



Universität Hamburg  
DER FORSCHUNG | DER LEHRE | DER BILDUNG

## MASTER THESIS

# Study and Creation of Datasets for Comparative Questions Classification

vorgelegt von

Steffen Stahlhacke

MIN-Fakultät

Fachbereich Informatik

Studiengang: Master Informatik

Matrikelnummer: 7093326

Erstgutachter: Prof. Dr. Chris Biemann

Zweitgutachter: Dr. Seid Muhie Yimam

Betreuerin: Dr. Meriem Beloucif



# Abstract

Recent research of question answering systems (QAS) has focused on answering factoid questions rather than on answering comparisons. This thesis proposes two novel question answering datasets for comparative questions. The datasets are created with open-domain data in the English language. The data samples are collected from the community-driven platforms Reddit and Yahoo! Answers. A novel linguistic-based taxonomy for comparative questions has enabled the data mining of 245,000 potential comparative questions from these platforms. The first dataset is built up out of 10,000 human labelled samples for the binary classification of comparative questions. The second dataset contains 4,000 samples that were assigned sequence labels for comparative objects and comparative aspects in a human annotation task. In this thesis, both datasets are utilized in machine-learning experiments to classify comparative questions and to extract the comparative objects and aspects. In the classification task, a neural machine-learning model with ALBERT embeddings reaches an F1 score of 0.876, 0.022 points below the human performance. The identification of comparative objects and aspects, with a neural machine-learning model that uses BERT embeddings, reaches the human performance with an F1 score of 0.8054. Furthermore, both models are combined in a web application for the classification of comparative questions, in order to provide a simple way for users and QAS to receive a comparative classification analysis.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Related work</b>	<b>7</b>
2.1	Question answering . . . . .	7
2.2	Question taxonomies . . . . .	13
2.3	Related work on comparatives . . . . .	17
<b>3</b>	<b>Creation of comparative question datasets</b>	<b>23</b>
3.1	Linguistic background on comparative questions . . . . .	23
3.1.1	Rule-based comparisons . . . . .	23
3.1.2	Phrase-based comparisons . . . . .	28
3.1.3	Generation rules of questions . . . . .	29
3.1.4	Definition of a comparative question . . . . .	32
3.2	Linguistic-based taxonomy for comparative questions . . . . .	36
3.3	Data mining . . . . .	39
3.3.1	Evaluation of data sources . . . . .	39
3.3.2	Gathering and pre-processing of source data . . . . .	53
3.3.3	Filtering for comparative questions . . . . .	57
3.3.4	Data mining results . . . . .	63
3.4	Human annotation task 1: Classification . . . . .	68
3.4.1	Task preparations . . . . .	69
3.4.2	Pilot . . . . .	73
3.4.3	Main phase . . . . .	76

## CONTENTS

---

3.4.4	Dataset statistics . . . . .	79
3.5	Human annotation task 2: Sequence tagging . . . . .	83
3.5.1	Task preparations . . . . .	83
3.5.2	Pilot . . . . .	89
3.5.3	Main phase . . . . .	92
3.5.4	Dataset statistics . . . . .	94
<b>4</b>	<b>Experiments with comparative question datasets</b>	<b>97</b>
4.1	Classification of comparative questions . . . . .	97
4.1.1	Experimental setup . . . . .	98
4.1.2	Training and results . . . . .	104
4.2	Extracting comparative objects and aspects . . . . .	109
4.2.1	Experimental setup . . . . .	109
4.2.2	Training and results . . . . .	112
4.3	Discussion of results . . . . .	116
4.4	Comparative classification web app . . . . .	119
<b>5</b>	<b>Conclusion</b>	<b>123</b>
<b>6</b>	<b>Future work</b>	<b>125</b>
	<b>List of figures</b>	<b>X</b>
	<b>List of tables</b>	<b>XIII</b>
	<b>List of listings</b>	<b>XVII</b>
	<b>List of abbreviations</b>	<b>XIX</b>
	<b>Appendices</b>	<b>XXIII</b>

# Chapter 1

## Introduction

Nowadays, we are surrounded by data and information. Especially the world wide web seems to be an unlimited resource of information. The multinational technology company Cisco Systems estimated that the world’s collective internet use entered the Zettabyte Era in 2016 with over 1.2 zettabyte traffic per year. To get a better understanding of this number, Cisco tried to quantify the number with this analogy: “If each terabyte in a zettabyte were a kilometre, it would be equivalent to 1,300 round trips to the moon and back (768,800 kilometres)” [1]. The company also estimated that the globally averaged internet user will generate 84.6 gigabytes of traffic per month in 2022. The estimation for the year 2017 was 28.8 gigabytes per month, which is an increase of 194% [2]. It is estimated that there are more than 1.5 billion websites<sup>1</sup> today (see Figure 1.1) [3]. For users, it can be exhausting to find the information they desire within these millions of documents. Using a search engine is one way to acquire the needed information.

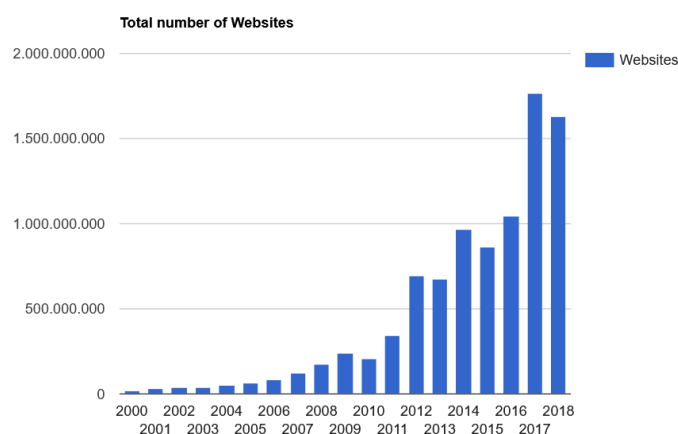


Figure 1.1: The figure shows the total number of existing websites per year. Since 2016, the number has stabilized above the 1 billion mark [3].

<sup>1</sup>Website in this context means a unique hostname, which can be resolved into an IP address [3]

## CHAPTER 1. INTRODUCTION

Search engines like Google traditionally try to provide the user with links to webpages that contain the desired information. In 2016, Google handled more than 2 trillion global searches per year [4]. Even though Google is not releasing statistics on the search intent of users, for the United States it was estimated that 8% of all searches are expressed as a question searching with the intent to retrieve information [5]. Since 2012, Google shows knowledge panels (see Figure 1.2) to provide an overview of information and to directly answer questions [6]. The knowledge panels are based on Google’s large-scale knowledge base (KB), covering over 70 billion facts gathered from various sources and providing information on entities like places, people and companies [7]. Today, the knowledge panels are shown in 37% of the search queries made on Google [8].

The image shows a Google search interface for the query "What football team is in San Francisco?". The search results page features a prominent knowledge panel for the San Francisco 49ers. The panel is divided into several sections: a top header with the team name and logo, a "GAMES" section with a table of recent matches, a "NEWS" section with a "Final" result, a "STANDINGS" section with a table of current standings, and a "PLAYERS" section with a list of key players. Below the knowledge panel, there is a "People also ask" section with four related questions. The right side of the page shows a detailed view of the San Francisco 49ers, including their website, a description, their NFL championships, head coach, stadium, and upcoming match information.

Team	Score	Opponent	Score	Final
49ers	22	Falcons	29	Final 15/12
49ers	34	Rams	31	Final 22/12
Seahawks	21	49ers	26	Final 30/12
49ers	27	Vikings	10	Final 11/01
49ers	37	Packers	20	Final 20/01
Chiefs	31	49ers	20	Final Mon, 03/02

Player	Position	Number
Jimmy Garoppolo	Quarterback	10
George Kittle	Tight end	85
Nick Bosa	Defensive end	97

**People also ask**

- What NBA team is in San Francisco?
- What team is moving to SF?
- Does San Francisco have an NHL team?
- Who owns the San Francisco 49ers football team?

Figure 1.2: The figure shows a knowledge panel, which provides the answer to a question asked on Google (right-side). Google uses knowledge cards (top) to display information related to the searched entity. For some search queries, Google also shows a collection of questions similar to the one asked by the user (bottom).



In 2016, 85% of the costumers started their product research online for major purchases of \$500 or more. 38% of the customers checked online reviews before buying [9]. There are countless websites that offer their users comparisons as a service. Product-comparison websites like CNET<sup>2</sup> (82nd in the US web-traffic ranking [10]) or product reviews and comparisons on YouTube are a valid source for users to get information in order to decide which product they should buy. Furthermore, websites with a wider range of comparisons, like Check24<sup>3</sup>, offer their users a good way of comparing products, contracts or memberships. These websites often focus on comparing the prices and some features of the products.

On websites like Yahoo! Answers<sup>4</sup>, Quora<sup>5</sup> or Reddit<sup>6</sup> countless comparative questions can be found. On these websites people are not restricted to a predefined topic or a specific type of answer. These websites follow a community-driven approach by letting the users both ask and answer questions. The provided answers for a question are then rated to find the best fitting answer. This way, more complicated and opinion-based questions can be answered. Especially Quora prefers their users to answer questions under their real name. The people answering are supposed to have experience or expertise regarding the question to provide factually based answers [11].

---

#	Example
1	What is better Xbox One or PS4?
2	What is the difference between a cappuccino and a latte?
3	Why is natural sugar better than added sugar? What's different in the chemical construction?
4	Should i buy or rent in California?
5	Why do children learn languages faster than adults?
6	Can you please explain the difference between ham and shortwave, like if i wanted to set my parents (who live in ca fire country) up with a simple uncomplicated means of 911 communication in case cell towers go down?

---

Table 1.1: Examples of comparative questions from Yahoo and Reddit.

The examples of comparative questions, taken from Yahoo and Reddit, illustrate the variety of questions and topics (see Table 1.1). To comprehend a comparative question, it is

<sup>2</sup><https://www.cnet.com>

<sup>3</sup><https://www.check24.de>

<sup>4</sup><https://answers.yahoo.com/>

<sup>5</sup><https://www.quora.com>

<sup>6</sup><https://www.reddit.com/>

necessary to understand which objects or situations are compared. In the first example, the two gaming consoles “Xbox One” and “PS4” are compared. These linguistic objects will be referred to as the objects or the entities of a comparison, while keeping in mind that “object” can refer to feelings, situations, actions, people or any other non-material thing. Some comparative questions aim to compare two objects with regard to a certain aspect or feature of the objects. The fifth example demonstrates this. The two objects of comparison are “children” and “adults”. They are not compared in an overall manner, but instead, the question is asking for a comparison regarding the ability to learn languages. This will be referred to as the aspect of a comparison.

To enable users to compare these open-domain entities, the University of Hamburg takes part in the research project called Argumentation in Comparative Question Answering (ACQuA) [12]. The goal of the project is to understand a user’s comparative question, find supporting facts for an answer and then provide a fact-supported, comprehensible answer in a natural language. To achieve this goal, the research team developed an automatic Comparative Argumentative Machine (CAM) for question answering (QA) in an open-domain setting [13]. The CAM interface (see Figure 1.3) promotes users to input two objects and multiple aspects of comparison. The answer is presented in a horizontal bar chart showing the score that allows the users to get a general impression of the comparison. To provide a comprehensible answer, the Comparative Argumentative Machine also displays the supporting sentences for both comparative objects. The data used by the system is a preprocessed version of the Common Crawl Text Corpus<sup>8</sup>, which crawled from the world wide web. Sentences containing both objects are retrieved from the corpus and then classified into the categories better, equal, worse or none. The categories present the relation of the first object to the second object. The sentence “Xbox One is better then PS4” is an example for the category ”better”. The classified sentences are ranked and the score is calculated.

The Comparative Argumentative Machine lacks a natural way for users to input a comparison and get a summed-up answer in natural language, such as the one a human being would give (see Figure 1.3). Ideally, the system should work similar to a chatbot or a spoken-dialogue system (e.g. Amazon Alexa). The user’s input to the interface should be a comparative question expressed in natural language, similar to this example:

**Question:** I am thinking of buying a gaming console and have a budget of 300\$. Which one is better suited for me, Xbox One or PS4?

---

<sup>8</sup><https://commoncrawl.org/>

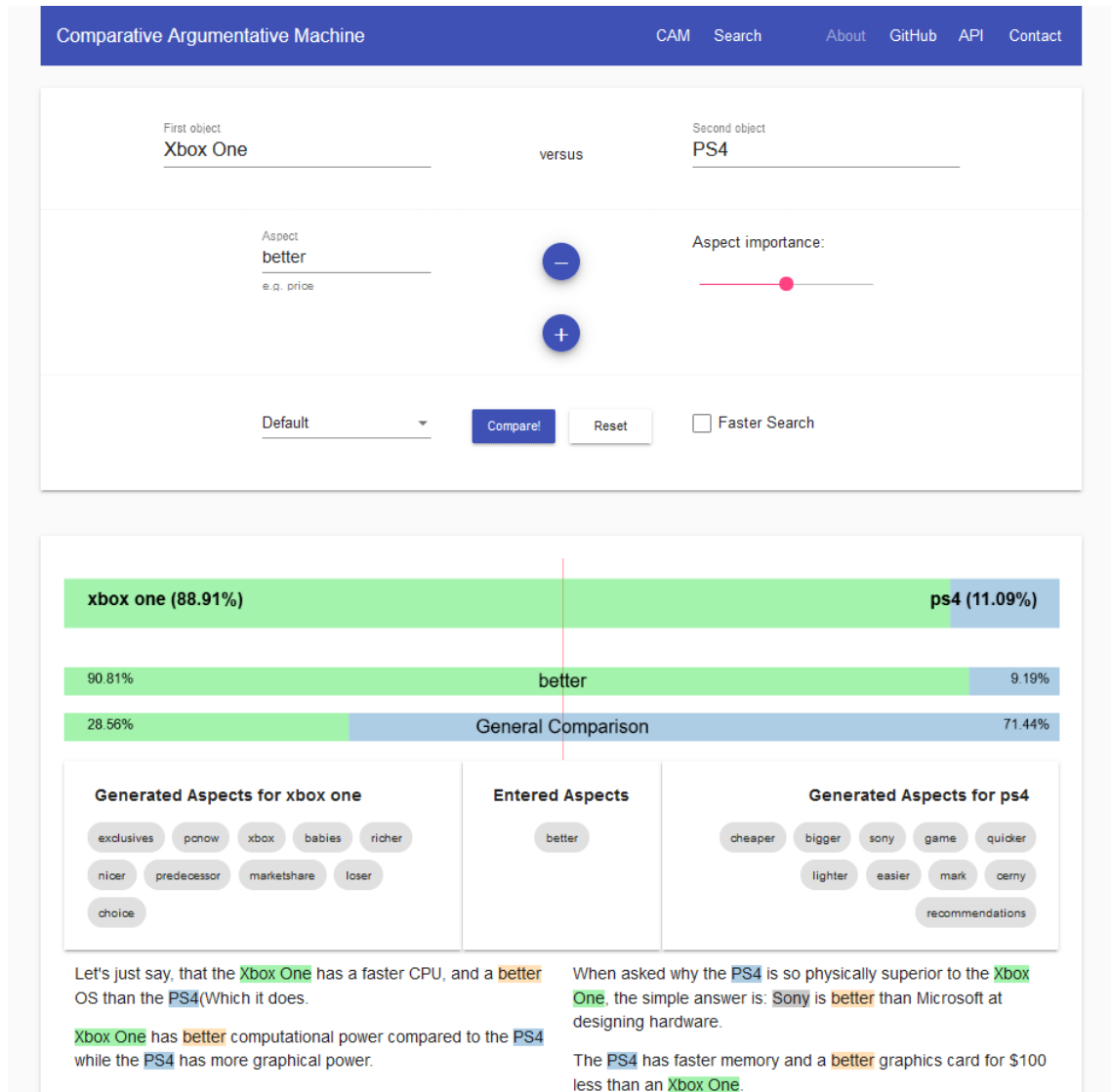


Figure 1.3: The figure shows the Comparative Argumentative Machine (CAM)<sup>7</sup> developed by the team of the ACQuA project. The interface provides the users with an input mask for two comparative objects and multiple comparative aspects (top). The system displays score bars, to present a general impression of the comparison, and supporting sentences for both comparative objects (bottom).

The system should answer in a natural language and sum up the key-supporting facts for the answer:

**Answer:** An Xbox One is better suited for you. Even though the average price is a bit higher, the Xbox One is equipped with the better hardware.

The primary purpose of this thesis is to comprehend the user's natural-language questions.

This means, to identify questions that contain comparisons and to analyse which entities are being compared. In case the entities are compared regarding a certain aspect, these aspects also have to be obtained. The goal is to create an open-domain corpus containing comparative questions for the English language with a binary classification of comparative and non-comparative questions. Furthermore, the aim of this work is to create a second dataset, which provides labels for comparative objects and aspects. The intention is to evaluate various classifiers on their performance of comparative-question classification. The second dataset will be utilized to extract comparative objects and aspects that can be used in question answering systems (QAS). One example of QAS is the Comparative Argumentative Machine as a part of the ACQuA project.

The thesis is composed of five additional chapters. The second chapter provides an insight into the background information and the related work needed for this work. In the third chapter, a definition of comparisons is presented and the creation of the human-annotated dataset for comparative questions is explained in detail. The fourth chapter focuses on working with the corpus to classify comparatives and to extract the entities of the comparison. The last two chapters conclude the work and present an overview about possible future research.

## Chapter 2

# Related work

In this chapter, background knowledge about question answering systems (QAS) and Question Classification is presented. The first section offers a general overview about question answering systems and their latest research. In the second section, a more detailed look on how questions can be classified is provided and it is explained why this is relevant for QAS. The third section studies the related work on comparative sentences and questions.

### 2.1 Question answering

Question answering (QA) was one of the early topics for computer systems. BASEBALL, one of the very first question answering systems, was proposed by Green et al. [14] in 1961. This system was created to answer questions about baseball games from one season. One of the typical questions could be “Who did the Red Sox lose to on July 5?”. The questions were provided in the English language and read into the system by punched cards. The system was limited in terms of the input structures of the questions. For example, logic connectors like “and” or “not” were forbidden. Also questions were limited to a single clause. Asking for sequential facts, such as “Did the Red Sox ever win six games in a row?”, was not allowed. The input data was syntactically analysed and the question was brought into a canonical format, which related to a predefined dictionary of input to value pairs. If “Red Sox” or “Who” was the input, the canonical predefined relation would be “Team”. For the following look-up, the data of the baseball season was hierarchically stored. For each game, the day, month, place, teams and scores were stored. The final response was not given in natural language, but as a list or a simple Yes/No.

A step in the direction of understanding a more natural English language was made by the LUNAR Prototype. It was developed by Woods [15] for scientists of the NASA Apollo

missions. The goal was to provide the NASA geologists with a system to access and evaluate chemical analysis on lunar rock and soil in a natural way. The input questions were limited to refer to the geological database and to not contain comparisons. Input text was analysed syntactically by the use of a grammar that replicates “a large subset of English” [15]. Furthermore, special rules and a dictionary of 3,500 words fitting the geological domain were utilized. After parsing the input, a semantic interpretation of the sentence was formed and converted into a formal query language. The data was stored in a database table and could be accessed by the system through the formal queries. The output was shown to the users as listings or calculations.

Both systems are good examples of rule-based question answering systems in very specialized closed-domain settings. The input queries to the systems were limited in terms of language syntax or vocabulary. In some cases, even simple paraphrasing of a sentence was impossible to overcome for the systems [15]. In order to build and maintain the data structures, experts were needed. This implied high costs and soon outdated or irrelevant data. Furthermore, none of the systems generated natural language answers for the users.

With the following research, there was a shift from closed and very limited domains to open-domain systems that rely on knowledge base (KB) representations of the data. Katz et al. proposed the “SynTactic Analysis using Reversible Transformations” (START) [16] system in 1988. START made a step towards understanding natural language in an open-domain setting. The system was able to build and extend its knowledge base with information, using English natural language texts as input. The START system was created to comprehend and generate English sentences with the same grammar. Lexicons and lexical classes helped to overcome key factors in understanding more complex parts of the natural language, such as lexical disambiguates. In their following work, Katz et al. used the Word Wide Web as an information resource for the START system [17]. In 2002, Katz et al. proposed Omnibase [18], a knowledge base for their START system to use semi-structured resources from the web, such as the Internet Movie Database (IMDB). The Omnibase knowledge base was built with an object-property-value data model, generated by the START system. Each sentence was interpreted as a relation of an object and a property associated to a fitting answer value. For example, the question “Who wrote the music for Star Wars?” is asking for the object “Star Wars” with the property “Composer” and the associated value “John Williams”. Katz et al. were aware that not all types of questions could be expressed by means of this kind of model [18]. They concluded that robust systems that are able to understand natural language were not expected to exist anytime soon, due to the fact that subjects like inter-sentence references, paraphrasing, summarization and common-sense implication were still considered a problem [17].

Today, one of the largest and broadest knowledge systems is Wolfram|Alpha [19]<sup>1</sup>. It provides information for different kinds of questions that can be answered with facts. Wolfram|Alpha relies on external and computational data, of which some is checked by experts. Some types of data, e.g. financial data, is updated constantly. Users can ask the system natural-language questions and get an answer of facts and computations. The answers are structured in a table-like format. Wolfram|Alpha is limited to facts and does not encompass opinions [20].

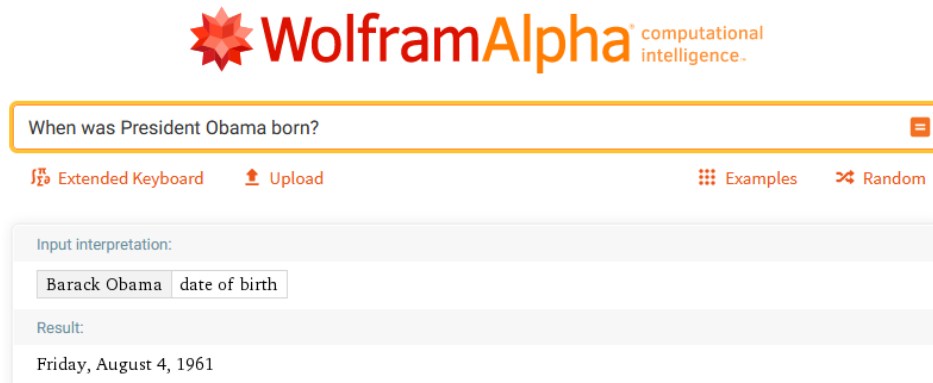


Figure 2.1: The figure shows an example input into Wolfram|Alpha. In this case, the input “President Obama” is interpreted to be “Barack Obama”.

One of the most prominent question answering systems is IBM’s Watson, which competed and won the quiz show Jeopardy! in 2011. Jeopardy! has been broadcasted in the United States of America for more than 50 years. In a special episode, Watson won against the two highest-ranked human players in the history of this show [21, 22]. In Jeopardy!, human contestants are provided with answers, called “clues”, to which they have to find the fitting question. The clues are grouped together by a topic or knowledge category and sorted by difficulty. The clues cover a broad range of topics, for example, culture, literature, languages, history and science. Table 2.1 displays some examples taken from the Jeopardy! episode with Watson [23].

