title, contribution ID, predicate label and object label. Some key stats about the data collected so far is shown in Table 4.1.

| **metric** | **value** |
|---|---|
| number of data points (rows) | 116421 |
| number of unique papers | 9379 |
| number of unique contributions | 14499 |
| number of unique predicate labels | 3436 |
| number of unique object labels | 38234 |

Table 4.1: key stats of the initial dataset collected during the steps described in section 4.2.1

### 4.2.2 Fetching Abstracts

A key data that is need for the question answering task we have at hand is the context where the answers are looked for. The paper abstracts would serve as context in our case, but the ORKG doesn't store abstracts, and so we have to rely on external services in order to fetch them. The code we use to fetch the abstract is written by Omar Arab Oghli [50] as part of the ORKG project, and make use of the semanticscholar API [49] and the crossRef API [16] services to search and get the abstracts if they exist. To search for an abstract, one can use the paper title or the paper DOI (Digital Object Identifier), that can be queries from the ORKG API by searching for statements with subject "paper id" and then filtering for predicate label "had doi" with the ID "P26". In our approach we first search by paper title and then if no abstract is found we search by the DOI. From the work here, the two methods almost yield equal results regarding the number of abstracts that we were able to fetch. At the end we were able to find abstracts for 5486 (58,5%) out of the 9379 papers we started with.

Since the Abstracts are absolutely required for the training, we omit any row from the dataset if no abstract is found for the paper corresponding to it. Additionally, since we are only considering the case where the expected answer does exist in the context (abstract), we also omit any rows where the object label cannot be located inside the abstract. It is quite important to note here that some object labels can be short words such as the acronym "STR" and so to check if this acronym actually exists in the abstract, a simple containment check might not be good enough since it might return True for words like "structure", instead tighter checks for instance using regular expressions that check for the start and end of the string are more suited here.

| **metric** | **value** |
|---|---|
| number of data points | 116421 |
| number of data points with abstracts | 67229 |
| number of data points with abstracts and object label can be found in the abstract | 14499 |

Table 4.2

As shown in Table 4.2 we started with 1164421 rows in the dataset, only for 67229 (57.7%) of those were we able to fetch abstracts, and among those only 14499 where

the object label is found in the abstract. This leaves us with only 12.5% of the dataset we started with to be used for training later. This is of course an area where further improvements could be achieved, for instance by requiring the users to provide abstracts for their papers to the ORKG.

### 4.2.3 Data Cleaning

The first thing we clean from the 14499 rows dataset we have so far is duplicates. When looking at all the fields in each row, there are no identical rows, but considering the question answering goal, we mainly care about the abstracts (contexts), the predicate labels (questions) and the object labels (answers), and so any row that have the same triples (context, questions, answers) are considered identical and should de-duplicated. The paper IDs can substitute for the abstract here since each paper should have a unique abstract, and so we consider the triple (paper ID, predicate label, object label) as a base for the de-duplication. The duplicates exist because a paper can have multiple contributions that share certain information, such as the research field.

The next thing we clean out from the dataset are rows with unsuitable object labels. The following is a list of criteria used to decide if a row with an object label is to be omitted.

- whole numbers from 0 to 999.
- Hyphens -
- Alphabet characters: A, B, C ...
- Boolean values: T, F, no, y, n.
- Not applicable: na.
- Common short words: “all”, ”and”, ”or” and so on.
- Also repeated phrases like “Any track”.

The reason these object labels are omitted is because they are ambiguous and don't usually point to a single position in the abstracts, which might affect the training results negatively.

At the end of this step, we are left with 5909 rows in the dataset.

### 4.2.4 Object Label Categorization

**Object label categorization**. The next step in the dataset construction is the categorization of object labels. This is done in order to gain insight into whether the AI performance varies depending on each category. Inspired by the categories used for the SQUaD dataset shown in [44] and the observation of the ORKG data, we chose to go with 12 categories presented in Table 4.3. The categories are assigned mostly automatically using defined heuristics as described below.

- research problem: any object label in a row with the predicate label "has research problem".
- url: any object label that starts with "http".
- location: any object label with predicate label in ["country, city, location, continent", "has location", "study location", "countries"].
- year/date: any whole number above between 1000 and 2100.
- number: any object label consisting of only digits after removing the minus sign (-), the dot (.) and the comma sign (,).
- count/measurements: any object label that contains at least a number in addition to another string(s). For instance, "5 meters".
- noun: the check whether a label is a noun was done automatically using Natural Language Processing pipelines, specifically the spacy python module [51], that can analyse a string and return the POS (part of speech) of each of its tokens. Nouns have exactly one token.

- adj: similar to the noun category.
- acronym: object labels with one upper cased token.
- noun phrase: object labels with less than 5 tokens and the last token is a noun.
- adj phrase: object labels with less than 5 tokens and the last token is a adj.
- sentence: any object label with more than 4 tokens and doesn't fall under any other category.

The checks above can only work correctly if they are implemented in the same order as they are mentioned in the list. And so, once a match is found for an object label, we assign the category and move to the object label in the next row. This is because some object labels can be a match to multiple categories. For instance, "2000" can be both a year and a number, but we decided to interpret all integers between 1000 and 2100 as years instead of numbers.

Table 4.3 shows the object label categories, their percentages in the dataset and an example of each category. From this table, we can see that 58.9% of the object labels are either nouns or noun phrases. It is noteworthy, that a manual check of the mappings is still needed since the heuristics and the spacy pipeline used to partition the object labels don't guarantee a 100% correctness. This is especially true for categories such as "acronym" and "location".

After The categorization step we end up with a 5909 rows dataset with further stats shown in Table 4.4. The number of abstracts with more than 510 tokens is included in the table because the models we are going to use have a maximum number of tokens for the input of 512 tokens. And when we consider the start and end of input tokens that will be appended later on, we are left with 510 tokens for the content of each abstract. The inputs with more than 510 tokens are truncated by the models. In our case, we have 37 rows (out of 5909)that have abstracts with more than 510 tokens. This is of course very minimal, and so we decided that this is not an issue and not to omit these rows. However, if the number is higher, then one can for instance trim the abstracts while making sure the answers inside are not trimmed out, in order to avoid the automatic truncation done by the models.

The avg number of tokens per object label of 2.43 will serve later on as a benchmark for the avg number of tokens returned by each model, more on that in section 6.3.3.

| **object label category** | **percentage in the dataset** | **example** |
|---|---|---|
| noun | 29.85 | Transistors |
| noun phrase | 28.74 | data mining |
| acronym | 10.12 | HMM |
| research problem | 9.53 | Performance of thin-film transistors |
| adj | 4.72 | high |
| Location | 4.37 | Serbia |
| number | 3.93 | 4977 |
| count/measurement | 3.47 | 2.45 GHz |
| sentence | 2.76 | raw data dumps and HDT files |
| year/date | 1.95 | 2011 |
| url | 0.3 | https://github.com/giannisnik/mpad |
| adjective phrase | 0.27 | Unsupervised and Adaptive |

Table 4.3: The object label categories resulted from the step described in section 4.2.4 with their percentages and example from the actual dataset.