IBM started the development of Watson in 2007. Their goal was to build a system powerful enough to understand the rich natural language of the clues and to master the complex reasoning of Jeopardy’s game-system in real-time. Equivalent to the other participants, Watson had to acquire all knowledge previous to the show and was not allowed to be connected to the internet or any other resources. Unlike the previously presented systems, Watson needed to generate correct answers in a limited period of time in order to win the game. On average, Watson had just three seconds to parse a clue, understand it’s meaning, find possible answers, determine the best one and, if confident enough, buzz in and express the answer out loud [21]. The developers accomplished that Watson could

<sup>1</sup><https://www.wolframalpha.com>

#	Category	Clue	Answer
1	What to wear	This plain-weave, sheer fabric made with tightly twisted yard is also used to describe a pie or a cake	What is Chiffon? <i>(Watson had a low confidence)</i>
2	Familiar Sayings	Even a broken one of these on your wall is right twice a day	What is clock? <i>(Watson answered correctly)</i>
3	Actors who Direct	A Bronx Tale	Who is Robert De Niro? <i>(Candidate was faster)</i>
4	Also on your computer keys	A loose-fitting dress hanging straight from the shoulders to below the waist	What is a Shift? <i>(Watson answered wrongly: Chemise)</i>
5	Dialing for Dialects	Sprechen Sie plattdeutsch? If you do, you speak the low variety of this language	What is German? <i>(Watson answered correctly)</i>

Table 2.1: The table shows examples from the participation of Watson in the quiz show Jeopardy!.

deliver correct answers with a precision higher than 85%, while buzzing in on at least 70% of the questions. This high target was necessary to be competitive. “Game Champions” buzzed in on an average of 45-50% of the clues, with an answer correctness of 85-95%.

The researchers of IBM made the success of Watson possible with an extensible software architecture called DeepQA [21]. The pipeline architecture integrated hundreds of algorithms that analysed the clues for features and found the evidence for the answers. For the analysis of the clues, Lally et al. [24] found the following four critical elements:

1. **Focus:** The part of a clue that is a reference to the answer.
2. **Lexical answer types (LAT):** Terms in a clue that indicate the type of entity the clue is looking for.
3. **Question Classification:** Classification of the clues into categories to tailor the following answering approach to the question. As this is an important part for QA systems and for this thesis, more information can be found in Section 2.2.
4. **QSection:** An annotated span in the clue or category that has a function in the interpretation of the clue. For example, a lexical constraint such as “this 4-letter word”.



Instead of focusing on one answer candidate, each stage of the pipeline could find multiple answers from different algorithms. For each candidate, the system tried to find supporting evidence. Every analysis taken for an answer was represented as a feature, which was combined to a single score of confidence. DeepQA utilized statistical machine-learning algorithms to weight the features. In DeepQA, structured and unstructured documents were used as knowledge sources for the candidate answers [25]. As unstructured resources, Watson used title-oriented documents from different sources (e.g. Wikipedia) and non-title-oriented documents, like news articles. To find answer candidates in these unstructured documents, reading comprehension was adapted as a strategy from other QA systems. For some clues, Watson relied on a structured knowledge-based representation of the data. Even if it has been proven to be difficult to translate all natural language into a machine-understandable knowledge representation, Chu-Carroll et al. considered this, for example, for questions where the clue and the category asked for a relation between the answer and a named entity (see Table 2.1 Example 3) [25]. While IBM Watson and Wolfram|Alpha demonstrate an impressive performance and made astonishing progress in question answering, both systems and their data are not open to the public but provided as Software as a Service (SaaS). To create datasets for comparative question answering, this thesis will use large-scale open-source data and make the datasets available to the public afterwards.

In recent research on open-domain question answering, large-scale datasets for reading comprehension have been published. Notable ones are the Stanford Question Answering Dataset (SQuAD) [26, 27]<sup>2</sup>, the ReAding Comprehension Dataset (RACE) [28] dataset<sup>3</sup> and the MACHine Reading COMprehension (MS MARCO) [29] dataset<sup>4</sup>. The SQuAD dataset consists of 100,000 answerable and 50,000 unanswerable questions generated by human workers. The questions refer to a passage of text taken from Wikipedia articles. The answers are a span within these passages. The dataset provides the text passage, the questions and the answers. On release, a logistic regression model performed an F1 score of 51.0. The RACE dataset is collected from English middle and high school examinations in China. The dataset consists of 100,000 questions on 28,000 text passages. For each question there are four answer candidates available. On release, state-of-the-art models achieved an accuracy of 43%.

In January 2018, a model from Alibaba’s artificial intelligence (AI) research arm exceeded the average human performance on the SQuAD dataset by 0.1% in the exact match metric for the first time [30, 31]. By that time, the unanswerable questions were not added to

---

<sup>2</sup><https://rajpurkar.github.io/SQuAD-explorer>

<sup>3</sup><https://www.cs.cmu.edu/~glai1/data/race/>

<sup>4</sup><https://microsoft.github.io/msmarco>

the dataset. In January 2020, the best models had a performance up to 3% higher than humans on the complete dataset. Even though the human performance on the RACE dataset has not been beaten so far, the leading models for both SQuAD and RACE are based on Google’s “A Lite BERT” (ALBERT) [32] language model. For SQuAD, the leading F1 score is 92.4 (41.4 higher compared to its release <sup>5</sup>) and for RACE the leading accuracy is 89.4% (46.4% higher compared to its release <sup>6</sup>).

Most of these open-domain reading comprehension datasets have in common the fact that they provide a small snippet of text in which the answer can be found (compare SQuAD). Qi et al. [33, 34] argues that systems trained on SQuAD, for example, are good in finding the answer in a provided text or article, but cannot help if the system does not already know where to look for information. So, finding the director of “A Bronx Tale” in the corresponding Wikipedia movie-article could be achieved, but finding it in a general manner within a large text collection would not be possible. Qi et al. state that this open-context and open-domain setting is much more challenging, but also more useful because it is not always clear where to find the answer to a question. Following Qi’s research, this work will only consider open-domain and open-context data sources to create a most diverse, versatile and general dataset for comparative question classification.

---

<sup>5</sup>Leader board available at <https://rajpurkar.github.io/SQuAD-explorer>. Scores as of 01/24/2020.

<sup>6</sup>Leader board available at [http://www.qizhexie.com/data/RACE\\_leaderboard.html](http://www.qizhexie.com/data/RACE_leaderboard.html). Scores as of 01/24/2020.

## 2.2 Question taxonomies

It is relevant for this thesis to study how comparative questions can be classified into existing taxonomies. This classification, together with the description of the existing taxonomies, is explained in this section.

As explained in Chapter 1, search engines like Google cover a massive amount of searches per year. The queried searches can be classified by a taxonomy proposed by Broder in 2002 [35].

- Informational queries
- Navigational queries
- Transactional queries

Informational queries have the purpose to address a information need of the users. Their content can range from simple broad queries, such as “cars”, to more specific queries, for example, “Game of Thrones Season 8 Review”. The informational class has the biggest share over all search queries. Studies assume a range from 48% to 80% of all searches to be informational [35, 36, 37]. The class of navigational queries ranges between 10% to 20% of all searches. Despite the low overall share, navigational queries are the most prominent in the US top 100 search queries of 2020 [38]. The class encompasses all queries with the user’s intention to reach a particular website. The three most searched navigational queries in 2020 are “facebook”, “youtube” and “amazon”<sup>7</sup>. Transactional queries cover all searches performing a certain action, for example, “download HD screen saver”. The share of transactional queries is estimated between 10-30% of all searches. In the search query taxonomy, comparative questions can be considered a part of the informational queries.

Most question answering systems employ Question Analysis to a certain extend. Similar to Watson’s analysis of clues (see Section 2.1), the systems need a question’s focus and a question’s class to produce more precise answers. For Watson, the question classes were very tailored to the Jeopardy! problem. Due to the nature of Jeopardy!’s categories and clues, IBM used classes like “Definition”, “Fill-in-the-blank”, “Number” or “Puzzle” [24]. More general class sets were proposed in early research by Lehnert [40] and Graesser et al. [41]. Lauer et al. [39] proposed the four psychological classes, shown in Table 2.2, that drive humans to ask questions. It can be seen that comparative questions only fit the categories “Correction of knowledge deficits” and “Monitoring common ground”.

---

<sup>7</sup>Data updated on the 04-12-2020 [38].

Class	Example
Correction of knowledge deficits.	Problem solving / contradiction / gap in knowledge
Monitoring common ground.	Confirmation of belief / establishing common ground / accumulating additional knowledge
Social coordination of action.	Indirect request or advise / asking permission / offer
Control of conversation and attention.	Greeting / change in speaker / reply to summons

Table 2.2: Psychological classes of questions proposed by Lauer et al. [39] in 1992.

In 2008, Graesser et al. [42] published a survey that summed up previous work on question taxonomies in different fields of research (see Table 2.3). They collected 16 question classes, which were either proposed by Graesser et al. or by Lehnert. In the survey, Graesser sorted the categories according to their complexity. They classified categories 1-4 as simple/shallow, categories 5-8 as intermediate and categories 9-16 as complex/deep. In this taxonomy, comparative questions belong to category eight. Therefore, Graesser et al. consider comparative questions to have an intermediate complexity.

Lauer et al. [43] proposed a taxonomy within the class of comparative questions (see Table 2.4). The study classified comparative questions into 12 classes and was conducted in the context of financial auditing. Despite the context, the classes provide an extensive overview of a speaker's intent when asking a comparative question.

The proposed taxonomies address questions and queries from different angles and in different levels of detail. What those taxonomies have in common is that they are focused on a question's intention or meaning, such as the taxonomy proposed by Lauer et al., which was based on the question's intention for comparative questions. The goal of this thesis is to provide a taxonomy with a better fit for comparative-question mining and classification. Therefore, a taxonomy based on the question's linguistic for comparative-question classification is introduced in Chapter 3.

#	Category	Example
1	Verification	Invites to give a yes or no answer.
2	Disjunctive	Is X, Y, or Z the case?
3	Concept completion	Who? What? When? Where?
4	Example	What is an example of X?
5	Feature specification	What are the proper-ties of X?
6	Quantification	How much? How many?
7	Definition	What does X mean?
8	<b>Comparison</b>	<b>How is X similar to Y?</b>
9	Interpretation	What does X mean?
10	Causal antecedent	Why/how did X occur?
11	Causal consequence	What next? What if?
12	Goal orientation	Why did an agent do X?
13	Instrumental/procedural	How did an agent do X?
14	Enablement	What enabled X to occur?
15	Expectation	Why didn't X occur?
16	Judgmental	What do you think of X?

Table 2.3: Taxonomy of questions proposed in a survey by Graesser et al. [42] in 2008.

Class	Description
Comparison with Verification.	Asks for verification of a comparative relationship.
Comparison and Disjunctive.	Asks which one of two comparative objects is the case.
Comparison with Concept Completion.	Compares the results of two implied questions.
Comparison and Feature Specification.	Asks for the comparison of features of comparative objects.
Comparison with Quantification.	Compares two quantities.
Comparison and Causal Antecedent.	Asks for a comparison of causes that affect two comparative objects.
Comparison and Causal Consequence.	Asks for the comparison of the effects of two causes.
Comparison and Goal Orientation.	Asks for the comparison of two goals.
Comparison and Enablement.	Compares the capabilities of two comparative objects.
Comparison and Instrumental/Procedural.	Asks for differences between two comparative objects.
Comparison and Expectational.	Compares an accrued situation with an expected one.
Comparison and Judgmental.	Asks for a comparison of two judgements.

Table 2.4: Taxonomy for comparative questions proposed by Lauer et al. [43] in the context of financial auditing.

## 2.3 Related work on comparatives

Additional to the previously proposed taxonomy for comparative questions by Lauer et al., there has been some more recent work on the classification of comparatives and the extraction of their comparative objects.

In their first paper from 2006, Jindal and Lui [44] proposed the Class Sequential Rule (CSR) as method to classify sentences into comparative and non-comparative. A Class Sequential Rule is a sequential pattern which gets assigned to a label. In this case, the labels are binary: comparative and non-comparative. To mine an initial dataset, a set of 83 keywords, commonly used in comparatives, was collected. The keywords included part-of-speech (POS) tags for comparative words, as well as manually collected phrases and words that are utilized in comparisons. Jindal et al. composed the initial dataset out of sentences containing at least one of the keywords. They found out that only 32% of their collected sentences were genuine comparative (precision), while it was possible to match 94% of the comparative sentences (recall). With this initial dataset the Class Sequential Rules were generated. The rules then acted as features for machine-learning classification methods. Finally multiple classifiers were trained. A Naïve Bayes classifier performed the best, with a precision of 79% and recall of 84%. The utilized source data came from different, non-disclosed sources, in the categories of consumer reviews, forum discussions and news articles. The datasets consisted of 4,985 non-comparative and 905 comparative samples (total: 5,890), which were manually labelled by four human annotators.

Jindal and Lui extended their work in the same year towards the extraction of comparative relations [45]. They defined a comparative relation as a set of entities that are compared, a relation word and one or more features to which the comparison is made: ( $\langle \text{relationWord} \rangle$ ,  $\langle \text{features} \rangle$ ,  $\langle \text{entityS1} \rangle$ ,  $\langle \text{entityS2} \rangle$ ). The definition allowed single nouns and pronouns as entities and features, but also blanks. Verb forms of nouns, like *cost* - *costs*, were left for future work. The comparative sentences taken from their previous work were classified into four subclasses:

1. **Non-Equal Gradable:** The relation between the entities provides an ordering, e.g. *greater than*, *better than*.
2. **Equative:** Equal relations between the entities, such as *equal to*.
3. **Superlative:** Superlative relations between multiple entities, e.g. *less than all others*.
4. **Non-Gradable:** Comparative sentences that do not explicitly grade the entities (e.g., “Toyota has GPS, but Nissan does not have”).

For each keyword from the list of keywords proposed in [44], the fitting subclass label (1-4) was determined. With this extended keyword list, the comparative sentences were re-evaluated to fit into the finer-graded subclasses. For the extraction of entities they only considered sentences from the gradable subclasses (1-3). Similar to the CSR approach, they proposed the Label Sequential Rule (LSR) as method to identify entities and features through sequential patterns. A naive base classifier archived an F1 score of 0.72. The precision of detecting the first entity was 100%, with a recall of 69%<sup>8</sup>, while the second entity had a precision of 85%<sup>8</sup>, with a recall of 59%<sup>8</sup>. According to Jindal et al., the different results of the prediction are due to the fact that the first entity had nicer characteristics, for example, occurring at the start of a sentence. The second entity usually appeared at any later point in a sentence and was harder to predict correctly. They stated that the recall of the system was significantly impacted by errors in the utilized part-of-speech tagger (POS tagger)<sup>9</sup>. An empirical evaluation of the dataset gave a distribution of 285 non-equal, 110 equative and 169 superlative sentences (total: 564). Between 65%-76% of the entities and features were nouns and not every sentence contained all three elements of the comparative relation set. Jindal et al. stated that for superlatives the second entity is normally empty, implying that the second entity is implicit or the comparison is against an open group like “all others”. The keyword list and annotated comparative questions have not been published.

Specialized on the topic of web search and search engines, Jain and Pantel proposed to use a comparable entity database to enable search engine users to perform a comparative analysis [46, 47]. Their work mainly focused on extracting comparative tuples (E1, E2) from a collection of 500 million web pages and 100 million query logs, both collected by the Yahoo! search engine, to build the comparable entity database. To extract the comparable entities from a text, they used a three stepped algorithm. In the first step, bootstrapping methods were used to learn extraction patterns for entity tuples. The algorithm started with a small set of given comparable tuples to find extraction patterns and then reapplied the found patterns to the text in following iterations. Exemplary patterns are “E1 compared to the E2”, “E1 versus E2” and “E1 or E2”. Further steps of the proposed algorithm took care of improving the quality of the entity tuples. To generate a comparative analysis Jain et al. furthermore proposed the use of a description database for each comparable pair, including a meaningful description with characteristics and attributes of the entities.

In 2010 Li et al. [48] extended the work of Jindal and Lui [44]. Their paper combined both, the classification of comparative sentences and the extraction of comparative entities, through bootstrapping sequential label rules, similar to the ones from Jindal and Lui.

---

<sup>8</sup>The score was taken out of a diagram and might not be completely accurate.

<sup>9</sup>POS-tagger: An automated system to label parts of speech in text.



The goal was to perform classification and extraction in on step. In contrast to Jindal, Li et al. made several different assumptions for their work. They defined a comparative question as a question that compares two or more entities (Comparators - \$C). The entities have to be explicitly stated in the question and can be a word, a POS tag or a symbol. Furthermore, POS constraints were allowed for the comparative entities, for example, “\$C/NN” requiring the entity to be a noun. Li et al. called their sequential rules indicative extraction pattern (IEP). They based their work on two assumptions:

1. “If a sequential pattern can be used to extract many reliable comparator pairs, it is very likely to be an IEP.” [48]
2. “If a comparator pair can be extracted by an IEP, the pair is reliable.” [48]

For their experiment they utilized 60 million questions collected from Yahoo! Answers. To generate an evaluation dataset (SET-A), 5,200 samples were manually annotated by two human workers with the labels “comparative”, “non-comparative” and “unknown”. The manual labelling resulted in 139 comparative questions and was therefore repeated with more data. For this second set (SET-B) of 2,600 samples, a pre-filtering with a keyword list was done to achieve a higher count of comparative samples. The list included keywords like “or” and “prefer”, but it was not disclosed completely. SET-B was again manually classified and also the comparative entities were annotated. The dataset contained 853 comparative questions. This results in an evaluation set of 7,800 questions with 992 comparatives. Li et al. implemented the CSR and LSR methods proposed by Jindal et al. and trained them on the labelled datasets. After training and evaluation, Li et al. concluded that the errors in Jindal’s experiments were caused by too specific rules and overfitting the small training set, not only because of errors in the POS tagger. Due to the nature of their approach, Li et al. could generate the sequential patterns in a weakly supervised way on all the unlabelled data and tested it against the labelled datasets. In question classification they achieved a precision of 83% and a recall of 82%. This is a plus of 35% compared to their implementation of Jindal et al. in recall, while the precision was nearly the same. For the extraction task, their method accomplished a precision of 92%(+6)<sup>10</sup> and a recall of 76%(+14)<sup>10</sup>. Li et al. found 328,364 unique comparator pairs with 6,869 extraction rules. In total, 679,909 comparative questions were identified by their method from 60 million samples. The keyword list, the extraction rules and the found comparative questions have not been published.

A more recent research (2019) was done by Panchenko et al. [49] as a part of the ACQuA project [12]. A dataset and a classifier to identify comparative sentences was proposed in this research. They utilized entity pairs from three different domains to mine sentences

<sup>10</sup>Compared to the implementation of LSR by Li et al.

from a web-scale corpus derived from the Common Crawl and explicitly excluded questions. The final dataset consisted of 7,199 sentences of which 1,957 (27%) samples were comparative. The sentences were labelled as BETTER, WORSE and NONE. The labels BETTER / WORSE describe the relation of the first comparative entity to the second entity in a sentence (“Golf is better than Tennis.” - Label: BETTER). The label NONE marked non-comparative sentences. In their experiments Panchenko et al. compared a variety of different machine learning models<sup>11</sup> and feature sets<sup>12</sup>. The best result for classification of comparatives was accomplished by a gradient boosting model (XGBoost<sup>13</sup>) with InferSent [50] sentence embeddings as a single feature. Over all three classes the classifier could reach an F1 score of 0.85. According to Panchenko et al., tested neural networks showed no improvement compared to the machine-learning models. The best model of the work of Panchenko et al. was employed in the Comparative Argumentative Machine (CAM), presented by Schildwächter et al. [13], for sentence retrieval and classification. CAM is an open-domain information retrieval system to argumentatively compare objects. According to Schildwächter et al., the system proved to be 15% more accurate and 20% faster than a “traditional” users search. Users can enter two comparative entities and multiple aspects in dedicated fields in the CAM frontend to get a comparative analysis (see Figure 1.3). One goal of this thesis is to extract comparative entities and their aspects from continuous text and to provide them to systems like CAM.

In the latest work on comparatives (2020), Bondarenko et al. [51] proposed a method to identify comparative questions in a Russian dataset and to classify them into 10 fine-grained subclasses. Their work used a combination of rule-based classification, encompassed by regular expressions and neural-based classifiers. The 10 classes categorized comparative questions into 5 general classes known from question taxonomies (e.g. argumentative or factoid) and further five syntactic and semantic classes like superlative questions, questions containing aspects or questions providing more context to the comparison. The ten classes were not mutually exclusive (multiple classes could be assigned to a question). The work of Bondarenko et al. defined questions that display any intent of a comparison between two or more sets of entities as comparative. The entities could be explicit in the sentence or implicit as an open set, like in this example: “Which tablet is best to buy?”. Four native Russian-speaking annotators labelled an individual share of 62,500 samples as comparative or non-comparative (one vote per sample). The source data came from a query log of the Russian search engine Yandex<sup>14</sup>(50,000 samples) and the Russian question answering platform Otvet<sup>15</sup> (12,500 samples). One of the goals

---

<sup>11</sup>The tested models include XGBoost, Logistic Regression, SVM , Decision Trees and more.

<sup>12</sup>The feature sets included Bag of Words, Bag of N-grams, POS-Tags, Word Embeddings and more.

<sup>13</sup><https://xgboost.readthedocs.io/>

<sup>14</sup><https://yandex.ru/>

<sup>15</sup><https://otvet.mail.ru/>

of Bondarenko’s work was to have near to perfect classification precision, to ensure that comparative labelled questions had no false positives. To achieve this goal, they classified questions first with a rule-based classifier, which only incorporated rules that had perfect precision. In case the rule-based classifier labelled a question non-comparative, an ensemble of a convolutional neural network (CNN) model and a logistic regression model were used to classify the question again and find a consensus. In their experiments Bondarenko et al. used their ensemble classifier to label their complete source data and then manually annotated 5,000 comparative questions. Through this extension, the set of comparative questions had a total size of 6,250 samples. Bondarenko et al. explained that the ensemble of rule-based, CNN and logistic regression classifier is the preferred classifier because it is faster, even though another combination had a 1% better recall. The ensemble reached a recall of 59% at a precision of 100%. Furthermore, Bondarenko et al. extend their binary classification with a following step of classifying the fine-grained subclasses of comparatives. For this step CNN and BERT [52] models were utilized. The classification was optimized for the micro-averaged F1 measure and achieved 0.91 F1 score (BERT). The not mutually exclusive subclasses with the most training data performed the best (up to 0.97 F1 score). The class for aspect-providing sentences had the worst F1 score (0.74). Together with the results, Bondarenko et al. released a separate set of 15,000 English questions collected from various QA datasets available for the public. The questions were manually labelled as comparative or non-comparative.

This thesis will study the identification of comparatives and the extraction of the comparative entities, similar to the goals of Jindal et al. [44]. For mining comparatives from a large scale data source, a linguistic-based approach on comparatives and their generation rules will be developed, as well as a keyword list similar to the one proposed by Jindal and Lui [44]. Unlike Jindal et al. and Bondarenko et al. [51], the utilized source data will be extracted from English open-domain datasets, which are publicly available. The data will be manually labelled in crowd-sourced annotation tasks to create comparative question datasets. In contrast to work from Bondarenko et al. [51] or Jindal et al. [44, 45], the classification and extraction of comparatives will be done with feature-based machine-learning algorithms, as well as neural machine learning, without any dependency on predefined or generated rules. Furthermore, this thesis aims to provide an entity extraction in an open domain, without prior definition of entities or entity pairs (in contrast to [48, 49]). Moreover, a more restrictive definition of comparative questions than the one from Bondarenko et al. [51], yet a more open one than the one from Jindal et al. [44], will be taken. To explain this definition of comparative questions, the next chapter will give an insight into the linguistic background of comparatives and questions.



## Chapter 3

# Creation of comparative question datasets

In this chapter, the linguistic background on comparative sentences is studied and a definition of comparative questions is formulated. With this definition a linguistic-based taxonomy for comparative questions is proposed and used to mine comparatives from publicly available data. This chapter will end with the creation of two novel open-domain comparative question datasets with the help of crowd sourcing annotations.

### 3.1 Linguistic background on comparative questions

Classical explanations in English school textbooks describe a certain structure to be used in comparatives. Despite these simplifications, the rich English language allows various ways and structures to express a comparison. To have a clear definition that fits this thesis goals, the various linguistic structures of comparisons, the restrictions and adaptations needed are studied and explained in this section.

#### 3.1.1 Rule-based comparisons

Bas Aarts explains comparisons in the book “Oxford modern English grammar” [53] as clauses that describe an equality or inequality between two terms. A comparison of equality is expressed with a comparative clause as a Complement of the preposition “as”. Whereas an inequality is expressed with the preposition “than”. The creation of these structures can be done using the comparative or superlative form of adjectives or adverbs.

**Comparison of equality:** The oak tree is as tall **as** the maple tree.

**Comparison of inequality:** The oak tree is taller **than** the maple tree.

### 3.1.1.1 Comparative and superlative adjectives

Forming the comparative or superlative form of adjectives follows some easy rules. From the base form (positive form) of an adjective there are two ways to create the comparative or the superlative form: the Suffix Method and the Adverb Method [53, 54, 55, 56].

**Suffix Method:** Add the suffix **-er** for comparative adjectives and the suffix **-est** for superlative adjectives.

**Adverb Method:** Add an adverb directly in front of the adjective. For example **more**, **less**, **most** or **least**.

Comparative Adjectives			
Rule	Method	Action	Example
one syllable	Suffix	ADJ+er	tall - taller
two syllable	Adverb	more/less ADJ	evil - more evil
two syllable & end in y	Suffix	drop y & ADJ+ier	angry - angrier
>= three syllables	Adverb	more/less ADJ	important - more important
consonant + single vowel + consonant	Suffix	double last consonant + er	big - bigger

Table 3.1: Inflection rules to build comparative adjectives.

More than 500 adjectives use the suffix method, but it is even more common that adjectives use the adverb method. The method that is needed mainly depends on the number of syllables and the word ending. Tables 3.1 and 3.2 show the rules, the method and the according actions in detail. Many adjectives are gradable and can be assigned a degree of “strength”. This is done by adding an adverb in front of an adjective. Even though this is similar to the Adverb Method for comparative and superlative inflections, both modifications deal with a different matter of speech. Adverbs like *very*, *extremely*, *mildly*, *truly*, *etc.* can be used to grade adjectives, for example, “Peter is an extremely kind child.”.

Superlative Adjectives			
Rule	Method	Action	Example
one syllable	Suffix	ADJ+est	tall - tallest
one or two syllable ending in e	Adverb	ADJ+st	rare - rarest
two syllable & end in y	Suffix	drop y & ADJ+iest	angry - angriest
all other two & $\geq$ three syllables	Adverb	more/less ADJ	important - most important
consonant + single vowel + consonant	Suffix	double last consonant + est	big - biggest

Table 3.2: Inflection rules to build superlative adjectives.

There are very few irregular adjectives, which form their comparative and superlative forms completely different and not based on the Suffix- or Adverb Method. From the six adjectives in their base form, eight comparative forms can be created (see Table 3.3). Best known are, for example, *bad - worse - worst* and *good - better - best* as irregular adjective inflections. A comparative sentence using the irregular adjective *good* is “Bananas are better than apples.”. For the words *far* and *old* the comparative and superlative forms depend on the intended meaning. The inflection *far - farther - farthest* is used for distances, “London is 50km farther away than Paris.”, while *far - further - furthest* can be used for distances or time, “London is 30 minutes further away than Paris.”. Similar, *old - elder - eldest* can only be used for people, while *old - older - oldest* is used for people and things.

Base form	Comparative	Superlative
bad	worse	worst
good	better	best
little	less	least
many / some / much	more	most
far	farther	farthest
far	further	furthest
old	elder	eldest
old	older	oldest

Table 3.3: Inflections of irregular adjectives.

Apart from the irregular adjectives, some words do not follow the method indicated by the rules or it is left as the author's choice how to spell the word. For example, the base adjective *unlucky* leaves the author the choice to say "He is the *most unlucky* person in his class." or "He is the *unluckiest* person in his class.". Another rule breaking word is "clever". The word has two syllables and should therefore follow the rule to add an adverb like "more/less" in front. Instead, for "clever" the suffix *-er* is added. The sentence "Batman is *cleverer* than the Joker." illustrates this.

**Rule breaking words that except a suffix or are up to the author's choice [54]:** able, bitter, clever, eerie, evil, feeble, foolhardy, gentle, handsome, humble, little, mellow, narrow, nimble, noble, pleasant, polite, quiet, remote, rotten, secure, serene, severe, shallow, sincere, subtle, sulky, unhappy, unlikely, unlucky, unruly

Figure 3.1 gives examples of sentences created with the Suffix (1) and Adverb Methods (2 & 3) using prepositions to express a comparison of equality (so/as) or inequality (than).

### 3.1.1.2 Comparative and superlative adverbs

Similar to forming comparisons and superlatives with adjectives, it is also possible to use adverbs [53, 54, 57, 58]. The same two Suffix- and Adverb Methods from Section 3.1.1.1 are used to create the adverb inflections. In contrast to adjectives, nearly all adverbs use the Adverb Method. Table 3.4 shows the 22 adverbs that use the Suffix Method. According to Brager [54], this list of 22 suffix adverbs can be considered complete and might only lack a few very uncommon adverbs. All *-er* and *-est* adverbs in the list are also adjectives and spelled the same way. Adverbs have the same irregular forms as the equivalent adjectives (marked with \* in Table 3.4). For example, the adjective *bad* and the adverb *badly* form the same inflection *worse - worst*. In many cases, it is possible to add *-ly* to an adjective to create an adverb.

The Adverb Method works the same way as with adjectives. An adverb is added in front of the comparative adverb, which keeps its base form. In comparative sentences the adverbs *more*, *most* and *least* are most commonly added, for example, *frequently - more frequently*. The sentences 4-6 in Figure 3.1 serve as examples of sentences created with the Suffix (4) and Adverb Methods (5 & 6) using expressions for comparisons of equality (so/as) and inequality (than).



#1	<i>The oak tree</i> Subject (Noun)	<i>is</i> verb	<i>taller</i> Comp. Adjective (Suffix)	<i>than</i> Preposition		<i>the maple tree.</i> Direct Object (Noun)
#2	<i>The oak tree</i> Subject (Noun)	<i>is</i> verb	<i>more</i> Addition (Adverb)	<i>important</i> Comp. Adjective (Base)	<i>than</i> Preposition	<i>the maple tree.</i> Direct Object (Noun)
#3	<i>The oak tree</i> Subject (Noun)	<i>is</i> verb	<i>not so/as</i> Addition (Adverb)	<i>tall</i> Comp. Adjective (Base)	<i>as</i> Preposition	<i>the maple tree.</i> Direct Object (Noun)
#4	<i>The oak tree</i> Subject (Noun)	<i>is</i> verb	<i>faster</i> Comp. Adverb (Suffix)	<i>growing</i> verb	<i>than</i> Preposition	<i>the maple tree.</i> Direct Object (Noun)
#5	<i>The oak tree</i> Subject (Noun)	<i>grows</i> verb	<i>more</i> Addition (Adverb)	<i>often</i> Comp. Adverb (Base)	<i>than</i> Preposition	<i>the maple tree.</i> Direct Object (Noun)
#6	<i>The oak tree</i> Subject (Noun)	<i>grows</i> verb	<i>as / not as / not so</i> Addition (Adverb)	<i>often</i> Comp. Adverb (Base)	<i>as</i> Preposition	<i>the maple tree.</i> Direct Object (Noun)

Figure 3.1: The figure shows examples of comparative sentences containing adjectives (1-3) and adverbs (4-6) that can be constructed with the Suffix (1 & 4) and Adverb Method (2-3 & 5-6). The examples display the expression of inequality (1-2 & 4-5) and equality (3 & 6) between two terms.

Base form	Comparative	Superlative	Base form	Comparative	Superlative
badly	worse	worst *	little	less	least *
bright (ly)	brighter	brightest	loudly	louder	loudest
close (ly)	closer	closest	low	lower	lowest
deep	deeper	deepest	much	more	most *
early	earlier	earliest	near	nearer	nearest
far	farther	farthest *	quick (ly)	quicker	quickest
far	further	furthest *	slow (ly)	slower	slowest
fast	faster	fastest	some	more	most *
hard	harder	hardest	soon	sooner	soonest
high	higher	highest	tight (ly)	tighter	tightest
late	later	latest	well	better	best *

Table 3.4: Inflections of adverbs using the Suffix Method. The asterisk (\*) marks the irregular forms. Some adverbs can be utilized with or without the displayed *-ly* in the parentheses.

### 3.1.2 Phrase-based comparisons

With the inflection of adjectives and adverbs, the English language provides a set of properly defined methods to create comparatives (introduced in Section 3.1.1). Furthermore, we can utilize the language more, by using lexical items, like single words or phrases, to create a comparative sentence. A well known example is the phrase *compared with*. As there is no clear definition of which lexical items can be used to introduce a comparison, an extensive list of possible words and phrases was collected from different sources [59, 60] for this thesis. The list (see Table 3.5) contains 55 words or phrases and was created in a non-restrictive way. In this case, non-restrictive means that many of the items are not exclusive for comparative sentences. Many phrases and especially the single words can be utilized in both, comparative and non-comparative sentences. They are included in the list to gain a high coverage of possible words or phrases utilized by people to express a comparison. For example, the word “like” (see example below) is not exclusively used in comparative sentences.

**Comparative:** Istanbul is *like* Athens.

**Non-comparative:** I do not *like* oranges.

Furthermore, for many of the collected lexical items it is not clear how frequently they are used by people to form a comparison. It might be possible to create an abstract example with a phrase, yet nobody would use it in a natural way of speaking. To provide an initial idea if a phrase or word can be used in a comparison, an example sentence was created for every word and phrase. Moreover, in Section 3.3 the usage frequency of the phrases in a set of questions, taken from online platforms, is analysed. Sorting the list by their part of speech, phrases represent the majority with a count of 25 (45%) expressions. Prepositions (27%) like *versus* and adjectives (22%) like *close to* represent the second biggest group. Conjunctions (6%) like *or* conclude the list. An empirical test based on the creation of examples, showed that at least 36 lexical items (65%) could be used in a comparative question. The box below provides some example sentences.

**Phrase:** Athens is small *compared to* Rome.

**Preposition:** Apple’s technology is *ahead of* Google’s.

**Adjective:** Gold is *equal to* Platinum.

**Conjunction:** Amazon is a company, while Athens is a city.

The creation of a relationship between two ideas or subjects is achieved by words that point out the similarities or the differences between them [61]. Showing similarities is done by means of comparison transition words, while opposing concepts can be achieved by means of contrast transition words. Some of the lexical items in the list can be assigned to one of these classes. The words *similar* and *dissimilar* are prominent examples for each class. Even though this classification can be made, this thesis will not distinguish between comparison and contrasting, but it will go with the more general meaning of a comparison of building a connection or relation between two objects, similar to the definition of the Oxford English Dictionary “Comparison: A consideration or estimate of the similarities or dissimilarities between two things or people.” [62].

after	by comparison	in contrast to/with	or	unequal to
against	close to	in relation to/with	over against	unlike
ahead of	compared to/with	just as	related to/with	unrelated
alike	different to/from	like	relative to	versus
all-time	dissimilar to/from	near to	relative	vs.
alongside	equal to/with	next to	relatively speaking	whereas
as much	even	not like	seen against	while
beside/s	in comparison to/with	not only ... but also	similar to	
between	in contradistinction to/from	not the same as	the same (as)	

Table 3.5: List of lexical items (phrases and words) utilized in comparisons. The full list can be found in the appendix in Table 1 and Table 2

### 3.1.3 Generation rules of questions

According to the definition of the Oxford English Dictionary, a question is “a sentence worded or expressed so as to elicit information” [63]. Interrogative clauses are commonly used to express a question. Interrogative clauses can be classified into open and closed interrogative clauses leading to open and closed questions [53]. Closed questions can typically be answered with a simple *yes* or *no* while open questions aim for a more complex and unrestricted response. The usual word order in interrogative clauses differs in the initial questions word (wh-word) used in open questions (see box below) [64].

**Usual word order in open question:**

question word + auxiliary/modal verb + subject + main verb ( + extra information...)

**Usual word order in closed question:**

auxiliary/modal verb + subject + main verb ( + extra information...)

In open questions, *wh*-words usually start the sentence. There are nine question words used in open questions [65]. Table 3.6 gives an easy example for each question word and lists the expected subject of an answer. Even though the content of an answer to open questions is not limited, the used *wh*-word determines the expected subject of the answer. For example, with the question *When is the supermarket closing?* the reader expects an answer related to time.

Question Word	Answer Subject	Example
What	Thing	What is your name?
When	Time	When is the party?
Where	Place	Where is the bank?
Which	Thing	Which car is yours?
Who	Person	Who is your brother?
Whom	Person	Whom do you believe?
Whose	Person	Whose dog is this?
Why	Reason	Why is there pizza?
How	Directions	How is the weather?

Table 3.6: List of question words for open interrogative clauses and the expected subject of the answer.

Closed interrogative clauses do not use *wh*-words but a auxiliary or a modal verb to express a question. It can be distinguished between the core auxiliary verbs *be*, *do*, *have* and the modal verbs *can*, *could*, *may*, *might*, *must*, *ought to*, *shall*, *should*, *will*, *would* [66]. Both lists show the positive form of the verbs. For each of them there is a form of negation, normally created with *not* and a short version of the negative form (e.g. *could* - *could not* - *couldn't*). Only the verbs *may* and *might* have no negative short form. In difference to the core auxiliary verbs, the modal verbs never change their form in a question. Modal verbs are used to express modality such as ability, possibility, necessity or intention. Thus, a

closed question with a modal verb asks for this modality. For example, the question “Might there be rain today?” asks for the possibility of rain. It is clear that the answer to this question can be *yes* or *no* and might be followed by a reasoning, for example “Yes, there are clouds in the sky.”. Furthermore, this questions demonstrates the difference between the type of expected answer and the response to a question. While the answer is determined by the closed interrogative clause (yes/no), the response could be “I hope so.” [53]. Table 3.7 provides examples of closed questions and their implied modality of modal verbs.

Verb type	Verb base form	Implied Modality	Example
Core auxiliary	Be		Is she Canadian?
Core auxiliary	Do		Does she have a brother?
Core auxiliary	Have		Had i told him that?
Modal auxiliary	Can	ability	Can i talk?
Modal auxiliary	Could	ability	Could he jump?
Modal auxiliary	May	possibility	May we go?
Modal auxiliary	Might	possibility	Might there be rain?
Modal auxiliary	Must	necessity	Must we fly?
Modal auxiliary	Ought to	necessity	Ought she to call the police?
Modal auxiliary	Shall	intention	Shall we walk home?
Modal auxiliary	Should	necessity	Should we walk home?
Modal auxiliary	Will	intention	Will i be rich?
Modal auxiliary	Would	intention	Would a dog like to fly?

Table 3.7: List of core and modal auxiliary verbs to use in closed interrogative clauses. For the modal auxiliary verbs the implied modality of the question is listed.

### 3.1.4 Definition of a comparative question

With the help of the previous sections about the generation of comparisons (Section 3.1.1 & 3.1.2) and questions (Section 3.1.3), it is possible to form a clear definition of what is considered to be a comparative question for this work.

The first general requirement is that, in contrast to the work by Bondarenko et al. [51], the questions need to be written in the English language. Therefore, in Section 3.3 only English source data will be considered for this task. The following obvious requirement for a comparative question, is to be a question in the sense of Section 3.1.3. Both open and closed question are considered here. The only restriction made is that the sentence needs to start with a question word (see Table 3.6) or, in case of a closed question, a core or modal auxiliary verb (see Table 3.7). In closed questions, all positive and negative forms of the core auxiliary verbs are allowed. It is not necessary for a sentence to follow the common punctuation. As a result, the question mark (?) at the end of a sentence is not considered a requirement for the English language. Furthermore, the semantics or further syntactics of a question are not relevant.

For a question, in order to qualify as comparative question, a comparison needs to be made. This work follows the linguistic definitions of a comparison introduced in Section 3.1.1 and extends it by the construction of comparative sentences through phrases, which are presented in Section 3.1.2. Similar to the definition of comparison from the Oxford English Dictionary [62], a sentence is considered a comparative question, if the question asks for an assessment or estimate of the similarities or dissimilarities of two “things”. The compared “things” are referred to as objects of the comparison or the comparative objects.

The group of sentences is limited to linguistic comparisons and, therefore, it excludes superlatives. Not only is this restriction strongly connected to the linguistic definition of a comparison, but also to the Comparative Argumentative Machine (CAM). Systems like CAM receive the output created by the natural language processing (NLP) components presented in Chapter 4. The goal is to provide exactly two comparative objects as an output. The two comparative objects can be anything. Common examples of these objects are: material objects, situations, actions, people, abilities, feelings and any other non-material things. Accordingly and in contrast to the work of Panchenko et al. [49], the content of a question is not restricted to a specific domain.

Although anything can be compared, the objects need to fulfil the following three conditions. Firstly, they must be comparable. Secondly, the objects must be in themselves complete. This means that it is not allowed for the comparative objects to represent an

#	Example
1	Why is BMW better than others?
2	Is the left one higher than the right one?
3	Is it healthier to eat two burgers or to drink two litres of cola per day?
4	Are jewelleries in Hong Kong cheaper than in Singapore?
5	Why do children learn languages faster than adults?
6	Why is it that hot water cleans better than cold water, when washing a jeans?

Table 3.8: Example sentences for the definition of comparative questions.

open group or class of things. In the first example (see Table 3.8), the comparison is made between the car manufacturer BMW and other car brands. These types of comparisons against open groups are hard to argue over and to answer, even for humans. Moreover, knowledge about the first comparative object is needed to correctly conclude that “others” stands for car manufacturers. Additionally to being self complete and comparable, the comparative object needs to be explicit in the sentence. For example, the second question (see Table 3.8) makes perfect sense with a picture of two mountains, but it is impossible to answer without this additional information, since the comparative objects are implicit in the sentence. These restrictions are again based on systems like CAM as receivers of the output. Restricting the objects to be self complete, explicit and comparable enables the systems to find answers for the questions.

Furthermore, a comparative object does not need to be one word or a noun phrase. It can be composed of multiple words and form longer structures. A typical example would be the comparison of two situations (see Example 3 in Table 3.8). The structure “eat two burgers” forms the first comparative object and “drink two litres cola” the second one. Due to the flexibility of the English language, two comparative objects might have a shared dependency within a sentence. The real-world example “Are jewelleries in Hong Kong cheaper than in Singapore?” from Yahoo! Answers shows two things (see Table 3.8). Firstly, the two objects *Hong Kong* and *Singapore* have a shared dependency to the word *jewelleries*. This shared dependency will be called a shared comparative object or shared object. The complete objects, which are compared in this example, are *Hong Kong jewelleries* and *Singapore jewelleries*. Secondly, the word “jewelleries” does not exist in English. It may be that the author meant jewellers, the places where jewellery is sold, or he refers to jewellery itself, which is uncountable and has no plural form. Anyhow, sentences with mistakes like this one, or with other spelling mistakes, will still be considered comparative. Another general definition taken is that the data is not restricted to single sentences, but it can also be made up out of multiple sentences. Besides, not every

sentence in these short texts has to be comparative.

Not only do some questions ask for a comparison of the two objects in an overall manner, but also for certain abilities or features of the comparative objects. Example five (see Table 3.8) displays one of these cases. The comparative objects are “children” and “adults” and the question asks for their ability to learn languages. This concretization of the aim of a comparison is referred to as the aspect of the comparison.

In addition to the definitions established so far, two features of comparisons, which are recognised, but not taken into consideration for this work, must be mentioned. Firstly, a sentence might contain a modulation or modification of an aspect. In example five (see Table 3.8), “faster” modifies the aspect “learning languages”. The word “faster” must not be a part of the aspect, as the sentence is not asking if children or adults are faster. Therefore, these constructs will be called the attribute of the aspect. Secondly, a part of a sentence can provide more information or restrict the context of the comparison. Example six (see Table 3.8) demonstrates that, while comparing “hot water” to “cold water” regarding the aspect of “cleaning”, the end of the question restricts the context to “washing jeans”. Both, the attributes of aspects and the additional information, will not be considered in this thesis, to keep the task simple enough for crowd sourcing (see Chapter 3.5).

In conclusion, this definition of a comparative is more open regarding the lexical type and word count of compared objects than, for example, Jindal and Lui’s work [44]. But also it is more restrictive than Bondarenko et al. [51], through the exclusion of superlatives or comparisons against open groups and the restriction to English sentences. Furthermore, there will be no prior definition of entities or entity pairs like Li et al. and Panchenko et al did. [48, 49]. The box below sums up all the definitions taken in this section. In the following section, a linguistic-based taxonomy is proposed based on this definition of a comparative question and the generation rules from the previous chapters.



### Definition of a comparative question

- The text ...
  - ... has to be written in the English language.
  - ... might be multiple sentences long.
  - ... has to be an open or closed question, starting with a question word or auxiliary verb.
  - ... might contain spelling mistakes or incomplete punctuation.
- The question ...
  - ... has to contain a comparative between two comparative objects.
  - ... must not be a superlative or compare more than two comparative objects.
  - ... might contain a shared object.
  - ... might contain an aspect of comparison.
  - ... might contain additional information or attributes of the aspect, which will not be part of this study.
- The objects ...
  - ... and sentences are not restricted in their domain.
  - ... need to be comparable, explicit and in themselves complete.

### 3.2 Linguistic-based taxonomy for comparative questions

Based on the definition of comparative questions and the background on their linguistic structures in Section 3.1, this section provides a taxonomy for comparative questions. In contrast to the work from Lauer et al. [43], the taxonomy is based on the linguistics of the question and the comparative instead of its content or meaning. This taxonomy has been developed to be suitable for data mining and classification of comparatives. The classes of the taxonomy are organized in a hierarchical tree-like structure. With every level of depth the definition of the class gets more detailed. The classes basically reassemble the construction cases explained in Section 3.1. Instead of naming the classes, a decimal encoded number system is utilized to address each class.

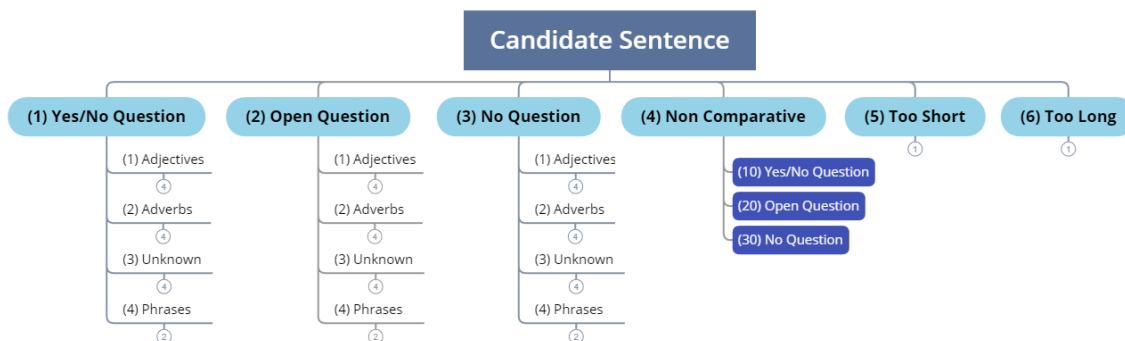


Figure 3.2: The figure shows the first two levels of the linguistic-based taxonomy for comparative questions. Each candidate sentence can be sorted into one of the groups. The Groups 1-4 diverge into more detailed subtrees, while the Groups 5 and 6 end in the first level.