**Training and evaluation datasets**. The training dataset consists of 4745 data points (82% for the entire dataset) while the evaluation set consists of 1036 data points (18%). In order to have an evaluation set that is representative of the training set, the data was split on the basis of the predicate label (the question in the context of QA) using the following heuristic

- For predicate labels with less than 10 data points, we assign the corresponding rows to the training set.
- For predicate labels with 10 or more data points, we assign the 74% of the corresponding data to the training set and the rest to the evaluation set.
- Note: the 74% value what chosen in order to have a training to evaluation ratio of around 80:20.

The first few rows of the final dataset (not split) are shown in Figure 4.4. The

| metric | value |
|---|---|
| number of rows | 5909 |
| number of unique papers | 2710 |
| number of unique categories | 12 |
| number of unique predicate labels | 853 |
| number of unique object labels | 3524 |
| avg number of tokens per predicate label | 2.01 |
| avg number of tokens per object label | 2.43 |
| number of unique abstracts | 2649 |
| avg number of tokens per paper abstract | 196.97 |
| number of abstracts with more than 510 tokens | 37 |
| number of unique abstracts with more than 510 tokens | 14 |

Table 4.4: The final ORKG dataset stats

complete dataset can be found in `https://huggingface.co/datasets/Moussab/ORKG-training-evaluation-set`. Note that not all some fields such are "subject_id" and "statement_id" were used to map the different fields together, and still in the dataset only for debugging purposes.

### 4.2.5 Data Preparation Script

The data preparation and the model training script can be accessed in this public repository `https://github.com/as18cia/thesis_work`. The Readme file in this repository provides detailed information on how to run the script.

The data preparation part of the script is structured as a waterfall as shown in Figure 4.5. This means it consists of several steps, each step takes some files as input and produces a file as output, the next step uses the output of the last as input and so on. This script is structured this way for several reasons. First a lot of the steps take a lot of time to finish and so this way in the case of failure, we don't have to repeat all the steps. This is because as long as the file produced by a step is there, this means that the last step was completed. This way we can simply repeat the steps that didn't produce any files. Also having the data in files after each step helps to get to know the dataset and to generate stats about the data in each step.

The script was written in the python programming language [41]. This is because

| research_field_id | research_field_label | paper_id | paper_title | statement_id | subject_id | predicate_label | object_id | object_label | paper_abstract | object_in_abstract | ner |
|---|---|---|---|---|---|---|---|---|---|---|---|
| R133 | Artificial Intelligence | R9567 | A Neural Approach for Te | S16055 | R9568 | has research problem | R10012 | scene text detection | In recent years, the prob | TRUE | research problem |
| R133 | Artificial Intelligence | R9567 | A Neural Approach for Te | S16054 | R9568 | has research problem | R10011 | text extraction from ima | In recent years, the prob | TRUE | research problem |
| R136 | Graphics | R9557 | An ontology of scientific | S15486 | R9558 | Ontology | R9559 | EXPO | The formal description o | TRUE | acronym |
| R136 | Graphics | R9548 | The Publishing Workflow | S15423 | R9549 | Ontology | R9550 | PWO | . In this paper we introdu | TRUE | acronym |
| R136 | Graphics | R9545 | The Document Compone | S15398 | R9546 | Ontology | R9547 | DoCO | The availability in machir | TRUE | noun |
| R136 | Graphics | R9524 | An ontology of scientific | S15253 | R9525 | Ontology | R9526 | EXPO | The formal description o | TRUE | acronym |
| R136 | Graphics | R9515 | The Publishing Workflow | S15190 | R9516 | Ontology | R9517 | PWO | . In this paper we introdu | TRUE | acronym |
| R136 | Graphics | R9512 | The Document Compone | S15165 | R9513 | Ontology | R9514 | DoCO | The availability in machir | TRUE | noun |
| R145 | Environmental Sciences | R9221 | The ACCESS coupled mo | S14620 | R9222 | has name | L8976 | ACCESS1.0 | 4OASIS3.2â€"5 coupling | TRUE | acronym |
| R145 | Environmental Sciences | R9221 | The ACCESS coupled mo | S14632 | R9228 | has name | L8977 | ACCESS1.3 | 4OASIS3.2â€"5 coupling | TRUE | acronym |
| R145 | Environmental Sciences | R9221 | The ACCESS coupled mo | S14630 | R9228 | has research problem | R9223 | CMIP5 | 4OASIS3.2â€"5 coupling | TRUE | acronym |
| R145 | Environmental Sciences | R9221 | The ACCESS coupled mo | S14637 | R9228 | Earth System Model | R9230 | Ocean | 4OASIS3.2â€"5 coupling | TRUE | location |
| R145 | Environmental Sciences | R9221 | The ACCESS coupled mo | S14693 | R9228 | Earth System Model | R9273 | Atmosphere | 4OASIS3.2â€"5 coupling | TRUE | noun |
| R145 | Environmental Sciences | R9221 | The ACCESS coupled mo | S14694 | R9228 | Earth System Model | R9274 | Land Surface | 4OASIS3.2â€"5 coupling | TRUE | noun phrase |
| R142 | Earth Sciences | R9094 | Development and evalua | S14329 | R9095 | has research problem | R9138 | CMIP5 | Abstract. We describe he | TRUE | acronym |
| R142 | Earth Sciences | R9094 | Development and evalua | S14293 | R9095 | Earth System Model | R9104 | Ocean | Abstract. We describe he | TRUE | location |
| R388 | Comparative Literature | R8624 | Revisiting Style, a Key Co | S13529 | R8625 | Has result | R8628 | Definition of style | <jats:title>Abstract</jats | TRUE | noun phrase |
| R388 | Comparative Literature | R8624 | Revisiting Style, a Key Co | S13528 | R8625 | has research problem | R8627 | Definition of style | <jats:title>Abstract</jats | TRUE | research problem |
| R136 | Graphics | R8356 | The anatomy of a nanop | S13008 | R8357 | Semantic representation | R8358 | Nanopublications | As the amount of scholai | TRUE | noun |
| R136 | Graphics | R8348 | Research Articles in Simp | S12967 | R8349 | Semantic representation | R8350 | RASH | <jats:sec><jats:title>Purp | TRUE | acronym |
| R136 | Graphics | R8345 | Decentralised Authoring, | S12942 | R8346 | Semantic representation | R8347 | Dokie.li | Abstract While the Web | TRUE | noun |
| R136 | Graphics | R8330 | An ontology of scientific | S12789 | R8331 | Ontology | R8332 | EXPO | The formal description o | TRUE | acronym |
| R136 | Graphics | R8312 | The Publishing Workflow | S12726 | R8313 | Ontology | R8314 | PWO | . In this paper we introdu | TRUE | acronym |
| R136 | Graphics | R8301 | The Document Compone | S12701 | R8302 | Ontology | R8303 | DoCO | The availability in machir | TRUE | noun |

Figure 4.4: An overview of the constructed ORKG dataset

python provides a lot of packages that facilitate the development of AI and data science type of tasks. And while python is much slower than languages such as java [27], the task at hand doesn't require a high performance language. And so, python with its simpler syntax, that improves productivity, is a perfect option.