Figure 3.2 shows the top level of the taxonomy tree (blue boxes). A candidate sentence is sorted into one of the six groups. Too short or too long sentences are sorted under Nodes 5 and 6. These two classes have no child nodes and are not divided any further. All non-comparative sentences are collected under the fourth node. The node is further divided into child nodes for closed questions (10), open questions (20) and non-questions (30). The class number is composed of the top-level node number, multiplied by 100 and followed by the number of the corresponding child. A non-comparative (4) open question (20) has the class number 420. The box below shows more examples of the class-number system. The top-level nodes one to three represent the comparative sentences, either as a question (nodes 1 & 2) or as a statement (Node 3). Each of these three “comparative nodes” has the same four children, which separate the candidate sentence further by its comparative generation structure (see Section 3.1.1 & 3.1.2).

**Examples of the class-number system:**

*Class 110:* Comparative closed questions of inequality, generated with adjectives and the Suffix Method.

*Class 232:* Comparative open questions of equality where no adjective or adverb is used.

*Class 321:* Comparative statements of inequality, generated with adverbs and the Adverb Method.

*Class 410:* Non-comparative closed question.

*Class 500:* A too short sentence.

Figure 3.3 displays the full sub-tree structure of the top-level Nodes 1-3. The second-level nodes categorize the sentences into a rule-based generation by adjectives (Node 1), adverbs (Node 2) and unknown word structures (Node 3). The fourth node categorizes the generation by phrases (Node 4). The rule-based nodes split up into for child nodes. The first two children combine a comparison of inequality with the Suffix (Child 1) and Adverb Method (Child 2). Children three and four are for comparatives of equality. The words written in brackets symbolize the exchangeable part of the comparative generation. For example: a candidate sentence with the class 210 is an open question with an adjective, generating the comparative of inequality with the Suffix Method and the preposition “than”. In this class the adjective and its suffix, as well as the verb, can change.

A special class is the second-level node for unknown structures. This node gathers all sentences containing an unknown combination of words instead of the exchangeable adjectives or adverbs between the fixed parts of a sentence. This is necessary since the rules to build a comparison are very simple and the language allows more combinations of words

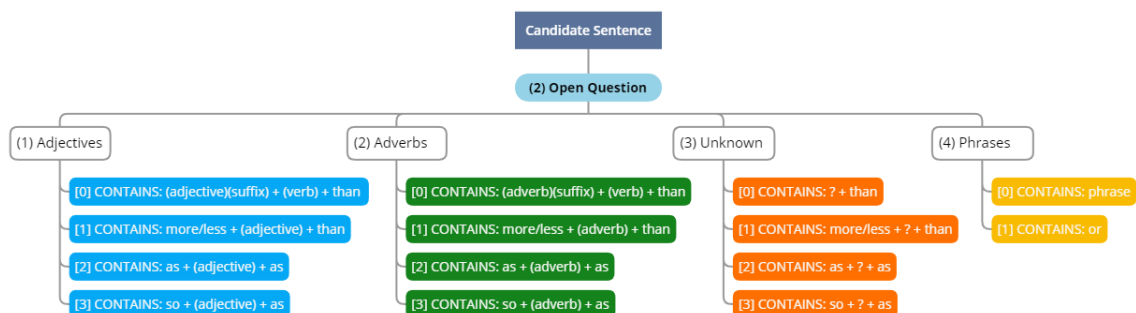


Figure 3.3: The figure shows the second and the third level of the linguistic-based taxonomy for comparative questions exemplary on the top-level node for open questions. The children categorize a sentence by its comparative generation.

in this place. “Was Europe and Greenland hotter in the past than they currently are?” is an example of a sentence in which a specification of the time is made. Furthermore, some authors might not follow the existing rules, for example, due to colloquial language or mistakes. By having a class for these unknown structures, the class system remains open to recognize all possible comparatives.

The fourth node collects all comparatives which are generated by the use of phrases. The node is further divided into a child for all comparative phrases and one for the special case “or”. The conjunction “or” can be used so versatile in the English language, that these sentences might have a high false-positive rate. For example, in the sentence “Could someone explain how a touchscreen on an iphone or android smartphone works?” the objects clearly form a list and are not comparative. A more difficult example is the sentence “Does coffee or espresso wake you up?”. It is not completely clear if the author had the intention to compare coffee and espresso, regarding the aspect of “waking people up”. Or if coffee and espresso are a list of examples for beverages containing caffeine.

## 3.3 Data mining

### 3.3.1 Evaluation of data sources

A large amount of data is needed to classify comparative questions with machine-learning techniques. The class distribution in such a large dataset should be balanced to easily achieve the goal. Because of this, the source data for those tasks needs to contain a high amount of comparatives. Previous work, for example, from Bondarenko et al., estimated a ration of 3% of comparative sentences in the search engine logs they utilized. The goal of this section is to find data sources that contain at least a few thousand comparative question samples. Calculating with the estimated ratio of Bondarenko et al., this would mean that the data source needs to be at least 160,000 samples large. Therefore, different data sources are evaluated in this section. There are five main requirements that a data source needs to fulfil. Firstly, it needs to have a high count of comparative questions. Secondly, the goal is to work with human written text and not just with short, keyword-focused, queries that are typical inputs in search engines. Thirdly, only data that is available to the public should be considered. On the one hand, this allows a publication of the final datasets created in this thesis and, on the other hand, it ensures that the datasets are available for future work. For example, for the extension of the datasets with more annotated data from the data sources. Fourthly, the goal of this thesis is to build an open-domain dataset. Therefore, in the best case, the source is not restricting any type of question or topic. The data should mirror the real behaviour of people and their language in questions. As a last requirement, the source should provide additional information with the question. This is not strictly necessary, but it would be much appreciated. A good example for additional information are ranked answers to the questions. These could be utilized in future research for information retrieval and for answering comparative questions. The box below shows a summary of the requirements:

**Requirements for source data:**

1. High count of potential comparatives
2. Higher average sentence length
3. Publicly available data
4. Open-domain data
5. Providing additional information

In the following sections, data from 7 different sources will be evaluated according to these requirements. Six out of the seven sources published the data as composed datasets. The

datasets can be downloaded directly, without further processing. Only the data from Reddit<sup>1</sup> needs to be fetched through an Application Programming Interface (API). In order to evaluate the data for possible comparatives, two rudimentary keyword filters are implemented. The first filter searches for question words in the beginning of a sentence. The second filter searches for four indicator keywords: *or*, *than*, *vs*, *compare/compared*. All keywords of the second filter need to be found within a question. These statistics are used to get an overall assessment of how many questions and potential comparatives exist in a dataset.

### 3.3.1.1 AOL User Collection

In 2006, the search engine AOL published a large amount of anonymised search requests [67]<sup>2</sup>. The collection consists of 20 million queries requested by 650,000 users over a period of three months. The dataset includes:

- *AnonID* - Anonymous user ID.
- *Query* - Search query issued by the user.
- *QueryTime* - The time at which the query was submitted.
- *ClickURL* - The search result's URL the user clicked on.
- *ItemRank* - The rank of the search result.

The user's identification was replaced by an anonymous ID. The queries were not filtered and no content was removed, hence, explicit language and data is still present in the data. Most of the queries do not include the URL click information. Normalizing the search queries (e.g. lowercasing) results in a set of 10 million unique search queries. Directly after the publication by AOL, the dataset gained massive public attention. Due to the fact that the queries in the dataset are not anonymized, the data still contains personally identifiable information. Individuals could be identified by information from their search history, such as numerical account data or addresses [68].

Table 3.9 shows examples from the AOL dataset. A high amount of the queries in the dataset are navigational queries, similar to the first example in Table 3.9. Also, as to be expected from search engine queries, most queries are very short. The average length of a query is 4.4 words. To have a more reliable prediction of how many comparatives are contained in this dataset, queries with a length shorter than 3 words are excluded from the

---

<sup>1</sup><https://www.reddit.com/>

<sup>2</sup>The data can be found here: <http://www.cim.mcgill.ca/~dudek/206/Logs/AOL-user-ct-collection/>

AnonID	Query	QueryTime	ItemRank	ClickURL
2005	www.weather.com	2006-05-08 01:07:49	1	www.weather.com
543587	gsm vs cdma	2006-05-05 12:12:52	2	www.bsnl.in
1036161	when kids eyes stay dilated more than normal	2006-03-02 22:49:02		

Table 3.9: Examples from the AOL dataset.

evaluation. Furthermore, queries that contain URL-specific parts like “www” or “http” are also excluded from the data. This results in 4.6 million unique queries. An overview of the evaluation statistics can be found in Table 3.16. In this set, at least 29,000 queries are closed questions and 148,000 are open questions. Despite this high number of questions, the indicator keywords show that there might only be a low number of comparative questions. The keyword “or” reaches the highest count of 1,807 samples and “than” has 371 occurrences. The conjunction “or” is expected to have a high false positive rate, this means that even high counts for this keyword might not clearly indicate high amounts of comparatives. Out of 4.6 Million samples, 2,490 (0.0005%) samples are potential comparatives. This low absolute values, the controversial release of the data and also the low average length of the queries are strong arguments against using the AOL data as a source for this thesis.

### 3.3.1.2 Google Natural Questions

The Google Natural Question (GNQ)<sup>3</sup> dataset was published by Google in 2019 as a question answering dataset [69]. The dataset is composed out of queries, issued by real users to Google’s search engine. It consists of 323,000 questions with the following information:

- *Question* - A real user’s question seeking factual information.
- *Wikipedia page* - May or may not contain information to answer the question.
- *Long answer* - A bounding box on the Wikipedia page that contains all information to infer the question.
- *Short answer* - May be one or more entities, yes, no or none.

While building the dataset, several heuristics were used to filter the search engine queries. The goal was to find questions, which seek factual information. For example, queries

<sup>3</sup>The data can be found here: <https://ai.google.com/research/NaturalQuestions>

needed to start with the question words *who*, *when*, *where* or contain multiple entities as well as an adjective, adverb, verb, or determiner. To gain a high complexity for natural language understanding (NLU) tasks, Google focused on queries longer than 8 words. After the pre-processing, the questions were anonymized and aggregated from multiple similar questions. The Wikipedia page that was assigned to a question was determined by running the questions through the Google search engine. Therefore, it may or may not contain the information to answer the question. The answers, generated by human workers, are two fold. A long answer provides a bounding box on the Wikipedia page which contains all information to infer the question. The short answer is composed out of entities from the Wikipedia page, or a simple yes or no. Either one or both of the answer fields can be empty.

Question	Wikipedia page	Long Answer	Short Answer
who plays casey kelso on that 70s show	Luke Wilson	... Wilson also had a role on That '70s Show, as Michael Kelso's older brother Casey Kelso, appearing sporadically from 2002 through 2005.	
when did the battle of palmito ranch happen	Battle of Palmito Ranch	The Battle of Palmito Ranch is considered by some criteria as the final battle of the American Civil War. It was fought May 12 and 13, 1865, on the banks of the Rio Grande ...	May 12 and 13, 1865
lighting reaches a temperature four times greater than the sun's surface	Solar energy		

Table 3.10: Examples from the Google Natural Questions dataset.

Table 3.10 shows examples from the Google Natural Questions dataset. As it can be seen in the third example, not every query is a question. The evaluation of the dataset shows that at least 232,500 samples are sentences starting with a question word. The open questions represent the majority with 225,000 samples. The dataset contains only 2,700 (0.009%) potential comparative questions. These low numbers of comparatives can be explained by Google's intention to provide a dataset containing factoid questions. The average sentence length of 9.2 is fitting good for the purpose of this thesis. An overview of the evaluation statistics can be found in Table 3.16. Even though the data can be considered open-domain and "natural", the strong regulation for factoid questions and the overall low comparative indicator number are arguments against using the GNQ as a data source.



### 3.3.1.3 MSMarco

The large scale MACHine Reading COMprehension (MS MARCO)<sup>4</sup> dataset was published by Microsoft (MS) in 2018 [29]. The open-domain dataset consists of 1,010,916 anonymized questions from Microsoft’s Bing search engine. The dataset contains the following information:

- *Question* - A real user’s question from MS Bing.
- *Passages* - Up to 10 paragraphs that may contain the answer to the question.
- *Answer* - Natural language, human written, answers.
- *Question Type* - Categories are Description, Numeric, Entity, Location and Person.

The questions used for the dataset were automatically collected from the Bing search query logs by a machine-learned classifier. For each question 10 text passages, taken from web documents, were provided. Human annotators were asked to answer the questions only with the help of these passages. If the answer could not be found in one of the passages, the question counted as unanswerable and the answer field was left empty. Some answers were enhanced in a review-and-rewrite process. For example, answers that could not be understood without knowledge of the question were rewritten (see Example 2 in Table 3.11). Furthermore, the type of answer a question was expecting, was annotated by a machine-learned classifier.

Question	Passages	Answer	Question Type
what is a corporation?	Corporation definition, an association of individuals, created by law or under authority of law, having a continuous existence independent of the existences of its members, and powers ...	A corporation is a company or group of people authorized to act as a single entity and recognized as such in law.	Description
albany mn population	Albany, Minnesota, as per 2017 US Census estimate, has a community population of 2,662 people. Albany is located in Stearns County ...	The population of Albany, Minnesota is 2,662. (original answer: 2,662)	Numeric

Table 3.11: Examples from the Microsoft MACHine Reading COMprehension (MS MARCO) dataset. The passages are shortened for this table.

The evaluation of the dataset shows that at least half a million samples are open questions and at least 64,000 are closed questions. The data contains 6,100 potential comparatives, of which 4,900 are built with the conjunction “or”. The average length of a question is

<sup>4</sup>The data can be found here: <https://microsoft.github.io/msmarco/>

6.3 words. Due to the way Microsoft selected the data, it can be assumed that the data is open domain. Taking all these estimates into account, the MS Marco dataset could be an option as a data source. However, it can be expected that there will be a high number of false positives because of the high number of potential comparatives with “or”.

### 3.3.1.4 Quora Datasets

The community QA platform, Quora<sup>5</sup>, provides people with a place to ask questions and get user created answers. The answers are rated by the community to find the answer with the best fit. In the recent years, Quora has released two datasets that contain questions sampled from their platform. Firstly, in 2017, a “Duplicate Question Pairs” dataset was released with the aim to machine classify questions with duplicate content [70]. Secondly, in 2018, the “Insincere Questions” dataset was released [71]. The dataset’s research purpose was to identify questions in which the author is insincere and intends to make a statement instead rather than searching for helpful answers. In the following sections, both datasets are evaluated in terms of their suitability for providing source data for this thesis.

#### *Quora Duplicate Question Pairs dataset*

The Quora “Duplicate Question Pairs” dataset [70] contains 400,000 lines of potential duplicate questions, which were sampled from questions asked on the Quora website. The dataset is structured in the following way:

- *Sample-ID* - A unique ID to identify each sample of the dataset.
- *Questions-IDs* - Each question has a unique identifier within the whole dataset.
- *Question1 & Question 2* - A pair of questions that are potential semantic duplicates of each other.
- *Duplicate* - A binary indicator whether the questions pair is a duplicate.

Each sample of the dataset is built out of two questions that might be semantic duplicates of each other. Different sampling and sanitation methods were combined to create the dataset out of Quora’s data. Because of this, the distribution of questions can not be considered to be representative of the distribution of questions asked on the website [70]. The authors do not elaborate further on which methods were utilized for the sampling. Therefore, it is not clear if a semantic or content-related selection was performed. It is to

---

<sup>5</sup><https://www.quora.com>

be expected that the data might not be open-domain. Moreover, their sampling created a highly unbalanced dataset towards true duplicates. This was fixed by adding negative examples, which were taken from the related questions section of a duplicate question. The topic of the added question was similar to the original, but not semantically equivalent. This way, it was ensured that a true negative sample was added to the dataset. Furthermore, the authors pointed out that the labels contain noise and are not guaranteed to be perfect.

Question 1	Question 2	Duplicate?
What are natural numbers?	What is a least natural number?	No
How do you start a bakery?	How can one start a bakery business?	Yes
Did Ben Affleck shine more than Christian Bale as Batman	No fanboys please, but who was the true batman, Christian Bale or Ben Affleck?	Yes

Table 3.12: Examples from the Quora Duplicate Question Pairs dataset.

The examples in Table 3.12 demonstrate the semantic similarities (see Example 2). The dataset is slightly unbalanced with 255,027 (63%) samples classified as non-duplicate questions. For the research goal of this thesis, a semantic similarity may be ignored as long as the wording of the sentence is not an exact duplicate. In the dataset, questions are reused in different sample pairs. This reduces the total number of questions to 537,933 unique questions. 340,000 of these are open questions and 82,000 are closed questions. The comparative indicator shows that there are 30,900 potential comparative questions in the dataset. A high amount of 73% of these potential comparatives are generated with the conjunction “or”. The average sentence length of 11.6 words fits the requirements. Quora’s restrictive license to republish the data and the likelihood that the data is not open-domain, are arguments against using it as a data source. Furthermore, the dataset does not contain additional information and the provided question identifiers are not equal to the ones used on the Quora platform. Therefore, there is no easy way to collect the answers to these questions.

#### *Quora Insincere dataset*

The Quora “Insincere Question” dataset has a total of 1.3 million labelled samples (training data). Additionally, for an online competition on the data-science platform Kaggle<sup>6</sup>, 375,000 unlabelled test samples were released. The training dataset contains the following information:

<sup>6</sup><https://www.kaggle.com/>

- *Sample-ID* - A unique ID to identify each sample of the dataset.
- *Question* - A question that is potentially insincere.
- *Label* - A binary indicator that shows whether the question is insincere.

The dataset has a very simple structure. It provides a question with a label. The label indicates if a question is sincere or not. Sincere questions are questions that look for helpful answers. For insincere questions the authors defined a few characteristics. An insincere question may have a non-neutral tone, for example, meant to imply a statement about a group of people. Furthermore, a insincere question may be disparaging or inflammatory, not be grounded in reality or contain sexual content for shock value. Similar to the “Duplicate Question Pairs” dataset, the authors do not guarantee that the distribution in the dataset is representative for the distribution of questions asked on Quora. Moreover, they advise again that the labels may contain noise and are not guaranteed to be perfect.

Question	Insincere?
What are the theories in critical thinking?	No
Has the United States become the largest dictatorship in the world?	Yes
Can acid absorb heat faster than water?	No
Do you think like me that mother Russia is better than the pussies of America?	Yes

Table 3.13: Examples from the Quora Insincere Question dataset.

Table 3.13 shows examples for sincere and insincere questions. The third and fourth example are comparative examples. The dataset is very unbalanced in favour of the sincere questions. Only 6% of the samples are labelled as insincere. The dataset has 265,000 closed and 829,000 open questions. The filtering indicates that 86,300 samples are potential comparatives. Out of these, 66,000 (76%) samples contain the word “or”. Similar to the “Duplicate Question Pairs” dataset, this is a very high number, but the absolute number of comparisons generated with the other filter words is higher than in any reviewed dataset so far. In total, there are 20,247 potential comparative samples that are not generated with “or”. The average sentence length in the dataset is 12.5 words. This fits with the requirements and is also close to the value of the “Duplicate Question Pairs” dataset. This indicates that the filtering and the pre-processing for both datasets was executed in a similar way. The dataset was published under the same restrictive licence as the “Duplicate Question Pairs” dataset and also has the same constraints, as no additional data is provided. Therefore, the dataset does not meet all the requirements.

### 3.3.1.5 Yahoo Answers

From Yahoo's data platform Webscope<sup>7</sup>, two datasets are evaluated for this thesis. The first one is a large log of search queries from the Yahoo! search engine (Webscope L18) [72]. The second one is a collection of questions and answers from the community QA platform Yahoo! Answers (Webscope L6) [73]. Both datasets are publicly available for research. After an initial manual review, the search query logs can be ruled out as a data source for this research. Before publication, Yahoo anonymized the data completely. By then, the textual search queries were converted into an 8-character alphanumeric string, which makes them useless for building the comparative question dataset. The data from the QA platform looks more promising. The dataset contains a total of 4.48 million questions with their answers. The data was collected from the Yahoo! Answers platform as of 2007. The dataset comes with an extensive amount of information and has a total size of 12GB data. The following list only shows the most important features:

- *Question subject* - A question taken from the QA platform Yahoo! Answers.
- *Question content* - A longer text providing context or specifying the question. This field is optional for each question.
- *Best answer* - The best answer is selected by the asker or by vote of the users. This field is mandatory.
- *All answers* - A list of all answers to the question. This field is mandatory.
- *Taxonomy* - The question is classified into a hierarchical taxonomy using a main category (e.g. "Travel"), a category (e.g. "China") and a sub-category (e.g. "Asia Pacific"). The taxonomy elements are optional.

The sample of the dataset at least provides a question, the best answer to the question and a list of other answers. The best answer is either picked by the asker or by the community, in case the asker does not choose one. For the list of answers, no ranking is provided and it is not clear if they are sorted by a ranking system. Optional to these fields, the dataset contains a content field in which a question can be further elaborated by the asker. A manual data review shows that the subject field is not always used to formulate the question. In some cases, it serves as a headline and the actual question is asked in the content field. Further to these fields, the sample might provide a taxonomy field. A question is classified by a not clearly specified question taxonomy with three hierarchical elements. Table 3.14 shows example samples taken from the dataset. The second example in Table 3.14 is a comparative question with an factoid answer. The third example demonstrates that in some cases the question is not written in the subject field. Furthermore, the text

---

<sup>7</sup><http://webscope.sandbox.yahoo.com>

in the subject field and the content field can also be a duplicate (see Example 4 in Table 3.14). For the evaluation of the dataset, both fields are utilized in the case that the content is not a duplicate. This strategy results in a total of 7.03 million samples for the evaluation. The evaluation shows that the dataset contains at least 922,000 closed question and 1.86 million open questions. The comparative indicator lists 265,000 possible comparative samples. At least 32,000 of these do not use the conjunction “or”. The samples have an average length of 21.2 words, which is a bit high. An evaluation that only uses the question subject fields shows 184,000 comparatives and an average word count of 10.6 words per sample. The dataset can be considered open-domain, as it is a copy of the QA platform’s data, without the application of any filters or restrictions.

### 3.3.1.6 Yelp

The online platform Yelp<sup>8</sup> provides its users with the possibility to review local businesses. In 2019, Yelp published a large-scale dataset with 8 million reviews from over 200,000 businesses in 10 metropolitan areas [74].

- *Identifiers* - Identifiers for the user, the business and the review.
- *Review text* - The review text written by the users of the platform.
- *Vote counters* - Counters for votes the users can place on a review, for example, “stars” or “useful”.

The dataset contains the user written review text, several identifiers to link the reviews to businesses and users and a number of vote counters. The reviews have an average word count of 111.5 words. This high word count can be expected as reviews are mainly continuous text in which people, for example, describe their experiences with a restaurant. The complete dataset contains only 165,000 closed questions and 94,000 open questions. 78,000 of the questions might be a comparative with a high amount of 55,000 samples containing the word “or”. Even though the dataset is available for the public, it enforces very strict terms of use. The usage for academical research is allowed, but it is forbidden to publish any part of the original dataset. Therefore, no examples from this dataset can be listed here. This strict policy, the closed-domain setting with reviews, as well as the high average word count rule out the Yelp data as a source for this thesis.

---

<sup>8</sup><https://www.yelp.com>

Subject	Content	Best answer	Taxonomy
What is the best off-road motorcycle trail ?	long-distance trail throughout CA	i hear that the mojave road is amazing!	Sports Hunting Outdoor Recreation
what has more caffeine? a double latte or a large coffee?	choosing between a double latte (or similar) and a 16oz cup of drip coffee, which would have more caffeine?	Based on this (LINK) i would say that a large coffe has about 80-135mg of caffeine, but a double latte (which has 2 espresso shots) has about 200mg+	Food & Drink Non-Alcoholic Drinks
Vacation rentals in the Turks and Caicos	We are considering renting a house in the Turks and Caicos... any recommendations of which islands might be best, and good places to rent from?	I like Providenciales best. Beautiful beaches are scattered on all sides of Providenciales, the most spectacular of which is a 12 mile stretch located on Grace Bay, which is protected by a healthy barrier reef. Provo has an 18 hole golf course, a casino, shopping centres, three marinas, a growing number of bars and excellent restaurants. Provo is also a divers' and water lovers' paradise.	Travel Turks & Caicos Caribbean
What's the best way to heat up a cold hamburger?	What's the best way to heat up a cold hamburger?	If you must eat a heated hamburger then I suggest the follow: First, scrape the ketchup, mayo and any other sauce that is on the hamburger and put it a side. Also take the vegetables out. Second, pre-heat a steak pan; put the hamburger separate form the bun on the hot pan. Third, heat up both parts of the bun near the hamburger on the same hot pan (you can get some souce from the hamburger on it). Fourth, you can either put back the original ketchup, mayo and the vegetables or, better yet, get new ones from what you have at home. Bon Appetite	Food & Drink Cooking & Recipes

Table 3.14: Example questions taken from the Yahoo! Answers dataset.

### 3.3.1.7 Reddit data

The last data source is Reddit<sup>9</sup>. Reddit is a large group of online forums, which registered users can use to talk about nearly everything. People are able to publish text (posts), upload images and share links, as well as vote and comment on other users content. Specialized forums are called *subreddits* and are linked with *r/“topic”*, for example, *r/science*. People can subscribe to a subreddit to receive news about the latest uploads. The top three most subscribed subreddits are *r/funny* (31 million users), *r/AskReddit* (28 million users) and *r/gaming*(26 million users). The subreddit *r/AskReddit* is according to its self description “the place to ask and answer thought-provoking questions” [75]. When publishing a post in this subreddit, people have to follow a few simple rules. There are two rules that regulate the subject of the question: “No personal or professional advice requests” and “No loaded questions”<sup>10</sup>. In order to allow a open-ended discussion, another requirement is that the question needs to be an open question. Since March 2019, every single day more than 10,000 new posts were created in *r/AskReddit* [76]. These enormous numbers give an idea of how big Reddit is in terms of user-generated content, which could be used in a dataset. However, there is no ready-to-use dataset available. On the one hand, this means that the data can only be manually collected through an Application Programming Interface (API). On the other hand, it allows to create a unique dataset fitting the needs of the task. To evaluate the usefulness of Reddit as a source for comparative questions, a set of posts from *r/AskReddit* is fetched trough the API. A detailed explanation of the programming and the utilized API can be found in Section 3.3.2. A total of 437,900 posts are collected with the following informations:

- *Identifier* - Identifies the post and is also part of the URL.
- *Title* - The user written text or question.
- *URL* - Address linking directly to the post.
- *Author* - Username of the Reddit user that created the post.
- *Subreddit* - The name of the subreddit in which the post is published. In this case, always *r/AskReddit*.
- *Creation Date* - The data of the publication. For this set all dates are between 31.07.-28.08.2019.

The dataset provides the post title, which is the users question. In the *AskReddit* subreddit only the title is utilized and the body of a post remains empty. For future work

---

<sup>9</sup><https://www.reddit.com/>

<sup>10</sup>No questions including an opinion, bias, or that lead respondents towards expressing a specific opinion. [75]



the identifier, the URL and the author are part of the dataset. The data was published between the 31. July and the 27. August in 2019. For this evaluation the data is not filtered and not preprocessed.

---

Post title
What was your worst day ever?
Why do Americans write the date month/day/year and not day/month/year?
Let's assume they could go on a journey through time. In what time would you travel and why?
Which TV show is better than Friends?
Does modern life give us more freedom or less freedom than in the past?
What TV shows pilot is most different compared to the rest of the series?
Why are robot lawn movers so outdated compared to robot vacuum cleaners?

---

Table 3.15: Examples from the Reddit forum r/AskReddit.

Table 3.15 shows examples of questions taken from the dataset. The last three questions contain comparatives and superlatives. The dataset has 179,000 sentences starting with an open-question word and 28,000 sentences begin with a closed-question word. The comparative indicator shows 22,000 possible comparatives with a high number of sentences built with the word “or”. Only 2,900 comparative questions are generated with the other indicator words. The majority of these (2,500 sentences) is using the word “than”. The average word count per sentence is 13.9 and, therefore, it fits the requirements. Even though the comparative numbers are low, Reddit provides huge amounts of possible source data. The subreddit r/AskReddit alone has 23.5 million<sup>11</sup> posts. Furthermore, Reddit has various smaller and more specialized question-centred forums. Trough the API it is possible to extract more information, for example, a list of answers and their user-provided rank.

### 3.3.1.8 Conclusion of the dataset evaluation

Table 3.16 displays a summary of the statistics on each dataset. The evaluation shows that the Yahoo data and the data from Reddit is the best fitting one for the task of this thesis. Both sources provide open-domain data and are available to the public. For both datasets the percentage of possible comparatives is high, compared to datasets like AOL, GNQ or MSMarco. Furthermore, both sources provide a high amount of data. The Yahoo dataset provides up to 7 million samples and the Reddit dataset provides 400,000 samples. For the Yahoo data it is the maximum number of samples, since more data is not

<sup>11</sup>As of 15.06.2020 via <https://api.pushshift.io/reddit/search/submission/?subreddit=askreddit&metadata=true>

## CHAPTER 3. CREATION OF COMPARATIVE QUESTION DATASETS

Name	Dataset Size	Questions (Open/Closed)	Comparative indicator	Average Sentence Length	Open-domain	Public
AOL	4,600,000	148,000/29,000	2,500 (0.0005%)	4.4	Yes	Yes
GNQ	323,000	225,000/7,500	2,700 (0.009%)	9.2	Partly	Yes
MSMarco	1,010,000	524,000/64,020	6,100 (0.0075%)	6.3	Yes	Yes
Quora: Insincere	1,225,000	829,000/265,000	86,300 (0.070%)	12.5	Unknown	Partly
Quora: Duplicates	537,000	340,000/82,000	30,900 (0.064%)	11.6	Unknown	Partly
Reddit: r/AskReddit	437,900	179,000/28,000	22,000 (0.050%)	13.9	Yes	Yes
Yahoo	7,030,000	1,860,000/922,000	265,600 (0.038%)	21.3	Yes	Yes
Yelp	6,686,000	94,000/165,000	78,700 (0.011%)	111.5	No	Partly

Table 3.16: Statistics on the data sources. The table shows the dataset size, as well as general statistics and information about the dataset. The comparative indicator provides a rough measure on how many comparatives are contained in a dataset.

available. In the case of Reddit, 400,000 samples is just the exemplary evaluation. More data can be collected from the AskReddit subreddit and other similar question subreddits. Additionally, both datasets provide answers to the questions. The answers can be used in future research on information retrieval and question answering. Even though the Yelp dataset has a high count of possible comparatives, the data is ruled out for this task. The review data’s average word count is too high and it is closed-domain data by its nature. Moreover, the terms of use, enforced by Yelp, are very restrictive. The two Quora datasets can be an alternative to the Reddit and Yahoo data. Both Quora datasets have a high enough number of possible comparatives, a fitting average word count and a high number of source samples. However, it is not completely clear if the data is open-domain, since some undisclosed filtering and pre-processing was performed by Quora. Moreover, the Quora datasets provide no further information than the question itself. This disadvantage might not be relevant for this task, but could be obstructive for future work with the data. The following section will work with the Reddit and Yahoo data and briefly include the Quora datasets to have a backup. First, the datasets will be preprocessed and then, the comparatives will be filtered out with the help of the taxonomy described in Section 3.2.

### 3.3.2 Gathering and pre-processing of source data

In this section, the gathering and pre-processing of the data is described. These steps are described for both data sources, Yahoo and Reddit, and briefly for the backup source Quora, which have been chosen to fit the task requirements in Section 3.3.1.8.

Gathering the data is fairly easy for Yahoo and Quora, while Reddit needs programming. In the case of the Yahoo data, an application on the Yahoo Webscope data platform is necessary to gain access to the data. After the access is granted, the datasets can be downloaded from the website and are ready to be used. The Quora data can be downloaded without any application from the Quora website. Solely the Reddit data needs to be fetched through an API. To make use of the API, a Python<sup>12</sup> script was implemented. The script is described in the following paragraph.

Reddit provides a RESTful<sup>13</sup> Application Programming Interface<sup>14</sup> to access different endpoints in order to fetch data, ranging from personal account details to subreddit posts and comments. To use the API, the authentication with a Reddit user agent is required. An easy way to utilize the API is to use the Python Reddit API Wrapper (PRAW)<sup>15</sup>. The wrapper provides methods to create an instance of PRAW with user credentials and then interact with Reddit, for example, by creating a comment or retrieving a submission. Even though the API is granting full access to Reddit and the user profile, the major drawback is that clients can only make up to 60 requests per minute [77]. A solution is provided by the website Pushshift<sup>16</sup>. Pushshift is a big-data storage and analytics project created by the Reddit user Jason Baumgartner. Pushshift is a copy of all Reddit's comments and submissions and provides access to all these Reddit objects via its own Pushshift Python API Wrapper (PSAW) [78, 79]. In contrast to the PRAW API, PSAW does not limit the requests per minute and supports the filtering and the sorting of comments, submissions and subreddits. The downside of Pushshift is the point on which the data is copied. The copy procedure into Pushshift happens directly after a data object (e.g. a comment) is published. The data objects are not updated at any later point. Therefore, the data does not include any changes that are made after publishing and may not reflect the version that is visible on Reddit. Most importantly, this includes edits of a submission's title or body text and, for example, the rating scores of comments. To overcome the restrictions of both APIs, the program code shown in Listing 3.1 uses the PSAW API to collect submissions (posts) from a dedicated subreddit after a specific start date. If activated, the

---

<sup>12</sup>Python is an interpreted, object-oriented, high-level programming language.

<sup>13</sup>A web service with a Representational State Transfer (REST) software architecture style.

<sup>14</sup><https://www.reddit.com/dev/api>

<sup>15</sup><https://github.com/praw-dev/praw>

<sup>16</sup><https://pushshift.io/>

script collects the three highest ranked answers to a submission from the live Reddit data via the PRAW API. For the data gathering in this thesis, the collection of the answers is not activated, since it would take 1 second per submission to retrieve this information. With millions of available data samples, this practise would consume too much time. For future research, a retrieval of the ranked answers is possible, if the samples are limited to positive identified comparatives. This will result in only a few thousand API calls and it will be a matter of hours for collecting the information.

```

1 | #Reddit data scraper (redditScraper.py)
2 | #PRAW user setup
3 | reddit = praw.Reddit(user_agent='user account')
4 | #Pushshift API Wrapper setup
5 | api = PushshiftAPI()
6 | #Subreddit to scrape provided as commandline argument
7 | subredditname = sys.argv[1]
8 | #Configure collection of best ranked answers
9 | downloadAnswers = False
10 |
11 | with open(outfile, 'w') as tsvfile:
12 |     writer = csv.DictWriter(tsvfile, fieldnames=fieldnames, delimiter='\t')
13 |     #Define the earliest date for the collection
14 |     start_epoch = int(dt.datetime(2011, 7, 27).timestamp())
15 |     #Retrieve submissions from the subreddit
16 |     gen = api.search_submissions(after=start_epoch, subreddit=subredditname,)
17 |     for rPost in gen:
18 |
19 |         #If activated, collect the best 3 comments
20 |         topCommentsList = []
21 |         if downloadAnswers and rPost['num_comments'] > 0:
22 |             submission.comment_sort = 'best'
23 |             submission.comment_limit = 3
24 |             for comment in submission.comments:
25 |                 if isinstance(comment, MoreComments):
26 |                     continue #continue if comment is not topLevel
27 |                 topCommentsList.append([comment.id, comment.body, comment.score])
28 |
29 |         #Write blanks if no comment was fetched
30 |         if len(topCommentsList) < 3:
31 |             topCommentsList.append(['-', '-', '-'])
32 |
33 |         #Write submission fields directly as a new line in the document after retrieval
34 |         writer.writerow(retrievedFields)

```

Listing 3.1: Pseudocode of the Reddit scraper script with an optional collection of the highest ranked answers.

The Python pseudocode shows the general structure of the Reddit data collector. The user agent, required for using the PRAW API, is set up with a separate configuration file and referenced when initializing the API (see Listing 3.1 Line 3). The Pushshift API can be initialized without a user agent (Line 5). The name of the subreddit from which submissions are collected is provided via a command line parameter (Line 7). To keep the memory footprint small, the retrieved posts from Reddit are directly processed and written as data lines in a tab-separated file (lines 11-12 + 34). For simplicity, the retrieved fields are not listed in the pseudocode. In total, for each post 29 values are saved, including the submission ID, the title, various metadata and the best answers. Table 3.17 shows the list of saved data fields. The Pushshift API fetches posts sorted from the newest post to

the oldest one. It stops at the start date, which can be configured in Line 14. If activated, the script collects the three best comments to a submission (Lines 20-27). Only top-level comments (comments directly replying to the submission) are collected. If less than three answers are found or if the feature is deactivated, the fields are filled with a hyphen (Lines 30-31). The pseudocode neither displays the code for outputs to the user, sanity checks of the data for empty submissions nor the feature to break after a certain amount of collected data.

id	created	stickied	best_answer_2_body
title	is_original_content	score	best_answer_2_score
selftext	is_video	total_awards_received	best_answer_3_id
url	is_self	removed_by	best_answer_3_body
author	num_comments	best_answer_1_id	best_answer_3_score
author_fullname	num_crossposts	best_answer_1_body	
subreddit	over_18	best_answer_1_score	
subreddit_id	pinned	best_answer_2_id	

Table 3.17: List of data fields stored for each submission from Reddit.

The script allows to collect data from any subreddit. In Section 3.3.1, only the largest question-centred subreddit, *r/AskReddit*, was evaluated for possible comparatives. Nevertheless, Reddit has many more subreddits, which are centred around asking and answering questions. All of these are smaller than *r/AskReddit* in terms of daily submissions, however some of these subreddits have multiple hundred thousands or even millions of submissions. As no predefined list or dictionary of subreddits with high amounts of questions exists, a list of ten subreddits was collected. By manual evaluation, through reading the submissions, subreddits that seemed to be centred on asking questions were chosen. The list below provides the subreddit names and a short description for each subreddit:

- *r/AskReddit* - The biggest subreddit for questions (evaluated in Section 3.3.1).
- *r/ask* - For “thought-provoking” open questions. No technical support, legal or medical advice is allowed.
- *r/AskMen* - Open questions typically answered by men. The title must contain the question. No medical advice is allowed.
- *r/AskWomen* - Open questions typically answered by women. The title must contain the question. No exclusion of minorities.

- *r/AskEngineers* - Questions must be about engineering. The title must contain the question.
- *r/askscience* - Questions about science. No medical advice is allowed.
- *r/explainlikeimfive* - Subreddit to provide easily understandable explanations for complex topics. Questions must seek objective explanations.
- *r/NoStupidQuestions* - Open to all serious questions. No joke questions, no illegal or disturbing question subjects. No medical advice is allowed.
- *r/Questions* - Open to any type of question without specific rules.
- *r/technology* - Dedicated to news articles about the creation and use of technology.
- *r/techsupport* - For people that seek help on technical issues.

The subreddits vary in size and topic. For example, *r/explainlikeimfive* has a total of 1.4 million submissions and has the aim to provide easily understandable explanations for complex topics. In comparison to *r/explainlikeimfive*, the subreddit *r/Questions* is completely open to any type of questions, including biased or “stupid” questions. The list also contains subreddits which are limited to a certain domain. For example, the subreddit *r/AskWomen* is supposed to hold questions that are directed at women and that should be answered by women. The subreddits *r/technology* and *r/techsupport* clearly define the topic technology as a domain. For each subreddit in the list, a data-sample of one month of submissions is collected with the Reddit collector script. Section 3.3.3 will evaluate the number of possible comparatives in each of these subreddit datasets.

With the successful collection of Reddit data through the script, all datasets are available as single data files. Nevertheless, the datasets come as various data structures and with different data fields. To be able to load the data, each dataset needs its own reader function and some simple pre-processing. These reader functions are part of the script to filter out potential comparatives (see Section 3.3.3).

```
1| ##### Reddit Dataset #####
2| if(datatype == "REDDIT"):
3|     querys = {}
4|     with open(infile, 'r', newline='', encoding='utf-8') as csvfile:
5|         reader = csv.DictReader(csvfile, delimiter='\t', quoting=csv.QUOTE_ALL)
6|         for row in reader:
7|             querys[row['id']] = row['title']
8|     filterQuerys(querys, outfile)
```

Listing 3.2: Pseudocode of the Reddit data reader script.

Loading the Reddit data can be done by simply parsing each line of the tab-separated file that was created by the Reddit collector script. Listing 3.2 shows the pseudocode for this. Only the submission's title (containing the question) and the submission's ID (identifying the exact submission later) are necessary at this point. Both values are loaded into a Python dictionary with the submission ID as the dictionary key (Lines 3-7). The Quora data is also provided in a value separated format and can be loaded in a similar way as the Reddit data. Only the field names and the delimiter (tab and comma) are different. For the Quora "Duplicate Question Pairs" dataset, the questions are preprocessed to gather only unique questions (see Section 3.3.1 for more details). The Yahoo data requires the highest amount in pre-processing because the data comes as a complex structured XML<sup>17</sup> file. Listing 3.3 shows the code that extracts the questions and the question's IDs and writes them into a Python dictionary.

```

1| ##### YAHOO Dataset #####
2| if(datatype == "YAHOO"):
3|     querys = {}
4|     # Get an iterable
5|     context = ET.iterparse(infile, events=("start", "end"))
6|     is_first = True
7|     for event, elem in context:
8|         #Get the root element
9|         if is_first:
10|             root = elem
11|             is_first = False
12|         #Find the element that holds a question (vespaadd element)
13|         if event == "end" and elem.tag == "vespaadd":
14|             subject = elem[0][1].text
15|             if len(subject) > 0:
16|                 #Get the ID and the text
17|                 querys[elem[0][0].text] = elem[0][1].text
18|                 if elem[0][2].tag == "content":
19|                     if elem[0][1].text != elem[0][2].text:
20|                         querys[elem[0][0].text+'c'] = elem[0][2].text
21|                 #Clear elements in order to not run out of memory
22|                 root.clear()
23| filterQuerys(querys, outfile)

```

Listing 3.3: Pseudocode of the Yahoo data reader script.

### 3.3.3 Filtering for comparative questions

After describing the evaluation of data sources and the gathering of data in the Sections 3.3.1 and 3.3.2, this section describes the process of the actual data mining for comparative questions. Regarding the data mining, the taxonomy and the linguistic creation rules for comparatives from Section 3.1 are implemented into program code. The program is written as a Python script and it extends the data-reader functions explained in Section 3.3.2.

<sup>17</sup>Extensible Markup Language (XML) is a markup language that defines a human and machine readable file format.

For this thesis, the three-digit classes starting with 1 or 2 (1xx or 2xx) from the taxonomy are of interest. Sentences categorized in these classes start with a question word and are comparative. The goal of the filtering is to narrow down the amount of source data to a level that is reasonable to process in a human annotation task. For example, the comparative indicator for the Yahoo dataset (see Section 3.3.1) shows that only 0.038% of the samples in the dataset might be comparative questions. To produce a final dataset with 5,000 positive comparative questions, human annotators would need to annotate around 131,000 samples. Therefore, the filtering will help to reduce the amount of annotation data to a more reasonable level. The target is to annotate twice the amount of positive comparative questions. To match the goal of collecting 5,000 comparative questions, the annotation dataset should be at least 10,000 samples big. It also means that the filtered data needs to have a precision of 50% on comparative questions. While keeping the precision at a medium level, the recall should be high enough, to include as many relevant comparative questions in the filtered data as possible.

The Python script implements this concept by means of a decision pipeline. Every sentence runs through the decision pipeline to categorize the text into one of the taxonomy's classes. Figure 3.4 shows the decision pipeline on an abstract level. The first step of the pipeline performs necessary pre-processing steps on each sentence. For example, all words are converted to lowercase for the following analysis. The second step is a token counter, which sorts out too short or too long sentences. Sentences that are not sorted out reach the third step of the pipeline, which determines whether the text is a question simply by looking up if the first word is a question word. The last two steps sort the text into one of the comparative categories. The category is determined by searching for keywords in the text. For example, the rule filter searches the text for “than” or “as”. If the text contains one of these, they are classified as comparative. The same rules apply to the phrase filter: the script will only check if the text contains one of the phrases from Table 3.5. For these steps there are no rules implemented, for example, regarding the position of the keyword in a sentence or regarding the connection of other words to the keyword. Only the determination of the subtype requires the evaluation of neighbouring words. In the case that the rule filter finds a “than”, it evaluates if the sentence contains “more” or “less”. If this is true for any point of the sentence, the word previous to “than” will be evaluated to check if it belongs to the group of adjectives or adverbs. This last step finalizes the classification. In case the last two filters of the pipeline do not classify the text, it is ruled as not comparative. The inaccuracies and relaxations in the pipeline's filters allow variations in the language and the achievement of a possible high recall on comparative questions. Although these inaccuracies are intentional, they come at the price of having the possibility of a high amount of non-comparatives in the filtered data. For example, the text “Why is it raining? I like sun more than rain” will be tagged as comparative



question by the script. The reason is that the text starts with a question word (Why - open question) and contains the keyword “than” in the second sentence. Furthermore, the second sentence is a statement and there is no adjective or adverb between “more” and “than”. The filtering will finalize with the class 131 for unknown comparative subtypes.

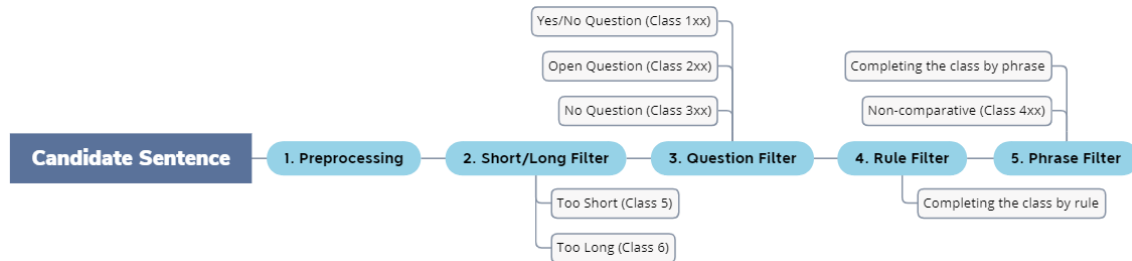


Figure 3.4: The figure shows the decision pipeline for comparative question filtering, which was implemented into code.

To get an initial idea of how good the script performs, data from the Yahoo dataset was fed through the pipeline. The classified results were manually evaluated. This was done by finding wrong categorized samples in the filtered data and then trying to confirm the existence of a pattern for the mistake. For example, the sentence “what about technique, why is your approach better than anyone else?” makes a comparison against a non-comparable entity (anyone). As these kind of sentences are not allowed in the final dataset, an exclusion rule is formulated. For this case, all sentences in which the word “than” is followed by an indefinite pronoun should be excluded from the comparative classes. This manual evaluation resulted in 13 exclusion rules (see Table 3.18). Furthermore, every phrase from Table 3.5 was manually evaluated for its suitability to classify as a comparative phrase word. It was counted how many appearances out of the evaluated sentences with the phrase are comparative (see Table 3.19). For example, sentences containing the phrase “against” are comparative in 0 out of 200 evaluated sentences (e.g. Are you against ... / Is it against god / law / someone ... ). The phrase with the most controversial ratio was “the same” with 41 comparatives out of 200 evaluated sentences. 39 of these 41 samples used “and” as a connection word in the sentence, for example, “is lacoste and crocodile the same?”. To solve this problem, the sole phrase “the same” was excluded from the list of phrases and a special rule was added to the script searching for the word “and” at any position of the sentence when “the same” is found. All 23 exclusion rules were implemented in the script after this evaluation.