Among the many useful python packages, one that is heavily used in the script is the pandas DataFrame package [40]. This package provides a lot of features to help in processing tabular data. For example it makes it easier to store tabular data in a CSV format, or reading a CSV file into a python object. It also provides very simple interfaces to calculate stats regarding the dataset, such as the number of unique items, means, averages and so on.

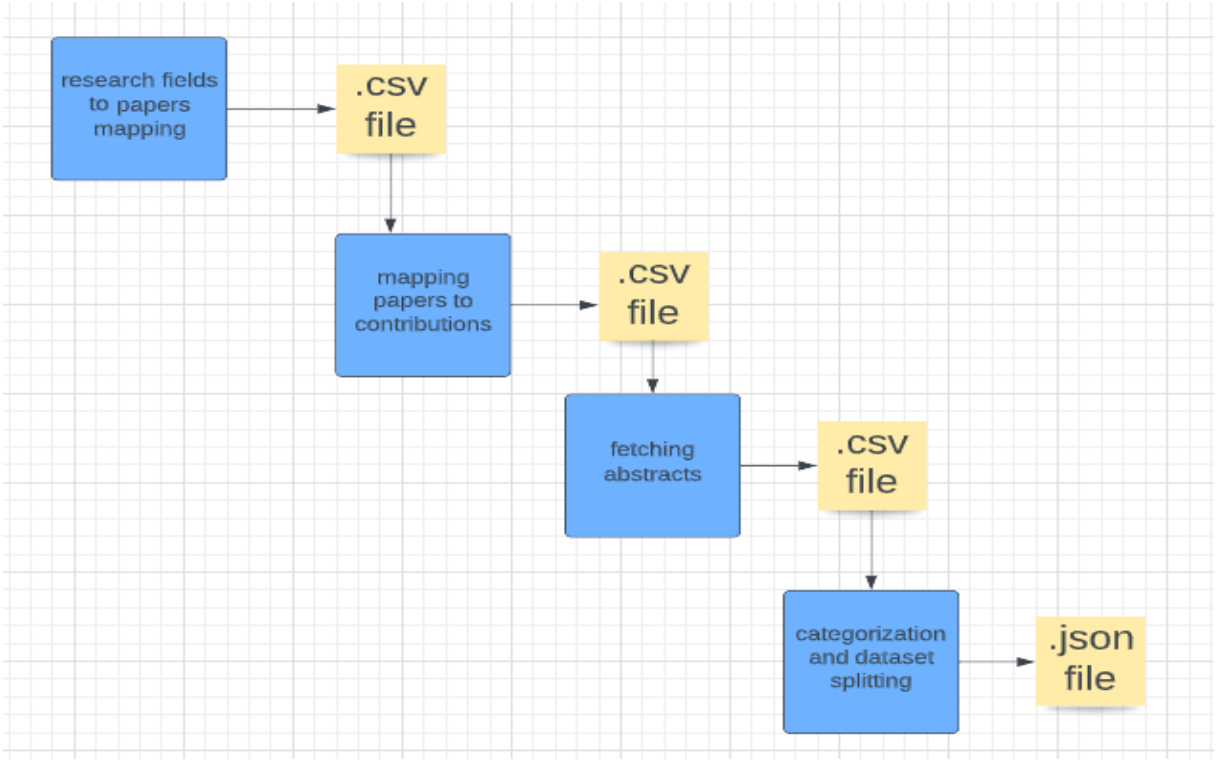



Figure 4.5: The structure of the data preparation script. This shows the general structure and so some steps are not shown.

# Chapter 5

# Implementation

A transformer model is a neural network that learns context and thus meaning by tracking relationships between tokens in sequential data, like the words in a sentence. Transformers were first introduced in 2017 in the "Attention Is All You Need" paper [56] and have since achieved great success in tasks such as Natural language processing, under which the Question answering task we have in this work lies.

Transformers are used for a wide variety of problems, ranging from natural language processing tasks such as question answering, natural text translation to a computer vision, Trajectory Forecasting and traffic flow forecasting. But also for tasks such as protein structure prediction.

Before transformers, recurrent neural networks (RNN) [48] were the go-to solution for natural language processing. An RNN processes a sequence of words by processing the first words and feeding back the result to the layer that will process the next word. In this manner, the RNN keeps track of the entire sequence instead of processing one word at a time as is the case with the classic feed-forward neural networks, that were only designed to map one input to an output which is not optimal for natural language, where word are related to each other and each word can have different meanings in different contexts.

RNN had however some disadvantages. First, they were slow, since they had to process input sequentially. Second, they couldn't handle long sequences of text. As the RNN got deeper into a text excerpt, the effects of the first words of the sentence gradually faded. This problem, known as "vanishing gradients," was problematic when two linked words were very far apart in the text. Consider the example the sentence "Bob was born in England. He is 20 years old, and is a student. Bob wants to become a software developer and later start his own business. Bob is very fluent in ....", by looking at the word England at the beginning of the text we can infer that the last token should be English, but since the distance between the two tokens are big when the RNN reaches the last token it might 'forget' the context. Third RNNs are unidirectional, this means they can only capture the relation between a word and the words before it in the sequence. In the sentence "The mouse is used to ... many things in a computer" the RNN can only check the words before the token to be predicted, and so the model cannot decide whether the animal mouse is meant or the computer device.

Transformers have the advantage of being able to deal with the issues mentioned above. First, they made it possible to process entire sequences in parallel, making it possible to scale the speed and capacity of sequential deep learning models to unprecedented rates. And second, they introduced "attention mechanisms" that made it possible to track the relations between words across very long text sequences in both forward and reverse directions.

For the task at hand, we choose 3 BERT based models pre-trained on the SQUaD dataset. BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based machine learning model for natural language processing introduced in 2018 [18]. Bert is designed to be pre-trained using unlabeled text, which allows the model to be fine-tuned to achieve state-of-the-art performance for a range of tasks. Indeed, the Bert model has achieved state-of-the-art results in tasks such Question answering on the SQUad dataset and Sentiment Analysis on several languages [15].

## 5.1 Transformer Model Variants

BERT has inspired the development of many variations, such as Roberta, Hubert, TinyBERT, Distillbert and more. The model we are going to fine-tune for our task are models based on Roberta, Distillbert and MiniLIM. in the next subsection we

present the 3 models we are going to train deepset/minilm-uncased-squad2 [60], deepset/roberta-base-squad2 [34] and distilbert-base-cased-distilled-squad [47].