With the exclusion rules in place, the source data was filtered again. This time, large amounts of samples from the Yahoo dataset, from both Quora datasets and from Reddit r/AskReddit were filtered. For each of the filtered datasets, 200 samples were taken and manually classified into three classes. One class (Class 0) for non-comparative questions

#	Rule	Target class	Example / Notes
1.	than + personal pronoun (object form)	by rules	“is it okay to date someone nine years younger than me?”
2.	as/so + as + personal pronoun (object form)	by rules	“is there anyone as pathetic as me?”
3.	than + intensive pronoun	by rules	“I am looking for someone who is somewhat younger than myself and loves to laugh?”
4.	as/so + as intensive pronoun	by rules	“is it possible for a human to develop artificial intelligence as good as himself?”
5.	than + indefinite pronoun	by rules	“what about technique , why is your approach better than anyone else?”
6.	as/so + as + indefinite pronoun	by rules	“is iming as good as everyone says it is?”
7.	word distance (so/as) > 5 AND so before as	by rules	For queries where “as” is farther away than 5 words from “so”
8.	word distance (as/as) > 5 AND so before as	by rules	For queries where the first “as” is farther away than 5 words from the second “as”
9.	or + Q-Word	141/241	“should cell phones and pdas be allowed in school? why or why not?”
10.	ADJ or ADJ	141/241	“is chocolate good or bad for health?”
11.	ADV or ADV	141/241	“what dates will be best to travel to colorado in the spring time, late may or early june?”
12.	or + preposition	141/241	“are you for or against gay marriage?”
13.	or + pronoun	141/241	“is it harmful for her or her partner?”

Table 3.18: The table shows the exclusion rules that were established after a manual evaluation of the filtered data.

and two classes (Class 1 and Class 2) for comparative questions. Class 1 is considered a “normal comparative”. Class 2 is also comparative, but might be hard to answer, hard to detect for a machine classification system or even hard to annotate for a human reader. Samples were tagged with class two in order to identify harder classification cases for later parts of this thesis, like the evaluation of human workers. Table 3.20 shows the distribution of the classes in the manual annotated data. It can be seen that the Yahoo and the Quora Duplicates datasets come near to the defined precision of 50% on comparative questions. Reddit performed way worse than expected from the analysis results in Section 3.3.1. For all filtered datasets, it can be seen that the sentences with the phrase “or” have a high rate of false positives. This was already expected after the evaluation of the data sources in Section 3.3.1. To achieve a higher precision on sentences with “or”, more exclusion rules were defined after this manual annotation. In total, six new exclusion rules were

#	Rule	Comparative / Evaluated	Examples / Notes
1.	against	0/200	“are you against / is it against (god/law/someone)”
2.	ahead of	0/6	“ahead of (their time / my classmate / me)”
3.	beside/besides	1/100	Mainly open groups and personal pronouns compared with entities, e.g., “what is the best client besides azureus”
4.	close to	0/50	Used only for directions / locational
5.	just like	3/50	Non-comparative example: “Is anyone sad and lonely just like me?”
6.	like	0/100	Mainly used to express “to like something/somebody” e.g. “Do you like ponies?”
7.	near	0/100	Mainly directional or time related e.g. “what is the cheapest parking lot near lax?”
8.	next to	1/50	Non-comparative example: “how do i get an icon next to the web address in my browser?”
9.	the same + and	41/200	Excluding queries without “and” in the sentence
10.	&nbsp;vs	-	It specifies the use of “vs” better by requiring a blank space in front of “vs”, e.g, the word “Tvs” is no longer matched

Table 3.19: The table shows the phrases that were excluded after a manual evaluation of the ratio of comparative to non-comparative sentences.

added (see Table 3.21). Four rules were formulated in connection with “or”. The first rule changed the program structure in order to search for all phrases first and then separately check for “or” afterwards. Rules 2-4 extended the limitations for “or” in combination with other words. The second one omit sentences in which “or” is used after punctuation, for example, “Are all roses red? Or is there another color?”. The third rule excludes sentences in which “or” is used and followed by a dedicated list of words, for example, in the sentence “Are roses red or not?”. The fourth rule is restricting the use of “or” followed by numbers. For example, “do you own a 2005 or 2006 honda odyssey?” is omitted by this rule. Numbers followed by “or” are explicitly not excluded because a lot of products use numbers in their name and it is not intended to exclude these. For example, “should I buy an xbox 360 or wait for the ps 3?” is a valid comparison and, therefore, kept in the dataset. Furthermore, two more phrases were removed from the keyword list because they mainly produced false positives.

As a next step, the filter script was run again with the new rules implemented. From the filtered data output the same samples were taken for a re-evaluation. Table 3.22 shows

Data source	Overall (%)			Only “or” (%)		
	Class 1	Class 2	Precision	Class 1	Class 2	Precision
Yahoo	37	11	48	18	13	31
r/AskReddit	7	3	10	5	1	6
Quora Duplicates	34	15	48	15	5	20
Quora Insincere	28	11	39	9	9	18

Table 3.20: Class distributions after the manual classification of data from the datasets. Precision is measured for all true positive comparative questions. (comparative = Class 1; hard comparatives = Class 2)

the distribution after annotating the samples again. The figures in brackets denote the change to the first annotations from Table 3.20. For all data sources the overall precision improves between 2-9%. Taking a closer look, the precision within the samples containing an “or” also improved for all data sources except for “Quora Insincere Questions”. Furthermore, it can be observed that the number of true positive samples was reduced by the use of the new rules. This is a trade-off which can be accepted because the absolute numbers for non-comparatives with “or” is significantly reduced. For the Yahoo dataset it was reduced from 72 to 43 samples. This is a reduction of 29 non-comparative samples, while only 9 true positive comparative samples were excluded through these measures.

#	Rule	Target class	Example / Notes
1.	Search list of phrase before “or”	by phrases	-
2.	PUNCTUATION or	141/241	To exclude all sentences where or is used after punctuation, e.g., in sentence beginnings
3.	or + [not, no, so, yes, something, against, if, like, is, do]	141/241	“Are roses red or not?”
4.	NUMBERS + or	141/241	“do you own a 2005 or 2006 honda odyssey?”
5.	between	by phrases	“how do you manage time between work and study?”
6.	the same time	by phrases	It matches trough “the same”

Table 3.21: Exclusion rules formed after the manual classification of samples from the source datasets.

Data source	Overall (%)			Only “or” (%)		
	Class 1	Class 2	Precision	Class 1	Class 2	Precision
Yahoo	43 (+6)	11 (0)	54 (+6)	24 (+6)	13 (0)	37 (+6)
r/AskReddit	9 (+2)	4 (+1)	13(+3)	7 (+2)	1 (0)	8 (+2)
Quora Duplicates	42(+12)	15(0)	57(+11)	23(+8)	8(+3)	31(+11)
Quora Insincere	29 (+1)	11 (0)	41 (+2)	11 (+2)	6 (-3)	17 (-1)

Table 3.22: Class distributions after the manual classification of the datasets with the implemented rules from Table 3.21. Precision is measured for all true positive comparative questions. The figures in brackets denote the change to the first annotations from Table 3.20. (comparative = Class 1; hard comparatives = Class 2)

### 3.3.4 Data mining results

In this section, all data sources are filtered with the script from the previous section and then, for each data source a manual annotation and a evaluation is performed. Throughout this section, statistics and estimations for all data sources are presented and, at the end of the section, it is finally decided which data sources will be utilized for the following human annotation tasks.

The Python filtering script from Section 3.3.3 is used to filter the data from all 14 sources. For computational simplicity, the sample count was reduced for the filtering. For r/AskReddit, the dataset containing submissions from August 2019 (see Section 3.3.1) was utilized. In the case of the other 10 subreddits, the submissions within a one year period, with a breakpoint at a maximum of 400,000 samples, were collected. Only the subreddit r/NoStupidQuestions reached the breakpoint. The filter was set up to cut texts with less than 5 words and more that 50 words. Table 3.23 shows a list of all data sources with the corresponding statistics from filtering. The second column of the table shows the absolute counts of comparatives and the percentage of filtered comparatives in relation to the source data size. It can be observed that the Quora datasets, Reddit r/NoStupidQuestions and r/askscience are the only sources breaking the mark of 5% filtered comparatives. The columns three to six show the distribution within the filtered comparatives according to their generation structure. The subreddits r/NoStupidQuestions, r/askscience and r/explainlikeimfive show high percentages of possible comparatives generated by the linguistic rules described in Section 3.1.1. The generation of comparatives by phrases is split up into the taxonomy classes 140 and 240 for keyword based phrases (x40) and the “or” classes (141 / 241) with open and closed questions. The distributions of the three categories vary significantly depending on the data source, for example, r/AskReddit compared to r/explainlikeimfive. Furthermore, the values within one class vary widely. For example,

Data source	Filtered comparatives	by rule (%)	by phrase (x40)(%)	phrase “or” (141)(%) (241)(%)	
Quora Duplicates	25,511 (5.26%)	24	46	12	19
Quora Insincere	64,183 (5.2%)	27	34	17	21
Yahoo	7,304 (2.6%)	24	23	23	30
Reddit r/AskReddit	10,907 (2.5%)	25	15	15	45
Reddit r/ask	508 (2.7%)	24	21	31	25
Reddit r/Questions	310 (2.3%)	28	20	34	18
Reddit r/NoStupidQuestions	23,946 (6.0%)	37	23	21	18
Reddit r/askscience	9,326 (6.9%)	42	29	16	14
Reddit r/AskWomen	2,990 (2.9%)	27	18	21	34
Reddit r/AskMen	3,610 (2.5%)	26	18	23	32
Reddit r/AskEngineers	359 (2.2%)	24	31	26	19
Reddit r/explainlikeimfive	2,519 (1.7%)	46	31	3	21
Reddit r/technology	450 (0.4%)	27	19	16	39
Reddit r/techsupport	1,137 (0.6%)	24	21	38	16

Table 3.23: Statistics from filtering the source data. The second column shows the absolute count of filtered comparatives for each data source. The following columns show the distribution of the samples by their generation structure.

the values for the generation by phrases vary between 15% and 46%, for the class 141 even between 3% and 38%.

Similar to the manual annotation of the filtered data from Yahoo, Quora and Reddit r/AskReddit in Section 3.3.3, data from the 10 new subreddits were manually classified into the following three classes: non-comparative (0), normal comparative (1) and hard comparatives (2). Table 3.24 provides an overview of the precision of the filtered data for all data sources. The “overall precision” is given in relation to all taxonomy classes. Three data sources break the 50% mark, with the Quora “Duplicate Question Pairs” data reaching the highest value (57%), followed by Yahoo (54%). A separation of the texts without “or” and with “or” provides further details about the data. In case both classes for “or” (141 / 241) are excluded from the data, the precision reaches up to 72%, while in total four data sources reach at least 62% precision. Once again, the Quora “Duplicate Question Pairs” dataset reaches the highest precision value (72%), followed by Yahoo (67%). Most Reddit sources show a large improvement in precision when excluding the “or” classes. For example, r/NoStupidQuestions improves by 23% and has the third best precision (63%) in the group without “or”. It is followed by r/explainlikeimfive with one

Data source	Precision overall	Precision without OR	Precision only OR	Annotation factor	Needed Samples
Quora Duplicates	57	72	31	1.8	8,772
Quora Insincere	41	57	17	2.4	12,195
Yahoo	54	67	37	1.9	9,259
Reddit r/AskReddit	13	21	8	7.7	38,462
Reddit r/ask	24	37	16	4.2	20,833
Reddit r/Questions	25	39	11	4.0	20,000
Reddit r/NoStupidQuestions	40	63	0	2.5	12,500
Reddit r/askscience	34	52	0	2.9	14,706
Reddit r/AskWomen	22	46	0	4.5	22,727
Reddit r/AskMen	28	29	27	3.6	17,857
Reddit r/AskEngineers	32	50	5	3.1	15,625
Reddit r/explainlikeimfive	50	62	15	2.0	10,000
Reddit r/technology	40	55	29	2.5	12,500
Reddit r/techsupport	18	38	3	5.6	27,778

Table 3.24: Columns 2-4 show the precisions after manually annotating data from the filtered data sources. The precision is measured for all true positive comparative questions. The fifth column calculates an annotation factor, indicating how many samples need to be annotated by a human annotator in order to get one true positive comparative question. The last column offers the calculation of how many filtered samples need to be annotated to receive 5,000 comparative questions.

percent less. Taking a look only at texts that contain “or”, it can be determined that the Quora data source (31%) and Yahoo as a source (37%) have again the highest precisions. To allow an easier ranking of the results, the fifth column calculates an annotation factor, indicating how many samples need to be annotated by a human annotator in order to get one true positive comparative question. The goal of the filtering is to reach an annotation factor of 2 (50% precision). The values go as high as 7.7 samples per one true positive for the r/AskReddit source data. By means of the annotation factor, the last column offers the calculation of how many filtered samples need to be annotated to receive 5,000 comparative questions. Because the calculated precision for each data source can only be considered an estimate (only a small part of the filtered data was manually classified), the values in the last column can only be considered as a magnitude of needed data.

Based on the precision of the filtering methods and following the conclusion of the evaluation of the data sources (see Section 3.3.1.8), Yahoo is selected for the further use in this thesis and for building the dataset for the human annotation (annotation dataset).

Furthermore, to extend the data pool, the submissions of the Reddit subreddits `r/NoStupidQuestions` and `r/explainlikeimfive` are selected. Both sources (Yahoo and Reddit) proved to be most suitable for the task of building a comparative question dataset according to the evaluation in Section 3.3.1.8. This is also the reason why the Quora data is not used further on, even though it has the highest precision overall. It is uncertain if the Quora data is open-domain and no further information (e.g., answers) is available for the data. The three chosen data sources also provide high absolute counts of data, although the percentage of possible comparatives in the Yahoo and `r/explainlikeimfive` data is low compared to `r/NoStupidQuestions` or compared to the Quora datasets. To make use of the boost in precision for the Reddit data in the case that the texts containing “or” are omitted, the taxonomy classes 141 and 241 will be excluded from the Reddit data for building the annotation dataset. This will enable the decrease of the overall needed number of human annotations to a greater extent. In order not to leave linguistic comparative structures containing “or” out from the dataset, all samples from the Yahoo data, including samples with “or”, are allowed.

Data source	Source size	Filtered comparatives	by rule (%)	by phrase (x40)(%)	phrase “or” (141/241)(%)
Yahoo	7,037,759	132,749 (1.89%)	23.1	20.3	56.5
Reddit <code>r/NoStupidQuestions</code>	1,174,971	38,827 (3.3%)	60.1	39.8	-
Reddit <code>r/explainlikeimfive</code>	1,402,561	73,847 (5.27%)	60.7	39.3	-

Table 3.25: Statistics after fetching and filtering the full data sources. The third column shows the absolute count of filtered comparatives for each data source. The following columns show the distribution of the samples by their generation structure.

In order to proceed with the maximum amount of source data, all submissions from the two subreddits were fetched with the Reddit scraper script. Furthermore, the complete Yahoo dataset is utilized now. The source data totals in 9.6 million samples. Applying the filter script to the data results in a total of 245,423 filtered possible comparative samples. Multiplying the filtered samples by the precision for each of the data sources leaves an estimate of 141,931 true comparative question samples. This high estimate of comparative questions exceeds the expectations and also the necessary figure of positive samples by far. Table 3.25 shows the distribution of sample classes for the full-size filtered data sources. The subreddit `r/explainlikeimfive` has 5.3% possible comparatives. In comparison to the previous filtering, the number of comparatives for `r/NoStupidQuestions` has decreased by 2.7% to 3.3% and Yahoo has decreased by 0.7 to 1.9% of filtered comparatives.



The false omission rate (FOR) for the data is 1%, that means that 1% of the total number of negative (not comparative) calls are wrong. Therefore, 1% of the non-comparative tagged data might actually be comparative. That would result in 93,600 missed comparatives. Calculating the recall out of these numbers indicates that the recall over all datasets is 60.2%.

This section utilized the comparative taxonomy and the filtering scripts to decide which data source should be used for creating a comparative question dataset. It was decided to proceed with Yahoo and Reddit data. Data from both sources is provided as filtered datasets, ready for the annotation by human workers. The next section will describe this process of using the filtered data as annotation datasets in human annotation tasks. The goal is to build two datasets: one for the task of machine-classifying comparative questions and one for identifying the comparative objects and their aspects.

### 3.4 Human annotation task 1: Classification

This and the following section describe the process of labelling the annotation datasets from Section 3.3.4 in crowd sourced annotation tasks. As previously explained in Chapter 1, one goal of this thesis is to provide datasets for two tasks. The first task is the classification of comparative questions. This task aims to identify comparative questions in a set of given samples. The second task is the identification of comparative objects and aspects within the set of comparative questions. For both tasks, a different labelled dataset is needed in order to use supervised machine learning. The following subsections outline the process of building the labelled dataset for the classification task. Section 3.5 describes the process of building the sequence labelled dataset for the second task. Both tasks will be conducted with the Mechanical Turk (MTurk)<sup>18</sup> platform provided by Amazon Web Services (AWS). MTurk is a crowdsourcing marketplace, which provides a distributed workforce for digital tasks to businesses and researchers. Figure 3.5 shows the working principle of MTurk. The marketplace provides requesters and workers with a common platform to conduct projects together. It is a marketplace in the sense that the requester publishes a project and sets a reward for a Human Intelligence Task (HIT). A HIT is a small part of a project that a worker needs to complete in order to be paid. For example, the project is to classify 1,000 sentences into categories. Each task might contain 10 sentences that a worker needs to read and classify. Workers can complete multiple tasks in a row and will, for example, be paid 20 cents per finished task. To get a more reliable answer, a HIT can be assigned to many unique workers and the answers can be accumulated afterwards. Projects are often split into multiple batches containing a number of tasks.



Figure 3.5: The figure shows the working principle of Amazon Mechanical Turk. Source: [mturk.com](http://mturk.com)

To create both datasets with MTurk, a total budget of \$1,000 was available. Each annotation project is split into two phases: the pilot phase and the main phase. The purpose of

<sup>18</sup><https://www.mturk.com/>

the pilot phase is to evaluate workers and find high quality ones for the main phase. In the pilot, workers are presented tasks to which the answers are known by the requester. This way, workers participating in the pilot can be evaluated, for example, on their accuracy at classifying sentences. In the main phase, the project data is provided in batches to the workforce for annotation. Only workers that completed the pilot and fulfilled the project's requirements can take part in this phase. As a result, the budget needs to cover two pilot phases and two main tasks. A cost calculation shows that with the budget a maximum of 11,000 samples can be classified. The samples with a positive label can be sequence labelled by human workers in the second annotation project (more details see appendix Table 3). For the calculation, an annotation rate of 2 was applied. This results in 5,500 samples for the sequence tagging. The costs per classification HIT (with 20 questions) was estimated at \$0.22. The costs for a sequence tagging HIT is estimated higher due to the larger effort and, therefore, longer work time per task. The sequence tagging HITs are estimated at \$0.25 for 10 questions. Both tasks have three assignees per HIT. Including an estimation of \$60 for both pilot phases, this would lead to total cost of \$990. As Mechanical Turk is a marketplace, it will become clear during the execution of the projects if workers, whose work quality is good enough, accept this price per HIT. Often, prices might change between batches depending on the response of the workers. The aim is to get as many annotations as possible with the budget. Classifying around 10,000 samples in the first task seems like a reasonable goal.

### 3.4.1 Task preparations

There are two essential parts in the task preparation for Mechanical Turk. Firstly, a Website providing an interface for the assignment is needed for the workers to conduct the task. Secondly, the data that should be used in the task needs to be prepared for the task, for example, the data needs to be split in batches. The following paragraphs explain the steps that were taken for the comparative classification task.

Mechanical Turk allows requesters to create a task specific website for a project. The website is shown to the workers and their inputs are recorded. MTurk allows HTML, CSS and JavaScript to customize the websites for the requester's needs. The requester can define placeholders in a template, which will be filled with data when the task is active. A batch of data must be uploaded to MTurk when publishing the task. A simple example is a template website displaying a line of text and some options to be chosen by the worker. The text line is exchanged in each HIT and the option that the worker clicked on is recorded when sending the HTML form.

Instructions

Examples and task definitions

### Search Definitions

If you do not know the meaning of any terms in the texts below, write it here and search (opens in new window):

**1. Select the appropriate option for each of the ten sentences:**

#	Sentence	Classification
1.	\${text1}	<input type="radio"/> <b>Comparative Question</b> <input type="radio"/> <b>Not comparative</b> or not a question <input type="radio"/> I am not sure
2.	\${text2}	<input type="radio"/> <b>Comparative Question</b> <input type="radio"/> <b>Not comparative</b> or not a question <input type="radio"/> I am not sure

**2. Tell us about yourself (Only first TIME):**

**Age** **Country**

Select... ▾

**Are you a native English speaker?** **If you are not a native English speaker, what is your native language?**

Select... ▾

English ▾

**3. Your comments:**

Type your comment about this HIT here

Submit

Figure 3.6: The figure shows the top part of the template for sentence classification.

Basically, the same approach is used for the pilot and the main classification task. Figure 3.6 is a screenshot of the template shown to the workers. Instead of one line of text, the template displays up to 10 samples from the annotation data to the worker (in the `text` place holders). For each sample, three options are given to the worker: “Comparative question”, “Not comparative or not a question”, “I am not sure”. Furthermore, the worker can find a button on the top of the page to open the task instructions and a list of ten labelled examples with explanations (see appendix Figures 1 and 2). A special “Tell us about yourself” part of the template is dedicated only to the pilot task (see Figure 3.6 2.). The workers are asked to introduce their age, their country and, most importantly, if they are a native English speaker. In case they are not native English speakers, the form asks them to introduce their native language. The age is entered via a drop-down field, allowing the selection of an age group (e.g. 18-24). The country and language fields pro-

vide a convenience function that suggests country names and languages via a JavaScript function. This might simplify the post-processing of the data because the workers have the chance to select the proposed language or country. Since the provided list might not be complete, the fields allow free typing of text. The data can also be used to filter the workers after the pilot. Because the annotation datasets are completely in the English language, native English speakers or residents of countries with English as an official language might comprehend the texts better. Understanding the questions and having a large vocabulary to understand the comparisons and to know the compared objects is essential for the tasks. Moreover, these personal information fields are utilized to calculate some statistics about the workers. For example, the rough number of workers that are native speakers can be determined. This number might not be accurate, since nothing prevents the workers from lying about their personal information. The last part of the page (see Figure 3.6 3.) provides a comment field, which is not required to use. Only the classification radio buttons and the personal information fields are required. To avoid that the workers cheat in the classification task, by sending an empty form, the required fields are checked. The MTurk template is based on a template from a task previously conducted by the UHH Language Technology Group<sup>19</sup>. The template was modified to fit the classification task and a Bootstrap<sup>20</sup> based design was added to provide a clean and clear layout.

To carry out the task with the template, the data needs to be prepared. Amazon's Mechanical Turk demands a comma-separated file format that contains the data for one batch. Each line in the file needs to contain all data for one HIT. Regarding the classification task of this thesis, one HIT needs text input for 10 questions and the question's IDs to be able to identify them after the task is completed. Furthermore, the file contains a sub-batch counter and the data source for each HIT. To generate the batch files, a Python script is implemented. Listing 3.4 shows the most important parts of the script. At the top of the script, important parameters need to be set. Most importantly, the sub-batch size (Line 1) and the total data count for a batch, as well as the distribution of samples from the source data (Lines 2-5). The general function of the script is based on three parts. Firstly, an exclusion database for already used IDs (KeyExclusionList) in form of a comma-separated file, which is loaded at the beginning of the main function (Lines 22-32) and stored at the end of it (Lines 58-63). Used IDs from previous batches are loaded this way and, after the creation of the new batch, all used keys are saved again. This ensures that each sentence is only processed once in the human annotation. In case no key exclusion list can be found, the script generates a new one. Secondly, the source data is loaded and processed. In this case, the Yahoo data, the Reddit r/explainlikeimfive (eli5) and r/NoStupidQuestions (NSQ) data sources are loaded (Line 34).

---

<sup>19</sup>Code repositories can be found here: <https://github.com/uhh-lt>

<sup>20</sup><https://getbootstrap.com/>

```

1 | subBatchSize = 10
2 | dataCount = 1020 # pick a number dividable by part
3 | yahooPart = 3 # in 1/part totaling up 1/1
4 | eli5Part = 3 # in 1/part totaling up 1/1
5 | nsqPart = 3 # in 1/part totaling up 1/1
6 |
7 | def shuffleKeyList(keyList, part):
8 |     #Shuffles a list of given keys and gives back a list in the size of the data sources part
9 |     shuffle(keyList)
10 |     calcSize = (int(dataCount / part))
11 |     roundUp = 0
12 |     if (calcSize % subBatchSize > 0):
13 |         roundUp = subBatchSize - (calcSize % subBatchSize)
14 |     dataSize = roundUp + calcSize
15 |
16 |     if (dataSize > len(keyList)):
17 |         dataSize = len(keyList)
18 |         print("The source data count was not high enough. The maximum is taken.")
19 |     shuffledKeyPart = keyList[:dataSize]
20 |     return shuffledKeyPart
21 |
22 | if __name__ == '__main__':
23 |     ###Load KeyExclusion file, if it exists##
24 |     usedKeys = [[], [], []]
25 |     try:
26 |         with open("KeyExclusionList.csv", 'r', newline='', encoding='utf-8') as csvfile:
27 |             yReader = csv.DictReader(csvfile, delimiter='\t')
28 |     #... Read found Keyfile into usedKeys[]
29 |     except EnvironmentError:
30 |         with open("KeyExclusionList.csv", 'w', newline='', encoding='utf-8') as tsvfile:
31 |             #Create new keyExclusionList as non was found
32 |             print('KeyExclusionList.csv not found. An Empty one was created.')
33 |     ###Load data source files###
34 |     yahooQueryys = open(yahoofile)
35 |     #...do the same for the other datasets
36 |
37 |     ### Create Random ID lists ###
38 |     unUsedYahoo = list(set(yahooQueryys.keys()) - set(usedKeys[0]))
39 |     yahooShuffledKeys = shuffleKeyList(unUsedYahoo, yahooPart)
40 |     #...do the same for the other datasets
41 |
42 |     ### Generate Output for MTurk ###
43 |     with open(outfile, 'w', newline='', encoding='utf-8') as tsvfile:
44 |         writer = csv.DictWriter(tsvfile, fieldnames=fieldnames, delimiter=',')
45 |         batchID = 0
46 |         for subBatch in range(0, int(len(yahooShuffledKeys)/subBatchSize)):
47 |             row = {'batch_id': batchID, 'source': "yahoo"}
48 |             for b in range(0, subBatchSize):
49 |                 str_id = str("id" + str(b + 1))
50 |                 row[str_id] = yahooShuffledKeys[b+subBatch*subBatchSize]
51 |                 str_text = str("text" + str(b + 1))
52 |                 row[str_text] = yahooQueryysClean[yahooShuffledKeys[b + subBatch *
53 |                                     subBatchSize]]
54 |                 writer.writerow(row)
55 |                 batchID += 1
56 |             print('Exported ' + str(batchID) + ' Batches for Yahoo')
57 |             #...do the same for the other datasets
58 |
59 |         with open("KeyExclusionList.csv", 'w', newline='', encoding='utf-8') as tsvfile:
60 |             writer = csv.DictWriter(tsvfile, fieldnames=fieldnames, delimiter='\t')
61 |             for key in usedKeys[0] and for key in yahooShuffledKeys:
62 |                 row = {'key_id': key, 'source': "yahoo"}
63 |                 writer.writerow(row)
64 |             #write previous used and new keys to exclusion list

```

Listing 3.4: Pseudocode of the MTurk data preparation script for the human classification project.

From the loaded data, the unused IDs are filtered out and shuffled in the function `shufflekeyList` (Line 7). The function also cuts from the list of IDs the number required by the script's parameters. Thirdly, the data is exported in a MTurk compatible way (Lines 43-56). In this step, the data of each data source is broken down to the parts needed for a sub-batch. Each HIT (sub-batch) contains samples only from the same source.

### 3.4.2 Pilot

The first step of the human classification task is dedicated to find good, reliable and accurate workers. For this, the workers need to classify a series of sentences from which the answer is known. The sentences are selected from the source data. The manual classification from Section 3.3.3 and the categorization into normal comparative and complex comparative sentences allows to find fitting examples. Whereas the pilot provides three answer possibilities to the workers, only comparative or non-comparative examples are selected. The "Not sure" option has the sole purpose to filter out workers that pick random answers or do not know the answer. Even though some sentences might be controversial, the goal is to find workers that read the task rules and follow the task definition of the comparative questions. From the source data, 10 sentences are selected: five negative and five positive ones. A category is assigned to each sentence, in order to try to estimate its level of difficulty. The aim is to have a set of easy, medium and hard questions for the pilot. The selected sentences with their correct label (gold label) and their difficulty can be found in Table 3.26. Question 2 is considered to be of medium difficulty because it has no real comparative objects. These objects form a listing. Question 5 is considered hard to answer because, even if it uses the language of a comparison, the comparative objects are not explicit in the sentence. Therefore, it is not comparative by definition. Question 7 is considered of medium difficulty because it is a listing of things and not a comparative. Question 8 is considered of hard difficulty, as it is a comparison by language, but not by definition. This sentence compares "some people" to "others", which are both open groups. And finally, Question 9 is considered of hard difficulty because the language and the topic of the question are complicated. The topic might not be known to everyone, especially non-native speakers may have problems here. Using hard and medium difficult questions aims at finding workers that have read and understood the definition and the examples of the task.

With the samples from Table 3.26, the pilot project was published and a maximum of 100 workers were allowed to take part in it. After the project was completed on Mechanical Turk, the results were evaluated. The most important measure for each worker is the classification correctness. The pilot project resulted in one worker with 100% correct clas-

#	Gold label	Difficulty	Sentence
1	Comp	Easy	what is the difference between a cappuccino and a latte?
2	Non-Comp	Medium	what can i do with a bachelors degree in history? should i stay in grad school or try to find a real job?
3	Comp	Easy+	should i buy or rent in california?
4	Comp	Easy	what is the difference between burning and ripping?
5	Non-Comp	Hard	what are the differences between the those beams?
6	Comp	Easy+	can you please eli5 the difference between ham and shortwave, like if i wanted to set my parents (who live in ca fire country) up with a simple uncomplicated means of 911 communication in case cell towers go down?
7	Non-Comp	Medium	what calculations can only be done by supercomputers or quantum?
8	Non-Comp	Hard	why some people tolerate alcohol less than others ?
9	Comp	Hard	why does everyone say hitting a pitch from a mlb pitcher is so impossible but catching the throw is expected every time when the difference is still less than a quarter of a second
10	Non-Comp	Easy	why are objects in my mirror closer than they appear?

Table 3.26: The samples for the MTurk classification pilot with their correct label (gold label) and the estimated difficulty for the workers. The samples are taken from the filtered datasets and were only converted to lower case letters.

sifications, 8 workers with 90% and 19 workers with 80% correct classifications. 85% of the workers stated that they were native speakers. With the aim of increasing the amount of workers who achieve a correctness higher than 80%, a second pilot was conducted. In order to receive a finer gradation of the calculated correctness, the number of pilot questions was increased to 15. The added samples are shown in Table 3.27. Question 14 can be considered of medium difficulty as it compares “other colors” with black, which is not considered a comparison by definition. The other four questions are categorized as easy.

The evaluation of the second pilot results in four workers with a perfect score, 9 workers with 93% correctness (one question wrong), 16 workers with a score of 87% and 24 workers with 80% correct classifications. 79% of the workers from the second pilot stated that they are native speakers. Accumulated with the first pilot on all 15 questions, this results in 81 workers with a correctness higher than 80% and still 22 workers with a correctness higher than 90%. In total, 5 workers managed to score 100% correctness (see Table 3.28). The evaluation of the Samples 1-10, which were the same in both pilots, shows that 7 workers have a 100% correct classification and 63 workers have answered more than 80% of the samples correct. The evaluation of the samples that were categorized with a hard



#	Gold label	Difficulty	Sentence
11	Comp	Easy	why do girls walk different than men?
12	Non-Comp	Easy	do ghosts or unseeable creatures exist?
13	Comp	Easy+	what is the difference between a load balancer and a reverse proxy server?
14	Non-Comp	Medium	when printing out a resume, are there any other colors than black you can use to print?
15	Non-Comp	Easy	how do i make a google doc as secure as possible?

Table 3.27: The additional samples for the second MTurk classification pilot with their correct label (gold label) and the estimated difficulty for the workers. The samples are taken from the filtered datasets and were only converted to lower case letters..

difficulty, shows that less than half (40%) of the 200 workers could answer more than one sample right.

In this paragraph, statistics over both pilots are described. In total, workers from 15 different countries participated in the pilots. 69% of the workers are from the United States of America and 90% are from countries that have English as an official language <sup>21</sup>. 83% of the workers consider English as their native language. The biggest group of workers is between 30 to 39 years old (39%). The mean time to classify one sample was 19.71 seconds. For the people with 100% correct classifications, the mean time per question was 15,91 seconds. This indicates that these people were more efficient in using their time or could understand the samples more easily. The evaluation of the workers correctness only on the hard questions shows that 8% were able to answer all of them correctly. Table 3.29 shows the percentage of correct answers for each question. It can be seen that the allotted

<sup>21</sup>USA, United Kingdom, Canada and India.

Samples	Classification correctness									
	0%	20%	30%	40%	50%	60%	70%	80%	90%	100%
		-29%	-39%	-49%	-59%	-69%	-79%	-89%	-99%	
1-15	3	1	3	11	15	40	46	59	17	5
1-10	3	1	6	12	22	35	58	41	15	7
5,8,9 (hard)	40	-	79	-	-	65	-	-	-	16

Table 3.28: The table shows the number of workers sorted by their classification correctness (%). The data is accumulated from both pilots and evaluated on the samples shown in column one.

categories fit with the resulting correctness. Only Questions 13 and 14 seem to be easier than expected. This affirmation regarding the suitability of the categories determines that the pilot was a success and the high scoring workers might be good for the task at hand. Anyway, it was to be expected that workers might make efforts only once in the pilot to take part in the main task and “just” collect the money. Furthermore, the examples do not necessarily represent all types or difficulties of samples in the data. Therefore, an evaluation and a constant monitoring was still necessary in the main phase to get good results.

#	Gold label	Difficulty	Correct	Sentence
1	Comp	Easy	92%	what is the difference between a cappuccino and a latte?
2	Non-Comp	Medium	62%	what can i do with a bachelors degree in history? ...
3	Comp	Easy+	71%	should i buy or rent in california?
4	Comp	Easy	91%	what is the difference between burning and ripping?
5	Non-Comp	Hard	43%	what are the differences between the those beams?
6	Comp	Easy+	71%	can you please eli5 the difference between ham ...
7	Non-Comp	Medium	67%	what calculations can only be done by ...
8	Non-Comp	Hard	53%	why some people tolerate alcohol less than others ?
9	Comp	Hard	33%	why does everyone say hitting a pitch from a mlb ...
10	Non-Comp	Easy	82%	why are objects in my mirror closer than they appear?
11	Comp	Easy	85%	why do girls walk diffrent than men?
12	Non-Comp	Easy	92%	do ghosts or unseenable creatures exist?
13	Comp	Easy+	91%	what is the difference between a load balancer and ...
14	Non-Comp	Medium	84%	when printing out a resume, are there any other ...
15	Non-Comp	Easy	94%	how do i make a google doc as secure as possible?

Table 3.29: The table shows the samples of the MTurk classification pilot with their correct label (gold label), the estimated difficulty for the workers and the percentage of correct answers from the 200 workers (Samples 11-15 were classified by 100 workers only).

### 3.4.3 Main phase

The main phase of the human classification task uses the workers from the pilot phase to classify the data from the three sources. The goal is to get 10,000 human labelled samples. Each sample is classified by three unique human workers. The classification project is run in batches. The data for the batches is evenly split using a third from each source per batch. The human classification began with a test batch containing only 180 samples. The batch had 20 samples per HIT and, therefore, 9 unique HITs and 60 samples from

each source. All workers with a classification score better than 80% in the pilot could participate. In total, 62 workers were unlocked for the task. The purpose of the first small test batch was to validate the process of the HIT publication and the unlocking of workers. Furthermore, the batch was run to test the post-processing of the classified data. For each finished batch, a Python script analysed the results and a manual analysis in Excel was carried out.

The Python script extracts the important information from the recorded data of the batch. In this case, the extracted data is the input text and the sample ID, as well as the ID of the worker, the work time and the answer classification. The answers of the workers are assigned an integer, “Comparative” - 100, “Non-Comparative or No Question” - 10 and “Not sure” - 1. To find a common answer from all three workers for a sample, the assigned integers are summed up. This will be called “vote” from now on. A majority vote (>50%) can result in the four categories “comparative”, “non\_comparative”, “not\_sure”, “inconclusive”. This means that a sample, which was voted comparative twice (2\*100) and once not comparative (10), will be a comparative (210) by vote of 66.6% of the voters. This simple majority vote will be called “66%votes” as an abbreviation. A stricter voting process is the absolute majority vote, where 100% of the workers have to agree. This will be called “100%votes” as a short form. Besides the clear votes for comparative or non-comparative, there are votes that are inconclusive (111) and votes in which “not sure” has the majority.

After the automated analysis with Python, the data of the test batch was manually analysed. Using an Excel file, statistics on the votes and workers were generated. For example, the first batch has 50% comparative votes and 43% non-comparative majority votes. The mean time per sample is 14.7 seconds, a bit lower than in the pilot, but expected as the personal information fields are not part of the main task. The inconclusive votes and the “not sure” votes were utilized to find workers that might not answer honestly. For this, the answers of a worker are manually reviewed and it was tried to estimate if the answers are reasonable. Workers for whom the answers seemed random or who had impossible low task completion times, for example 51 seconds for 20 questions, were excluded from the following batches. In order to get a clearer vote from the workers, a fourth option was included after the test batch. The “Not comparative or Not a question” option was split up into “Not comparative” and “Not a question”. This provides a clearer result about how many of the samples are actually not a question. In the examples shown to the workers during the HIT, it was added that a comparative statement should be tagged as “Not a question”. Further to a clearer result, the fourth option made it easier to find workers that classified randomly or not honestly. The “no\_question” label was integrated into the votes with the integer 1000. In the voting process, every sample, which was classified as “Not a

question” by at least one user, was flagged and assigned the vote as “no\_question”, even though this might not have been the majority vote. In the manual analysis, these samples and users could then be easily reviewed. Table 3.30 shows the votes and the possible categories. Further to the generation of statistics on the workers and the votes, a manual classification of at least 100 randomly picked samples was made for each batch. Samples flagged as “No question” are excluded from the manual evaluation. Binary gold labels (Comparative / Non-comparative) were assigned to the samples. With this gold labels, the precision, the recall and the F1 score of the workers could be calculated for each batch.

Comparative	Non-comparative	Inconclusive	Not sure	No question
201	21	111	12	1xxx
210	120		102	2xxx
300	30		3	3000

Table 3.30: The table shows the vote categories with the calculated votes in each column. In the case that at least one worker voted “No question”, the sample is flagged and assigned to the category.

The process of preparing a batch of data, conducting the HIT online, analysing the results and generating the gold labels was conducted for ten full sized batches with 1,020 samples each batch. The batches can be separated in Series A and Series B. Series A contains Batches 2-4 (the test batch is Batch 1) and Series B includes Batches 5-11. The difference between the two series is the type of worker. In Series A, workers with a correctness of 80% or higher in the pilot were allowed. In total, this were 62 workers. The first three full batches showed a good participation of the workers. Yet the macro F1 scores were in average at 0.848, showing that an improvement was possible. Therefore, in Series B the number of workers was reduced to everyone who had a correctness of 90% or higher in the pilots. This resulted in 22 workers that were allowed to participate in the project. In Series B, the average macro F1 score increased to 0.918 through this measures. More statistics about the complete classification task and the complete human annotated dataset for comparative classification are described in the following section.

### 3.4.4 Dataset statistics

This section provides statistics about the final dataset as a result of the human annotation task for classification of comparative questions. The data from the eleven separate batches was collected and post processed with a Python data collector script. The post-processing steps in the generation of the final dataset include:

1. Loading and merging of the original annotation data and the results of the classification batches.
2. Excluding “No question”, “Inconclusive” and all votes that have at least one vote “Not sure”.
3. Excluding the gold labelled data from the set and exporting it as a test dataset.
4. Correcting the comparative votes based on pronouns keywords like “others”, “another”, “everyone” and “somebody” to non-comparative.
5. Revoteing the samples tagged as non-comparative in the second human annotation task (sequence tagging).
6. Extending the dataset with non-comparative data that was filtered out in the data-mining step (taxonomy class 410-430).
7. Exporting the dataset and generating statistics. The most important exported fields are: ID, text, comparative class, vote and the vote label (comparative / non-comparative).

The Steps 1,2 and 7 are mandatory. All other steps can be set up in order to be skipped. This way, there is not one final dataset but various datasets that contain different data. This approach has the advantage that the different methods (3-6) that might improve the quality of the data can be evaluated and compared in the machine-learning task (see Chapter 4). Without the additional steps, the workers labelled 10,380 samples. 10,106 samples have non-comparative or comparative votes and can be used further on (Step 2). Out of these 10,106 samples, 4,539 (44.91%) are labelled comparative and 5,567 (55.09%) are labelled non-comparative. Therefore, the goal of 5,000 comparative samples (see Section 3.3.3) out of 10,000 annotated samples was nearly met and the dataset has a almost balanced distribution of comparatives and non-comparatives. The “100%votes” have the majority (66%) of votes in the dataset. Within the category of the “100%votes”, 38% are non-comparative and 28% are comparative votes. The “66%votes” are represented with 17% for each class. This distribution indicates that 66% of the samples were more clear or easier for the workers to answer.

When analysing the distribution of all samples of the dataset within the classes of the comparative question taxonomy, it can be seen that the open question class with phrases (240) is the biggest part of the data with 24.5%, followed by the open questions with “or” (241: 9.9%), the closed questions with phrases (140: 8.5%) and “or” (141: 8.4%) and the open questions with “than” and an unknown word combination (230: 8.4%). The other classes have a maximum of 5.3% and a minimum of 0.02% (or 2 samples) per class. 10 out of the 28 classes have less than 0.62% (or 63 samples) samples per class. All of these ten classes belong to the xx2: as/as or the xx3: so/as constructs. A detailed overview can be found in Table 4 of the appendix.

Analysing the distribution of all comparative samples within the classes of the comparative question taxonomy shows that the open question class with phrases (240) holds a third of all comparatives (32.3%). The closest following classes are 230 and 140 with 8.8% each. Five classes do not have any comparatives. Looking at the data from the perspective of the generation type, the comparatives are evenly split. 50.3% of the samples are generated by phrases (140/141/240/241) and 49.7% generated by rules. Sentences containing “or” make up 9.2% (141/241) of all the comparatives. In total, there are 1,854 sentences containing “or” in the dataset and only 419 comparatives with “or”. This means that only 22.5% of the questions containing “or” are comparative. From the perspective of question type, the open questions make up the majority with 73.9%. Furthermore, the distribution of the samples per source has changed. The data input was evenly split between the three data sources. In the comparative samples eli5 has a share of 41.6%, NSQ has 35.0% and Yahoo has 23.6%. With this data, the precisions for the filtering task can be calculated again. All three sources remain behind the precision expectations from the filtering process in Section 3.3.4 (see Table 3.24). Reddit NSQ has a precision of 49% (-14%), eli5 a precision of 55% (-7%) and Yahoo a precision of 31% (-23%) on the comparative samples. While the Reddit data performs close to the 50% precision target, Yahoo remains far from it.

Throughout the human annotation phase, random samples were taken from each batch for a manual classification in order to get gold labels. With these gold labels the accuracy and scores such as the F1 measure could be calculated for each batch and series<sup>22</sup>. For each batch approximately 10% of the data was evaluated. For Series A the macro F1 score is 0.848 over all votes and for both classes, comparative and non-comparative. The “100%votes” in Series A reach a macro F1 score of 0.940 and the “66%votes” a F1 score of 0.717. For Series B the workers needed a higher classification correctness in the pilot to participate. The overall macro F1 score for Series B is 0.918 (+0.07). This is an increase of 8.25% by changing the workers. The “100%votes” in Series B reach a macro F1 score of 0.981 and the “66%votes” a F1 score of 0.772. Changing to better workers in Series B

---

<sup>22</sup>Series A: Batch 2-4; Series B: Batch 5-11.

increased the scores for both vote types, but it could not close the gap between the vote types. The “66%votes” are still 0.21 behind the “100%votes”, indicating again that the samples that only have a two third agreement seem to be harder or more controversial sentences. The macro F1 score over both classes for the complete data from both series is 0.898. A total of 1,096 samples were manually classified for that.

The four steps to improve the dataset are independent of each other and change the statistics of the dataset. Step 3 excludes the samples that were gold labelled after each batch from the dataset. As a result of this measure, there will be a dedicated test dataset that represents a gold standard. In total, 796 samples are excluded from the dataset. Step 4 cares about the requirement that comparatives against open groups should be considered non-comparative. The human annotation shows that the workers had difficulties with this. Therefore, this step corrects the votes for few selected keywords to a non-comparative vote. This measure is a big intervention into the voting process. All keywords were manually evaluated in terms of the frequency of positive comparatives. Only the non-ambiguous keywords are excluded. Step 5 is a revote on samples that were tagged non-comparative in the second human annotation task. Even though this step is listed here, it could only happen after the annotation in Section 3.5 was finished. The revote was done with a total of six votes for votes that were flagged with at least one non-comparative vote in the second task. In total, 28 samples were revoted to be non-comparative and 52 samples were inconclusive after the revote and were excluded. The last optional step (6) extends the dataset with non-comparative data. The data is taken from the filtering process of the data mining (see Section 3.3) in which non-comparative statements and questions were filtered out. The number of added non-comparative samples can be set up in the script. For the machine-learning task, datasets from 1,500 up to 9,000 added samples were created. The dataset with all options activated and an extra of 6,000 non-comparative data samples has a total of 15,278 samples, of which 3,814 samples are comparative.

#	Name	Size	Comparative	Non-Comparative
1	AllData	10,106	4,539 (44.9%)	5,567 (55.1%)
2	AllData ExtraTest	9,330	4,152 (44.5%)	5,178 (55.5%)
3	AllData ExtraTest NoOpenGroup	9,330	3,894 (41.7%)	5,436 (58.3%)
4	AllData ExtraTest Revote	9,278	4,072 (43.9%)	5,206 (56.1%)
5	AllData ExtraTest NoOpenGroup Revote	9,278	3,814 (41.1%)	5,464 (58.9%)
6	AllData ExtraTest NoOpenGroup Revote NonCompData1.5k	10,778	3,814 (35.4%)	6,964 (64.6%)
7	AllData ExtraTest NoOpenGroup Revote NonCompData3k	12,278	3,814 (31.1%)	8,464 (68.9%)
8	AllData ExtraTest NoOpenGroup Revote NonCompData6k	15,278	3,814 (25.0%)	11,464 (75.0%)
9	AllData ExtraTest NoOpenGroup Revote NonCompData9k	18,278	3,814 (20.9%)	14,464 (79.1%)

Table 3.31: The table shows the nine different classification datasets. As a result of the different post-processing steps on the human annotated data, the data samples and the balance between the two classes varies in the final datasets. The name of a dataset indicates the post-processing steps that were used.