### 5.1.1 deepset/roberta-base-squad2

The deepset/roberta-base-squad2 model [17] is a transformer model pre-trained on the reunion of many datasets such as the BookCorpus [10] and English Wikipedia and fine-tuned using the SQuAD2.0 dataset. The model is case-sensitive, meaning it makes a difference between WORD and word. The model is evaluated on the SQuAD2.0. The model achieved 79.87% exact score and 82.91% f1 score.

### 5.1.2 distilbert-base-cased-distilled-squad [19]

The DistilBERT model is a Transformer model trained by distilling Bert base model. Distillation refers to the technique of compressing a large model called the teacher into a smaller model called the student. The distillation of Bert leads to a smaller model that keeps a lot of the similarities with the original model while being lighter, faster and smaller. The Distilbert has 40% less parameters than Bert-base-uncased, can run 60% faster while preserving more than 95% of the performance of the than Bert-base-uncased. The Distilbert-base-cased-distilled-squad is pre-trained on SQuAD v1.1 and reaches 86.9 F1 score on the SQuAD v1.1 dev set.

### 5.1.3 deepset/minilm-uncased-squad2

Deepset/minilm-uncased-squad2 is also a distilled transformer model pre-trained on the SQuAD 2.0 dataset. It is based on the microsoft/MiniLM-L12-H384-uncased [60] model and achieves 76.1% exact score and 79.49% f1 score. The model is uncased.

## 5.2 Question Answering System Implementation

We train the 3 models presented above using the dataset prepared earlier. But final processing of the dataset is necessary. First, we only extract the needed fields for training: "paper abstract", "predicate label" and "object label". Second, the predicate labels (questions) are not formulated as questions which the pre-trained models expect. A question should start with a question label and end with a question mark. Additionally, a question should be a sentence. Processing the data manually to convert he predicate labels to questions seems to be the optimal way to produce correctly formulated questions, but that is not feasible giving the data size and

the future intended application of the ORKG completion feature. Because of this, we decide on a simpler solution. We choose 3 question labels "what", "how", and "which", and for each of these labels we create a variant of the dataset. Each variant has the same abstracts as the original and the same object labels, but the predicated labels are converted to have a question structure by attaching the question label at the start of the predicate label and a question mark at the end of it. Additionally, we add one more variant with no question label, in which we simply append a question mark at the end of the predicate labels. For instance, given the example of a part of the data set in Figure 5.1 we end up with the 4 variants shown in Figure 5.2, Figure 5.3, Figure 5.4 and Figure 5.5. The 4 variants of the dataset would then be used for training and evaluation. Additionally, for the uncased model we lower-case all text input and for the cased models we keep the data unchanged.

| | A | B | C |
|---|---|---|---|
| 1 | **paper abstract** | **predicate label** | **object label** |
| 2 | The paper presents the SemEval-2012 Shared... | Machine Learning Method | RNN |
| 3 | characterizes a novel approach for the task... | Data formats | XML |
| 4 | The SPAR Ontology Network is a suite of... | Languages | Latin |

Figure 5.1: an example of the dataset with unprocessed predicate labels

| | A | B | C |
|---|---|---|---|
| 1 | **paper abstract** | **predicate label** | **object label** |
| 2 | The paper presents the SemEval-2012 Shared... | Machine Learning Method ? | RNN |
| 3 | characterizes a novel approach for the task... | Data formats ? | XML |
| 4 | The SPAR Ontology Network is a suite of... | Languages ? | Latin |

Figure 5.2: an example of the variant of the dataset without any question label

| | A | B | C |
|---|---|---|---|
| 1 | **paper abstract** | **predicate label** | **object label** |
| 2 | The paper presents the SemEval-2012 Shared... | what Machine Learning Method ? | RNN |
| 3 | characterizes a novel approach for the task... | what Data formats ? | XML |
| 4 | The SPAR Ontology Network is a suite of... | what Languages ? | Latin |

Figure 5.3: an example of the variant of the dataset with the "what" question label

| | A | B | C |
|---|---|---|---|
| 1 | **paper abstract** | **predicate label** | **object label** |
| 2 | The paper presents the SemEval-2012 Shared... | how Machine Learning Method ? | RNN |
| 3 | characterizes a novel approach for the task... | how Data formats ? | XML |
| 4 | The SPAR Ontology Network is a suite of... | how Languages ? | Latin |

Figure 5.4: an example of the variant of the dataset with the "how" question label

| | A | B | C |
|---|---|---|---|
| 1 | **paper abstract** | **predicate label** | **object label** |
| 2 | The paper presents the SemEval-2012 Shared... | which Machine Learning Method ? | RNN |
| 3 | characterizes a novel approach for the task... | which Data formats ? | XML |
| 4 | The SPAR Ontology Network is a suite of... | which Languages ? | Latin |
| 5 | | | |

Figure 5.5: an example of the variant of the dataset with the "which" question label

Similar to the Squad dataset, we supplement the the the training data with the index of the start and the index of the end of the expected answers in the abstracts. With that, each data point in the dataset is a (abstract, predicate label, object label, answer start index, answer end index) tuple. Following is an example from the "what" variant of the dataset.

- abstract: ". Changes in disturbance due to fire regime in southwestern Pinus ponderosa forests over the last century have led to dense forests that are threatened by widespread fire. It has been shown in other studies that a pulse of native, early-seral opportunistic species typically follow such disturbance events. With the growing importance of exotic plants in local flora, however, these exotics often fill this opportunistic role in recovery. We report the effects of fire severity on exotic plant species following three widespread fires of 1996 in northern Arizona P. ponderosa forests. Species richness and abundance of all vascular plant species, including exotics, were higher in burned than nearby unburned areas. Exotic species were far more important, in terms of cover, where fire severity was highest. Species present after wildfires include those of the pre-disturbed forest and new species that could not be predicted from above-ground flora of nearby unburned forests."

- predicate label: "what Type of disturbance ?"

- object label: "Fire"

- answer start index: 32.0

- answer end index: 36.0

**Hyper-parameters**. For the Hyper-parameters we choose 4 epochs, 0.01 weight

decay, a batch size of 8 for training and evaluation, and we train on 2 learning rates: 0.0001 and 0.00005. The learning rates were chosen based on the Hyper-parameters of the vanilla models and from the results of initial experimentation where we trained on the 5 learning rates: 0.01, 0.00002, 0.00003, 0.0001 and 0.00005, with the models achieving the highest results on the last two. Also, from experimentation the models seem to start over-fitting starting from the third epoch, that is the reason for choosing no more than 4 epochs. The hyper-parameters are the same for all the models. And since we have 3 models, and we train each model on each dataset variant and on each learning rate, we end up with a total of 3 models * 4 datasets * 2 learning rates = 24 experiments.

**training and evaluation script**. Similar the data preparation script, the training and evaluation part of the Question Answering System is written in the python language. We especially make use of the Transformers [26] python package. This package provides APIs and tools to easily download and train pre-trained models.

As shown in the sequence diagram of the training script in Figure 5.6 the main.py module is the entry point for the script, where for each of the 24 experiments we initialize a training class. During the initializing of this class, first the vanilla model is loaded and then the training and evaluation datasets are loaded. Then the data is tokenized. Tokenization refers to the process of splitting a string into tokens (words) that then can be mapped to integers that the transformer model can work with. After the initializing phase the training of the model Begins, and after the training is done only the best model over the 4 epochs is saved and then loaded and used for the evaluation phase. Further information on the evaluation metrics and results in the next chapter.

The training script can be found in the public repository `https://github.com/as18cia/thesis_work` with detailed explanation on how to run it.

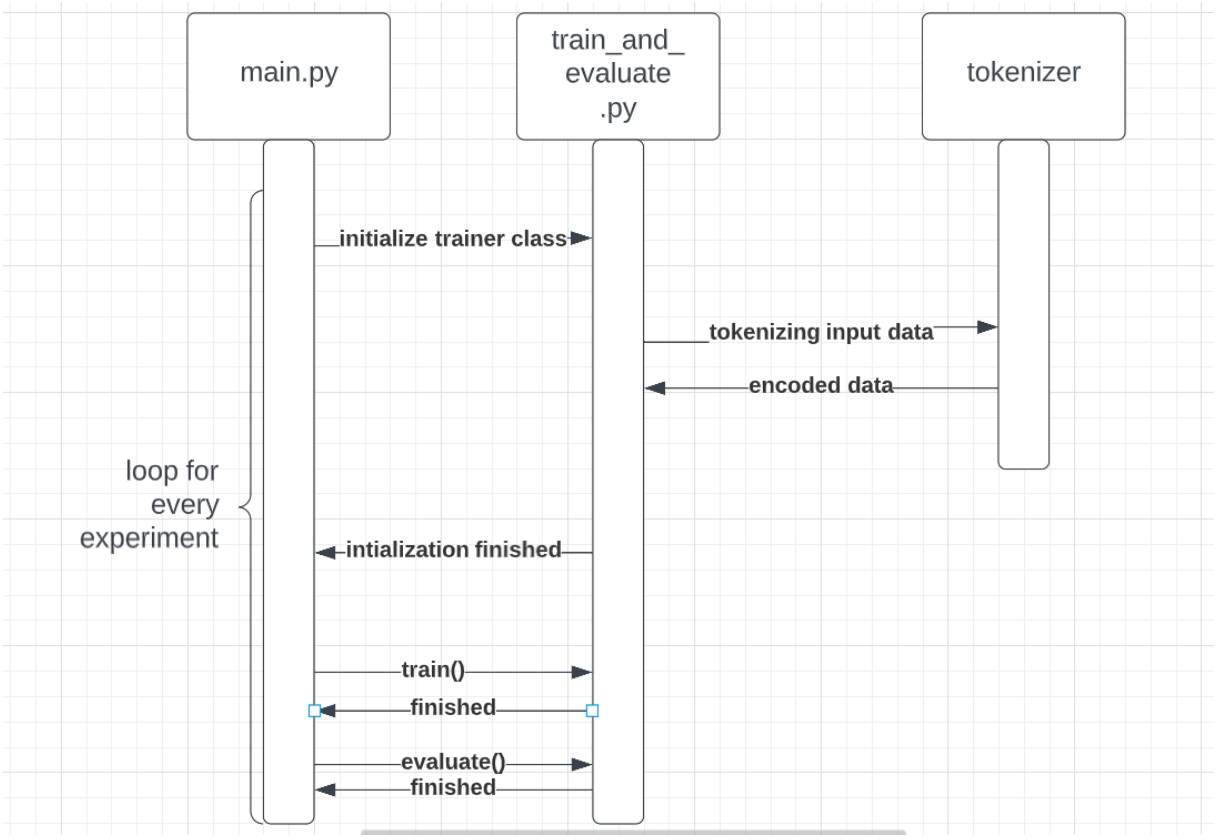



Figure 5.6: A sequence diagram showing an overview of the process of fine-tuning the models (as implemented in the training script)

# Chapter 6

# Evaluation

In this section, we define the metrics used to evaluate the models and then present and discuss the vanilla models and the fine-tuned models evaluation results.

## 6.1 Experimental Setup

**Hyper-parameter Tuning**. We chose to train the models in 4 epochs based on the hyper-parameters of the vanilla models, and also because initial experimentation confirms that after the 2 epoch the models start over-fitting. Based also on the hyper-parameters of the vanilla models, we opted to go with 0.0001 and 0.00005 for the learning rates. Here are the rest of the parameters.

- train_batch_size= 8
- eval_batch_size= 8
- weight_decay= 0.01

At the end of the training phase of each experiment, only the best model is saved and used in the evaluation phase.

**Evaluation Metrics**. 4 metrics are used to evaluate the vanilla and the fine-tuned models: accuracy-exact-match, accuracy-inexact-match, F1-exact-match and F1-inexact-match. The 4 metrics are defined in Equation 2, Equation 3, Equation 8 and Equation 9.

The accuracy-exact-match of a model on an Evaluation data set is the percentage of how many of the questions were answered with a string that is the exact - after initial processing - match of the expected answer. The mentioned processing of both expected and predicted answers entails trimming the trailing and leading white spaces, converting all answers to lower case, and removing any of the following characters ".", ",", ";", ":", "-", ")", "(", "_" and "+" if they are at the end of the answers, which was sometimes observed in the prediction results of the fine-tuned models. Formally, the accuracy-exact-match is defined as follows.

$$accuracy - exact - match = \frac{\text{number of exactly correct answers}}{\text{total number of questions}} \cdot 100 \tag{2}$$

The accuracy-inexact-match is similar to the accuracy-exact-match with the only difference being that a returned answer is also considered correct if the expected answer is contained within it. For example, "the city of Berlin" and "Berlin" would both be correct predicted answers to the expected answer of "Berlin". This means that scores for this metric can only be equal or higher than the accuracy-exact scores. Both the returned and expected answers are pre-processed the same way as the accuracy-exact-match answers. This metric is used because from observing the behavior of the models, multiple correct answers can exist for the same question, but some contain additional tokens. The formal definition is.

$$accuracy - inexact - match = \frac{\text{number of correct answers}}{\text{total number of questions}} \cdot 100 \tag{3}$$

For the F1-exact and F1-inexact we define the following equations.

$$recall - exact = \frac{\text{number of tokens in exactly correct answers}}{\text{number of token is the expected answers}} \tag{4}$$

$$recall - inexact = \frac{\text{number of tokens in correct answers}}{\text{number of token is the expected answers}} \tag{5}$$

$$precision - exact = \frac{\text{number of tokens in exactly correct answers}}{\text{number of token is the returned answers}} \tag{6}$$

$$precision - inexact = \frac{\text{number of tokens in correct answers}}{\text{number of token is the returned answers}} \tag{7}$$

With these definitions, we define the F1-exact and F1-inexact as follows.

$$F1 - exact = \frac{2 \cdot recall - exact \cdot precision - exact}{recall - exact + precision - exact} \tag{8}$$

$$F1 - exact = \frac{2 \cdot recall - inexact \cdot precision - inexact}{recall - inexact + precision - inexact} \tag{9}$$

Before the evaluation of the trained models, the vanilla models need to be evaluated using the 4 metrics defined above. This will serve as a benchmark against which we can measure how much uplift is generated by the training on the ORKG dataset. Similarly to the training, the evaluation is conducted on the 4 variants of the dataset. The results are shown and discussed in the next section. The 4 datasets are the same that will be used to evaluate the fine-tuned models later on.

## 6.2 Results and Discussion

### 6.2.1 Vanilla Model Evaluations

As shown in Table 6.1, the accuracy-exact are all below the 10% mark, which is very low considering Roberta and Minilim models have an Accuracy of above 75% on the SQuAD2.0 dataset, this might indicate that the ORKG dataset is significantly different from the SQuAD set, since the data in the ORKG is of scientific nature. In addition to that, the “which” the “what” datasets have a higher accuracy when compared to the other two.

The accuracy-inexact presented in Table 6.2 is higher than the accuracy-exact as to be expected while not exceeding the 21% mark. Following the same pattern as the accuracy scores, the containment scores are higher for the “which” and “what”

| question | distilbert-base-cased -distilled-squad | roberta- base-squad2 | minilm-unc ased-squad2 | *row avg* |
|---|---|---|---|---|
| none | 2.8 | 2.5 | 3.5 | 2.9 |
| what | 6.8 | 4.3 | 5.1 | 5.4 |
| how | 3.6 | 2.2 | 4.0 | 3.3 |
| which | 7.4 | 4.5 | 5.5 | 5.8 |
| *column avg* | 5.2 | 3.4 | 4.5 | - |

Table 6.1: The accuracy-exact results of the 3 vanilla models with the 4 evaluation data sets, in percent (%).

| question | distilbert-base-cased -distilled-squad | roberta- base-squad2 | minilm-unc ased-squad2 | *row avg* |
|---|---|---|---|---|
| none | 12.6 | 14.8 | 16.0 | 14.5 |
| what | 21.0 | 17.0 | 16.5 | 18.2 |
| how | 16.2 | 16.5 | 16.3 | 16.3 |
| which | 20.3 | 18.2 | 18.0 | 18.8 |
| *column avg* | 17.5 | 16.6 | 16.7 | - |

Table 6.2: The accuracy-inexact results of the 3 vanilla models with the 4 evaluation datasets, in percent (%).

| question | distilbert-base-cased -distilled-squad | roberta- base-squad2 | minilm-unc ased-squad2 | *row avg* |
|---|---|---|---|---|
| none | 2.1 | 4.0 | 3.6 | 3.2 |
| what | 8.0 | 5.1 | 6.3 | 6.5 |
| how | 5.0 | 3.0 | 4.6 | 4.2 |
| which | 8.5 | 5.5 | 5.9 | 6.6 |
| *column avg* | 5.9 | 4.4 | 5.1 | - |

Table 6.3: The F1_exact score results of the 3 vanilla models with the 4 evaluation datasets in percent (%).

datasets in comparison to the 2 others, and for these 2 sets the Distillbert model scores are decently higher than the other models.

| question | distilbert-base-cased -distilled-squad | roberta- base-squad2 | minilm-unc ased-squad2 | *row avg* |
|---|---|---|---|---|
| none | 12.0 | 20.3 | 21.3 | 17.9 |
| what | 24.0 | 22.8 | 24.4 | 23.7 |
| how | 20.2 | 23.7 | 22.7 | 22.2 |
| which | 22.0 | 25.1 | 23.0 | 23.4 |
| *column avg* | 19.6 | 23.0 | 22.9 | - |

Table 6.4: The F1-inexact results of the 3 vanilla models with the 4 evaluation datasets, in percent (%).

The values for the F1-exact in Table 6.3 are following exactly the same pattern as the accuracy-exact. This also goes for the F1-inexact values in Table 6.4, except for the fact that the Distillbert model is not better here.

## 6.3 Fine-tuned Models Evaluations

In this section, the 4 main metrics results of the fine-tuned models using the ORKG dataset are presented and compared with the benchmark values. We then present the results per object label category. And finally, we show some statistics regarding the number of tokens in the answers returned by the models, as this affects the already mentioned metrics and also the quality of the predicted answers themselves.

### 6.3.1 Dataset Level Results

The accuracy-exact scores of all models shown in Table 6.5 have increased significantly after training on the ORKG dataset. Moreover, looking at the average values over each dataset, it appears that the pattern of some datasets having better results has decreased after training. And unlike the vanilla models results, the Distillbert model has lower scores than the other two.

The accuracy-inexact scores in Table 6.6 have increased significantly after training. The Roberta model score on the "how" dataset, for instance, has jumped from 16% to 51%. And similarly to the accuracy-exact the difference between the avg scores for each data set is minimal

| question | distilbert-base-cased -distilled-squad | roberta- base-squad2 | minilm-unc ased-squad2 | *row avg* |
|---|---|---|---|---|
| none | 32.9 | 37.5 | 35.4 | 35.3 |
| what | 32.5 | 35.9 | 36.1 | 34.8 |
| how | 34.6 | 36.4 | 33.8 | 34.9 |
| which | 33.1 | 36.6 | 36.8 | 35.5 |
| *column avg* | 33.3 | 36.6 | 35.5 | - |

Table 6.5: The best accuracy-exact results of the 3 fine-tuned models, in percent (%).

| question | distilbert-base-cased -distilled-squad | roberta- base-squad2 | minilm-unc ased-squad2 | *row avg* |
|---|---|---|---|---|
| none | 44.0 | 50.6 | 48.3 | 47.6 |
| what | 44.3 | 49.5 | 46.0 | 46.6 |
| how | 44.1 | 51.2 | 47.0 | 47.4 |
| which | 43.8 | 48.6 | 47.9 | 46.8 |
| *column avg* | 44.1 | 50.0 | 47.3 | - |

Table 6.6: The accuracy-inexact results of the 3 fine-tuned models, in percent (%).

| question | distilbert-base-cased -distilled-squad | roberta- base-squad2 | minilm-unc ased-squad2 | *row avg* |
|---|---|---|---|---|
| none | 26.8 | 23.0 | 18.0 | 22.6 |
| what | 17.7 | 21.2 | 19.8 | 19.6 |
| how | 21.4 | 20.9 | 22.3 | 21.5 |
| which | 22.9 | 24.0 | 25.9 | 24.3 |
| *column avg* | 22.2 | 22.3 | 21.5 | - |

Table 6.7: The F1-exact results of the 3 fine-tuned models, in percent (%).

All in all, the best avg results are achieved with the Roberta model for all 4 metrics

| question | distilbert-base-cased -distilled-squad | roberta-base-squad2 | minilm-uncased-squad2 | *row avg* |
|---|---|---|---|---|
| none | 34.2 | 32.3 | 26.1 | 30.9 |
| what | 25.3 | 31.0 | 26.2 | 27.5 |
| how | 28.0 | 30.4 | 28.5 | 29.0 |
| which | 31.6 | 35.9 | 36.4 | 34.6 |
| *column avg* | 29.8 | 32.4 | 29.3 | - |

Table 6.8: The F1-inexact results of the 3 fine-tuned models, in percent (%).

regardless of the learning rate, the Minilim model comes second with the Distillbert model being the worst model contrary to its vanilla version which was the best of the 3.

### 6.3.2 Category Level Results

We chose to run the object label category using the accuracy-exact and the accuracy-inexact metrics, for this we only consider the best models and dataset variant combination. The best model for the 2 metrics is Roberta with 37.5% accuracy-exact using the "no question label" set and with 51.2% accuracy-inexact using the "how" set.

From Table 6.9 and Table 6.10 we can see that the accuracy scores for sentences, count/measurement and noun phrases is low with 0%, 10.7% and 27.3% respectively which might indicate that the model struggles with the prediction of answers with high number of token. On the other hand, categories that have typically lower number of tokens have a high accuracy.

| object label category | percent in the evaluation dataset | accuracy-exact of predictions (%) |
|---|---|---|
| sentence | 3 | 0.0 |
| count/measurement | 3 | 10.7 |
| number | 0.5 | 16.7 |
| noun phrase | 30 | 27.3 |
| year/date | 1 | 28.6 |
| acronym | 9 | 42.9 |
| research problem | 14 | 44.8 |
| noun | 30 | 44.9 |
| adj | 3 | 46.4 |
| location | 6 | 57.4 |
| url | 0.25 | 66.7 |

Table 6.9: The accuracy-exact per object label category using the deepset/roberta-base-squad2 that was fine-tuned on the "no question label" dataset, with an overall accuracy-exact of 37.5%

| object label category | number of rows | accuracy-inexact of predictions (%) |
|---|---|---|
| sentence | 3 | 0.0 |
| count/measurement | 3 | 35.7 |
| number | 0.5 | 39.9 |
| noun phrase | 30 | 50 |
| year/date | 1 | 50 |
| acronym | 9 | 56.6 |
| research problem | 14 | 57.4 |
| noun | 30 | 64.8 |
| adj | 3 | 66.7 |
| location | 6 | 68.9 |
| url | 0.25 | 71.4 |

Table 6.10: The accuracy-inexact per object label category using the deepset/roberta-base-squad2 that was fine-tuned on the "how" dataset, with an overall accuracy-inexact of 51.2%

### 6.3.3 Additional results

| model name | avg number of tokens in expected answers | avg number of tokens in predicted answers |
|---|---|---|
| distilbert-base-cased-distilled-squad | 2.4 | 9.5 |
| roberta-base-squad2 | 2.4 | 10.6 |
| minilm-uncased-squad2 | 2.4 | 18.8 |

Table 6.11: avg number of tokens for expected and predicted answers for the vanilla models

| model name | avg number of tokens in expected answers | avg number of tokens in predicted answers |
|---|---|---|
| distilbert-base-cased-distilled-squad | 2.4 | 6.9 |
| roberta-base-squad2 | 2.4 | 7.0 |
| minilm-uncased-squad2 | 2.4 | 7.1 |

Table 6.12: avg number of tokens for expected and predicted answers for the fine-tuned models

As shown in Table 6.11 and Table 6.12 the avg number of tokens predicted by the vanilla models is significantly higher than that of the expected answers with the Minilim model being way worse by a big margin. After fine-tuning this value went down noticeably, especially for the Minilim model that went from an avg of 18.8 tokens per answer to 7.1. The results help explain why the accuracy-inexact is higher than the accuracy-exact of the models, since the avg number of returned answers is higher than the expected, which means the models tends to return more tokens in addition to the correct expected tokens.

# Chapter 7

# Conclusions and Future Work

Towards the end goal of the ORKG completion, we explore the performance of transformer models pre-trained on The SQUaD dataset and fine-tuned using a dataset constructed using data extracted from the ORKG and supplemented with paper abstracts from third-party services. The results show significant uplift in comparison to the results of the vanilla models. However, an accuracy-exact score below 37% indicates that there is still work to be done before this completion feature could be implemented in any production setting. This is also true because the models are not trained on questions that don't an answer, this can drop the accuracy of the models even further.

An Area where further improvements could be achieved is the size of the training dataset, which significantly affected by issue of finding the abstracts for the papers in the ORKG. In the case of this work, we had to omit about 40% of the dataset because of this issue. Furthermore, from the object label category evaluation results, we can see that the models seem to struggle with predicting multi-token strings. This especially clear for the case of noun phrase category that form a large percentage of the dataset, while achieving medium to low relative results. The effect of the method used here to convert the predicate labels into questions is still not know. However, since we only have around 853 unique predicate labels in the dataset, doing the conversion manually and then comparing with the 4 variant created automatically might be possible given enough and might be worthwhile.

# Bibliography


[1] *A reintroduction to our Knowledge Graph and knowledge panels*. `https://blog.google/products/search/about-knowledge-graph-and-knoswledge-panels/`. Accessed: 2020-07-16. 2020.

[2] Mehdi Ali et al. "Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[3] Amir Aryani et al. "A Research Graph dataset for connecting research data repositories using RD-Switchboard". In: *Scientific data* 5 (2018), p. 180099.

[4] Sören Auer. *Towards an Open Research Knowledge Graph*. Version 1. Jan. 2018. DOI: `10.5281/zenodo.1157185`.

[5] Sören Auer et al. "Improving access to scientific literature with knowledge graphs". In: *Bibliothek Forschung und Praxis* 44.3 (2020), pp. 516–529.

[6] Jeroen Baas et al. "Scopus as a curated, high-quality bibliometric data source for academic research in quantitative science studies". In: *Quantitative Science Studies* 1.1 (2020), pp. 377–386.

[7] Justin Betteridge et al. "Toward never ending language learning." In: *AAAI spring symposium: Learning by reading and learning to read*. 2009, pp. 1–2.

[8] Caroline Birkle et al. "Web of Science as a data source for research on scientific and scholarly activity". In: *Quantitative Science Studies* 1.1 (2020), pp. 363–376.

[9] Kurt Bollacker et al. "Freebase: a collaboratively created graph database for structuring human knowledge". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008, pp. 1247–1250.

[10] *bookcorpus · Datasets at Hugging Face*. `https://huggingface.co/datasets/bookcorpus`. Accessed: September 23, 2022.

[11] Antoine Bordes et al. "Translating embeddings for modeling multi-relational data". In: *Advances in neural information processing systems* 26 (2013).

[12] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[13] B Barla Cambazoglu et al. "A review of public datasets in question answering research". In: *ACM SIGIR Forum*. Vol. 54. 2. ACM New York, NY, USA. 2021, pp. 1–23.

[14] Jeremy Carroll and Graham Klyne. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. W3C, Feb. 2004.

[15] Chiorrini et al. "Emotion and sentiment analysis of tweets using BERT". In: (2021). URL: http://ceur-ws.org/Vol-2841/DARLI-AP_17.pdf.

[16] *Crossref Unified Resource API*. https://api.crossref.org/. Accessed: September 22, 2022.

[17] *deepset/roberta-base-squad2*. https://huggingface.co/deepset/roberta-base-squad2. Accessed: September 23, 2022.

[18] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[19] *distilbert-base-cased-distilled-squad*. https://huggingface.co/distilbert-base-cased-distilled-squad. Accessed: September 23, 2022.

[20] Christiane Fellbaum. "WordNet". In: *Theory and applications of ontology: computer applications*. Springer, 2010, pp. 231–243.

[21] Martin Fenner and Amir Aryani. "Introducing the PID Graph". In: (2019). DOI: 10.5438/JWVF-8A66. URL: https://blog.datacite.org/introducing-the-pid-graph/.

[22] Luciano Floridi and Massimo Chiriatti. "GPT-3: Its nature, scope, limits, and consequences". In: *Minds and Machines* 30.4 (2020), pp. 681–694.

[23] Suzanne Fricke. "Semantic scholar". In: *Journal of the Medical Library Association: JMLA* 106.1 (2018), p. 145.

[24] Ginny Hendricks et al. "Crossref: The sustainable source of community-owned scholarly metadata". In: *Quantitative Science Studies* 1.1 (2020), pp. 414–427.

[25] Nikos Houssos et al. "OpenAIRE guidelines for CRIS managers: supporting interoperability of open research information through established standards". In: *Procedia Computer Science* 33 (2014), pp. 33–38.

[26] *huggingface/transformers*. https://github.com/huggingface/transformers. Accessed: September 23, 2022.

[27] *Java*. https://www.java.com/en/. Accessed: September 23, 2022.

[28] Guoliang Ji et al. "Knowledge graph embedding via dynamic mapping matrix". In: *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. 2015, pp. 687–696.

[29] Rob Johnson, Anthony Watkinson, and Michael Mabe. "The STM report". In: *An overview of scientific and scholarly publishing. 5th edition October* (2018).

[30] Esther Landhuis. "Scientific literature: information overload". In: *Nature* 535.7612 (2016), pp. 457–458.

[31] Jens Lehmann et al. "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia". In: *Semantic web* 6.2 (2015), pp. 167–195.

[32] Yankai Lin et al. "Learning entity and relation embeddings for knowledge graph completion". In: *Twenty-ninth AAAI conference on artificial intelligence*. 2015.

[33] Pengfei Liu et al. "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing". In: *arXiv preprint arXiv:2107.13586* (2021).

[34] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: (2019). URL: https://arxiv.org/abs/1907.11692.

[35] Paolo Manghi et al. *OpenAIRE Research Graph Dump*. Version 1.0.0-beta. Zenodo, Dec. 2019. DOI: 10.5281/zenodo.3516918. URL: https://doi.org/10.5281/zenodo.3516918.

[36] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. "A three-way model for collective learning on multi-relational data". In: *Icml*. 2011.

[37] Natasha Noy et al. "Industry-scale knowledge graphs: lessons and challenges". In: *Queue* 17.2 (2019), pp. 48–75.

[38] *ORKG REST API Documentation*. http://tibhannover.gitlab.io/orkg/orkg-backend/api-doc/. Accessed: September 22, 2022.

[39] *orkg-pypi*. https://pypi.org/project/orkg/. Accessed: September 22, 2022.

[40] *pandas.DataFrame - pandas 1.5.0 documentation*. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html. Accessed: September 23, 2022.

[41] *Python*. https://www.python.org/. Accessed: September 23, 2022.

[42] Pranav Rajpurkar et al. "Know What You Don't Know: Unanswerable Questions for SQuAD". In: (2018). URL: https://arxiv.org/pdf/1806.03822.pdf.

[43] Pranav Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: (2016). URL: https://arxiv.org/pdf/1606.05250.pdf.

[44] Pranav Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 2383–2392.

[45] Navid Rezaei and Marek Z Reformat. "Super-Prompting: Utilizing Model-Independent Contextual Data to Reduce Data Annotation Required in Visual Commonsense Tasks". In: *arXiv preprint arXiv:2204.11922* (2022).

[46] Navid Rezaei and Marek Z Reformat. "Utilizing Language Models to Expand Vision-Based Commonsense Knowledge Graphs". In: *Symmetry* 14.8 (2022), p. 1715.

[47] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: (2019). URL: https://arxiv.org/abs/1910.01108.

[48] Robin M. Schmidt. "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview". In: (2019). URL: https://arxiv.org/abs/1912.05911.

[49] *Semantic Scholar Academic Graph API*. https://api.semanticscholar.org/. Accessed: September 22, 2022.

[50] *services/metadata.py · master · TIB Hannover / Open Research Knowledge Graph (ORKG) / ORKG Bioassays Semantification · GitLab*. https://gitlab.com/TIBHannover/orkg/orkg-bioassays-semantification/-/blob/master/services/metadata.py. Accessed: September 22, 2022.

[51] *spaCy: Industrial-strength NLP*. https://pypi.org/project/spacy/. Accessed: September 22, 2022.

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

[53] Zhiqing Sun et al. "Rotate: Knowledge graph embedding by relational rotation in complex space". In: *arXiv preprint arXiv:1902.10197* (2019).

[54] Théo Trouillon et al. "Complex embeddings for simple link prediction". In: *International conference on machine learning*. PMLR. 2016, pp. 2071–2080.

[55] Sahar Vahdati et al. "OpenResearch: Collaborative Management of Scholarly Communication Metadata". In: *Knowledge Engineering and Knowledge Management*. Ed. by Eva Blomqvist et al. Cham: Springer International Publishing, 2016, pp. 778–793. ISBN: 978-3-319-49004-5.

[56] Ashish Vaswani et al. "Attention Is All You Need". In: (2017). URL: https://arxiv.org/abs/1706.03762.

[57] Denny Vrandečić and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase". In: *Communications of the ACM* 57.10 (2014), pp. 78–85.

[58] Jingbo Wang et al. "Graph Connections Made By RD-Switchboard Using NCI's Metadata". In: *D-Lib Magazine* 23.1/2 (2017).

[59] Kuansan Wang et al. "Microsoft academic graph: When experts are not enough". In: *Quantitative Science Studies* 1.1 (2020), pp. 396–413.

[60] Wenhui Wang et al. "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers". In: (2020). URL: https://www.microsoft.com/en-us/research/publication/minilm-deep-self-attention-distillation-for-task-agnostic-compression-of-pre-trained-transformers/.

[61] Zhen Wang et al. "Knowledge graph embedding by translating on hyperplanes". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 28. 1. 2014.

[62] Jason Wei et al. "Chain of thought prompting elicits reasoning in large language models". In: *arXiv preprint arXiv:2201.11903* (2022).

[63] Bishan Yang et al. "Embedding entities and relations for learning and inference in knowledge bases". In: *arXiv preprint arXiv:1412.6575* (2014).