## 3.5 Human annotation task 2: Sequence tagging

This section describes the process of labelling the comparative data from Section 3.4 in a crowd sourced annotation task. The goal of this annotation task is the identification of comparative objects and aspects within the set of comparative questions to use it for the training of machine-learning models. The following subsections describe the process of building the sequence labelled dataset. The human annotation task will be conducted by means of the Mechanical Turk (MTurk)<sup>23</sup> platform provided by Amazon Web Services (AWS).

### 3.5.1 Task preparations

The task preparation is similar to the one from the classification task. Firstly, a website providing an interface for the assignment is needed for the workers to conduct the task. Secondly, the comparative data of the classification task needs to be prepared. And thirdly, the results need to be analysed and a voting process on the sequences needs to take place. The following paragraphs explain the steps that were taken for the sequence-tagging task.

This sequence-tagging task is also conducted by means of Mechanical Turk and, therefore, it has to respect the same requirements as the classification task. Due to the nature of the annotations, the task has a different requirement regarding the user interface compared to the classification task. The aim is to mark a sequence of a provided text and add a label to that sequence. The template from the first human annotation task serves as a base for the new template. Especially the already proven concept for the Instructions, Examples and the “Tell us about yourself” part is useful for the second task. For the second task it is even more important to provide a clear and easy useable interface to the workers, as this task is more complicated and detailed. Marking a span of words or letters is the necessary core function of the template. There are two approaches for this: the first one is to mark single characters in the text with the mouse (click and drag) and record the position of the start and the end of the marking. The second approach is to separate the text in a pre-processing step before publication (tokenization). This way, each token (word, punctuation and so on) can be made clickable in the template and gets a unique ID. The ID needs to be recorded when clicking. In order to provide the user with the simplest tool possible, the second option was chosen. The Tool for Annotation of Low-resource ENtities (TALen)<sup>24</sup> project provides a function similar to what is needed for this task. The MTruk classification template was extended by code snippets taken from the TALen and the code was modified to fit the sequence labelling task. The final

---

<sup>23</sup><https://www.mturk.com/>

<sup>24</sup><https://github.com/CogComp/talen>

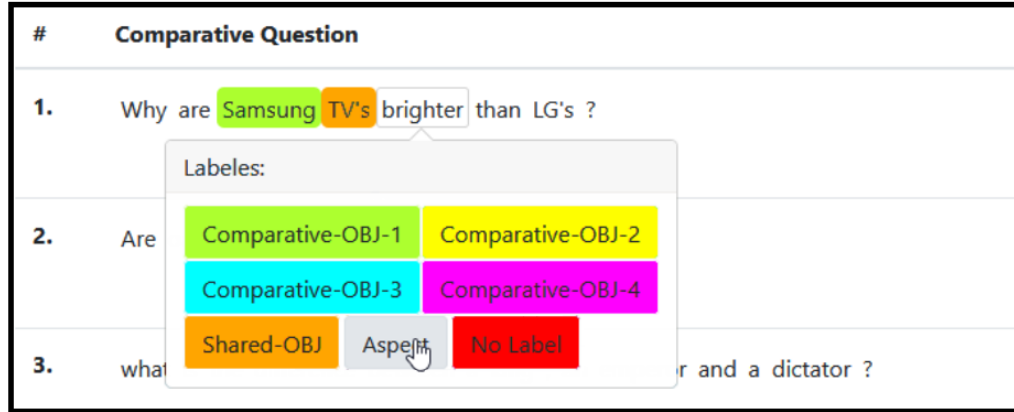
template is shown in Figure 3.7. The template is split up into two areas: the demo area on the top of the page and the task area below. The demo area gives a very short description of the task and displays an animated image of how to use the labelling tool. The labelling tool provides the functionality to left-click on a token and select a fitting label from a pop-up menu. Clicking and dragging the mouse can be used to select multiple tokens in a sequence and then pick a label. Right clicking on an already set label can be used to remove this label. The workers can label up to four distinct comparative objects, common shared objects and the aspect of the sentence (see the definition of comparatives for more details in Section 3.1.4). The pop-up menu also supplies another way of removing a label. The task area of the template gives more detailed information and example pop-ups and a definition search in the same way as in the classification task. The “work” area is separated in three parts. The first part shows the comparative questions that need to be labelled. Here, each question can be individually tagged with the labels provided by the labelling tool. Due to the fact that the human classified data only has an precision of 89% for the comparative samples, one out of ten questions might be not comparative. Therefore, the workers of the second task are also provided with the option to label a question as non-comparative. In order to prevent the workers from cheating, the “Comparative Objects annotated” option is selected automatically by the script if the worker tags a sequence and can not be set by the worker. The second and third part of the “work” area are the same as in the classification task. The “tell us about yourself” part is only visible in the pilot phase.

The second part of the task preparation is to supply the MTurk template with data. The basic functionality of the Python script remains the same as in the classification task. The exclusion of already utilized question IDs, the loading of the source data and the export to an MTurk valid format are only slightly changed. The main difference in the data preparation is the pre-processing of the data. The sentences of the source data need to be split into single tokens. The tokenization is an important step in the generation of the dataset. The format of a sequence dataset is often a tab-separated file in which each line holds a token and one or more tags. The tagging mechanism for this human annotation task works the same way. The tokens are individually labelled by the workers. Therefore, changing the separation of the tokens afterwards might result in complications. The box below shows an example text, which contains some difficult separation cases. For example, punctuation should be separated from neighbouring characters. Yet, there are exceptions in which this should not happen. For example, “N.Y.” as abbreviation of the city New York should not be split. Furthermore, short forms of words (contractions) like “is” in “Who’s” or “not” in “can’t” should be separated. For words that are separated by a hyphen, like “j-lo” or “large-scale”, it is difficult establish rules. The highlighted cases in the box below can be categorized as separation with hyphens and separation of negative short forms. To avoid the edge cases with hyphens and to have a more natural separation

## Demo:

### How to label **Comparative Objects** and **Aspects**:

Click on words to select a fitting label. Please mark objects that are compared to each other (entities, situations, people, real objects, companies, other comparable things, ...) and if available the aspect of the comparison. Please read the detailed Instructions and Examples!



## Task:

- [Instructions](#)
- [Examples and task definitions](#)
- [How to use the Labeling Tool?](#)

### Search Definitions

If you do not know the meaning of any terms in the texts below, write it here and search (opens in new window):

### 1. Label the Comparative Objects and (if available) the Aspect of the question:

#	Comparative Question	
1.	Are jewelleries in Hong Kong cheaper than in Singapore ?	<input type="radio"/> Question not comparative <input checked="" type="radio"/> Comparative Objects annotated
2.	why dose water produce more energy than oil when used as a fuel ?	<input type="radio"/> Question not comparative <input checked="" type="radio"/> Comparative Objects annotated

### 2. Tell us about yourself (**Only first TIME**):

Age	Country
<input type="text" value="Select..."/>	<input type="text"/>
Are you a native English speaker?	If you are not a native English speaker, what is your native language?
<input type="text" value="Select..."/>	<input type="text" value="English"/>

### (3. Your comments:)



Figure 3.7: The figure shows the template for the MTurk sequence-tagging task. The template has a demo area (top) with a video explaining how the the labelling tool works and the main task area (bottom) including the pilot questions. Note: The distance of the elements is displayed compressed to fit the figure.

for the workers, it is decided not to perform a contraction or a hyphen separation. In case it becomes necessary in future work, the separation can still be made, which might be easier than reversing an already fulfilled separation.

**Original text:**

Who's best elliot/melissa or j-lo? N.Y. can't beat Boston and isnt worth the large-scale attention...

**Tokenized text:**

"Who", "'s", "best" "elliot" "/" "melissa" "or" "**j-lo**" "?" "N.Y." "**can't**" "beat"  
"Boston" "and" "**isnt**" "worth" "the" "**large-scale**" "attention" "..."

For the tokenization of sentences the spaCy library <sup>25</sup> is utilized. The library provides natural language processing (NLP) with, for example, tokenization, lemmatization, part-of-speech tagging (POS) and named entity recognition (NER) for English texts and various other languages. SpaCy is chosen because of its easy use, its open source and because it allows the customization of its components. Listing 3.5 shows the Python code of the text pre-processing functions using the spaCy library. SpaCy provides general-purpose pretrained machine-learning models for the different languages. For the English language there are three sizes available. The script uses the medium sized model (Lines 1-4). Without any changes, spaCy would split words with hyphens and contractions into separate tokens. Therefore, the tokenizer rules are redefined in the *preProcessing* function (Line 6 and following).

The function receives a text as parameter upon call and returns the tokenized text as a list. Line 7 assigns a list of special cases, which should never be split, like "v.s." and "a.m.", to the tokenizer. The lists in Line 9-37 define cases that should be split by characters or more complex regular expressions. The definitions are taken from the original implementation of spaCy. Commenting the regular expression for infixes with hyphens prevents the separation of the words (Line 35). Lines 39-46 activate the new rules. Not defining any rules for splitting contractions prevents the default behaviour. In order to optimize the results of the tokenization, the incoming text (query) is altered (Lines 48-54). Tab stops and line breaks are removed, words directly connected to a question-mark are separated and on both sides of brackets a white space is added. As a last step, multiple white spaces are reduced to only one. These alternations do not take care of all possibilities and edge cases, but they cover frequent mistakes.

---

<sup>25</sup><https://spacy.io>

```

1 | #load spacy
2 | import spacy
3 | import en_core_web_md #English core model medium size
4 | nlp = en_core_web_md.load()
5 |
6 | def preProcessing(query):
7 |     nlp.tokenizer.rules = special_cases #Load special cases for daytime and and abrviation
8 |         like v.s.
9 |
10 |     prefixes = (
11 |         ["§", "%", "=", "", "", r"\+(?![0-9])"]
12 |         + LIST_PUNCT + LIST_ELLIPSES + LIST_QUOTES + LIST_CURRENCY + LIST_ICONS)
13 |
14 |     suffixes = (
15 |         LIST_PUNCT + LIST_ELLIPSES + LIST_QUOTES + LIST_ICONS
16 |         + ["'s", "'S", "s", "S", "", ""]
17 |         + [
18 |             r"(?<=[0-9])\+",
19 |             r"(?<=[FfCcKk])\.",
20 |             r"(?<=[0-9])(?:{c})".format(c=CURRENCY),
21 |             r"(?<=[0-9])(?:{u})".format(u=UNITS),
22 |             r"(?<=[0-9]{al}{e}{p}(?:{q}))\.".format(
23 |                 al=ALPHA_LOWER, q=CONCAT_QUOTES, p=PUNCT
24 |             ),
25 |             r"(?<=[{au}][{au}])\.".format(au=ALPHA_UPPER),
26 |         ])
27 |
28 |     infixes = (
29 |         LIST_ELLIPSES + LIST_ICONS
30 |         + [
31 |             r"(?<=[0-9])[+|-|*~](?=[0-9-])",
32 |             r"(?<=[{al}{q}])\.(?=[{au}{q}])".format(
33 |                 al=ALPHA_LOWER, au=ALPHA_UPPER, q=CONCAT_QUOTES
34 |             ),
35 |             r"(?<=[{a}]),(?=[{a}])".format(a=ALPHA),
36 |             #r"(?<=[{a}](?:{h})(?=[{a}])".format(a=ALPHA, h=HYPHENS),
37 |             r"(?<=[{a}0-9])[:<=>/](?=[{a}])".format(a=ALPHA),
38 |         ])
39 |
40 |     prefixes_re = compile_prefix_regex(prefixes)
41 |     nlp.tokenizer.prefix_search=prefixes_re.search
42 |
43 |     suffixes_re = compile_suffix_regex(suffixes)
44 |     nlp.tokenizer.suffix_search=suffixes_re.search
45 |
46 |     infix_re = compile_infix_regex(infixes)
47 |     nlp.tokenizer.infix_finder = infix_re.finditer
48 |
49 |     query = query.replace('\n', ' ')
50 |     query = query.replace('\t', ' ')
51 |     query = re.sub(r'(\w\w)\?(\w\w)', r'\1 ? \2', query)
52 |     query = query.replace('(', ' ( ')
53 |     query = query.replace(')', ' ) ')
54 |     query = query.replace(' ', ' ')
55 |     query = query.replace(' ', ' ')
56 |
57 |     doc = nlp(query)#work the query with spaCy
58 |     tokens = []
59 |     for token in doc:
60 |         if token.text != ' ':
61 |             tokens.append(token.text)
62 |     if len(tokens) == 0:
63 |         print("Zero token sentence detected!")
64 |     return tokens

```

Listing 3.5: Pseudocode of the text pre-processing functions for the human sequence-tagging project. The code loads the spaCy NLP library and redefines its tokenizer rules.

Line 56 is the execution of the spaCy tokenizer on the query and the following writing of the separated tokens to the return list. The overall data preparation script proceeds from then on in a similar way to the Python script of the classification task. Solely the export of the tokens is different, as each token gets its own ID within a sentence. The complete text, including the necessary HTML tags, is exported to the batch data file.

The third major part of the task preparation is to provide an analysis and a voting script for the completed sequence-tagging task. Compared to the classification task, the voting is more challenging. Instead of a voting process with 3 labels per sample, this task has seven labels per token and also labels per sample. The labels per sample are binary. In case that the worker tags at least one token with a label, the sample is considered comparative. If no token or sequence is labelled, the worker must choose to label the sample non-comparative. Within a sample text, each worker can tag as many single tokens as necessary with labels. Furthermore, the workers can tag tokens sequences with labels. In total seven labels for tokens exist. There are four comparative object labels (Obj1-Obj4), a shared comparative object label and a label for the aspect of a sentence. Implicitly, by not tagging a token, a “no label” is set. As a result, for each token a voting process on each of the seven labels needs to take place. To have a simple and correct voting process, a series of steps needs to be performed. The steps described below are implemented in a Python script with the necessary peripheral functions.

1. Loading the process data and accumulating the relevant data (WorkerID, work-time, annotated-sequence, sample-label) for each question from all workers.
2. Loading the original dataset and joining the original question text with the process data.
3. Dividing the worker’s annotations by label and merging or slicing the token IDs in order to group consecutive IDs.
4. Tokenizing the original question text with the tokenizer from the preparation script.
5. Calculating the vote per label type for a sample. The calculation is implemented for votes by majority (e.g 2 out of 3 workers) and votes by absolute majority (e.g.g 3 out of 3 workers).
6. Fetching the token texts of the vote from the original question and saving it with the voted sequences.
7. Aggregating convenience data out of the votes. For example, setting a boolean if a sample has an majority aspect or setting a boolean if a sample has successful votes for Obj1 and Obj2. Furthermore, a boolean-flag is set if one of the workers tags a sample as non-comparative.

8. Writing all aggregated data as an intermediate format (tab separated values) to be imported in Excel for further analysis.

### 3.5.2 Pilot

It is essential to find good workers for the sequence-tagging project in order to receive good results. Compared to the classification task, this annotation project is more detailed and requires a better understanding of the language, as well as some dedication to comprehend and to conduct the task. To find these dedicated good workers, a pilot study is performed with questions for which the answers are known and the workers can be evaluated on.

In order to select fitting questions for the pilot study, 10 samples were selected from the classification dataset for each source. The data consisted of comparative and a few selected non-comparative samples. Four people were asked to tag the 30 questions in a pre-pilot study. With the results from the pre-pilot, a guess on how difficult questions are for the workers could be made. This helped to choose 15 questions for the pilot study. Twelve questions are comparative and three questions are non-comparative by choice. To each question a category, which tries to estimate the difficulty of the sentence, is assigned. The goal is to have a set of easy, medium and hard questions for the pilot. The correct labels (gold labels) for the pilot study were agreed by three people familiar with comparative questions and the task of creating the dataset. The selected sentences with their gold labels and their difficulty can be found in Figure 3.8.

There are five questions in the set that are considered to be hard to label for the workers. Sentence 2 is considered of hard difficulty because the aspect of the sentence is split up into small pieces and the word “more” is an attribute of the aspect and not part of the aspect. Question 6 is considered of hard difficulty because of the shared object “resident” and the complexity of the sentence’s content. Question 10 is considered hard to label because of the shared object and the aspect of the sentence, which does not include “better”. Sentence 12 is part of the hard category, due to the sentence’s complexity and the split-up objects. And finally, Sentence 15 is considered of hard difficulty because “enjoy food” is an attribute of the aspect and not part of it. All non-comparative samples (Samples 5, 9, 14) are estimated with a medium or easy difficulty.

Five pilots needed to be conducted in order to find enough workers with a good labelling accuracy. The pilots were consecutive and the task was adapted for each new pilot as a measure to boost the performance of the workers. After each pilot, for each worker the F1 score was calculated sentence by sentence. This emphasized the importance of the correct annotation for each sentence’s tokens. To get a meaningful value, the mean of all F1 scores was calculated. The list below describes the pilots and the consecutive changes made:

## CHAPTER 3. CREATION OF COMPARATIVE QUESTION DATASETS

#	Sentence	Difficulty
1.	Are <b>jewelleries</b> in <b>Hong Kong</b> <b>cheaper</b> than in <b>Singapore</b> ?	easy
2.	why dose <b>water</b> produce more <b>energy</b> than <b>oil</b> when used as a <b>fuel</b> ?	hard
3.	Should I <b>Adopt</b> Or <b>Foster</b> ?	easy
4.	what is the <b>cost of living</b> difference between <b>arkansas</b> and <b>northern california</b> ?	easy
5.	How do you no if your makin love or havin s*x ?	medium
6.	What sort of <b>advantages</b> does a <b>resident</b> have living in a <b>US territory</b> versus a <b>US state</b> ?	hard
7.	Why do <b>rural areas</b> get more <b>snow</b> than <b>cities</b> ?	medium
8.	What is the <b>scientific technical</b> difference between a <b>soap</b> , a <b>shampoo</b> , and a <b>detergent</b> ?	easy
9.	What makes one person more flexible than the other	medium
10.	why <b>beer</b> tastes " better " in from <b>bottles</b> than <b>cans</b> .	hard
11.	Why does <b>homecooked food</b> similar to <b>fast food</b> not <b>taste</b> near as good as the fast food version ? Example; a homecooked burger vs a mcdonalds burger	medium
12.	What is the difference between a <b>3D movie</b> that you would see <b>with 3D glasses</b> in theaters and <b>3D CG</b> that is meant for viewing <b>without glasses</b> ?	hard
13.	Is <b>ripe fruit</b> worse for you than <b>unripe fruit</b> because ripe fruit is sweeter ?	easy
14.	Are there any games on the App Store similar to Tap Tap Revenge that are worth looking at ?	easy
15.	Do <b>dogs</b> enjoy <b>hot or cooked</b> food more than <b>uncooked food</b> ?	hard

Legend: Comparative-OBJ-1 Comparative-OBJ-2 Comparative-OBJ-3 Shared-OBJ Aspect

Figure 3.8: The figure shows the correct labels (gold labels) and the assigned category of difficulty for the pilot questions.

1. **Pilot:** A basic template without the “demo” area of the page was used. The pilot was open to everyone and, additionally, the workers that participated in the classification task were invited.

**Result:** 35 workers just marked “non-comparative” for every sentence in order to just receive the reward that was paid for the task. 5 workers sent an empty form. Only 6 workers achieved a mean F1 score higher than 0.8 on the comparative questions and 21 workers reached a F1 score of 0.7 or higher.



2. **Pilot:** The measures against sending an empty form were improved, the “No Label” button was added to the annotation overlay and the pop-up explaining the labelling tool was added.  
**Result:** 45 workers just marked “non-comparative” for every sentence. Only 3 workers achieved a mean F1 score higher than 0.8 on the comparative questions and 15 workers reached a F1 score of 0.7 or higher.
  
3. **Pilot:** The “demo” area was added on the top of the page in order to avoid the workers having problems with the use of the annotation tool.  
**Result:** 24 workers just marked “non-comparative”. This was less than in the first two pilots. It might be that the “demo” area helped. Anyway, the F1 scores did not improve: 3 workers are better than an F1 score of 0.8 and 19 workers are better than an F1 score of 0.7 on comparative samples.
  
4. **Pilot:** In this pilot, only workers that participated in the classification task, who had not already participated in Pilot1-3 were approved and invited to participate via e-mail.  
**Result:** No workers participated.
  
5. **Pilot:** In order to simplify the task, the non-comparative questions and the option to select “not comparative” were removed.  
**Result:** The number of workers with higher scores improved slightly. 10 workers achieved an F1 score higher than 0.8 and 34 workers higher than 0.7. But the improvement might not necessarily be connected to the the removal of non-comparatives, as roughly 25-30% of the annotations can be considered random annotations.

The pilots show that the identification of comparative objects and comparative aspects is a difficult task. Table 3.32 provides a detailed overview of the worker distribution on F1 scores. Just 3 out of 400 participants have a mean F1 score of 0.85-0.9 on the comparative samples and only 22 workers achieve a score higher than 0.8. Excluding those workers who scored less than 0.4 points, most workers have an F1 score between 0.6 and 0.7. The workers reach better F1 scores when identifying comparative objects instead of aspects. The peak of the F1 distribution is at 0.7-0.75 for comparative objects and at 0.6-0.65 for the aspects of the comparison. The evaluation of the questions that had been assigned to the category “hard” shows that only 4 workers have an F1 score between 0.8-0.85 and no other worker is better than this. Aggregating the F1 scores for each question shows that the assigned classes of difficulty most of the cases. For example, Question 1, which is assigned with “easy”, reaches a mean F1 score of 0.75 and Question 11 (hard) reaches a mean F1 score of 0.43. For both “hard” questions, Question 2 (mean F1 0.58) and Question 6 (mean F1 0.64), the results were better than expected. For question 12, which

was assigned “easy” before the pilots, the results indicate that it was more difficult (mean F1 0.53) for the workers to tag the right sequences. Evaluating the workers’ classification skills of the three non-comparative samples shows that 46% of the workers have classified all three questions correctly as non-comparative. 20% of the workers answered 2 questions correctly, 22% only one question and 12% did not give any correct answer. In total, workers from 13 different countries participated in the pilots. 50% of the workers come from the United States of America and 78% come from countries that have English as an official language <sup>26</sup>. 71% of the workers consider English as their native language. The biggest group of workers is between 30 to 39 years old (30%), directly followed by 25 to 29 years old workers (29%). The mean time to tag one sample was 48.3 seconds, which is 28.6 seconds longer than in the classification pilots.

Sample	Mean F1 scores										
	<0.4	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85
		-0.45	-0.5	-0.55	-0.6	-0.65	-0.7	-0.75	-0.8	-0.85	-0.9
Comp	148	7	16	13	24	48	48	34	33	19	3
Hard	190	32	35	51	32	24	14	8	3	4	-

Table 3.32: The table shows the number of workers sorted by their mean F1 score on identifying comparative objects and aspects. The data is accumulated from all 5 pilots.

### 3.5.3 Main phase

The main phase of the human sequence-tagging task uses the workers from the pilot phase to annotate the comparative data from the classification task. The goal is to provide sequence labels for 4,260 comparative samples. Each sample is annotated by three unique human workers. The sequence-tagging project is run in batches. The data for the batches is evenly split using a third from each source per batch (Yahoo, NSQ and eli5). The human classification began with a test batch containing only 180 samples. The batch had 10 samples per HIT and, therefore, 18 unique HITs and 60 samples from each source. All workers with an F1 score higher than 0.8 in the pilot study could participate. In total, 22 workers were unlocked for the task. The purpose of the first small test batch was to validate the process of the HIT publication and the unlocking of workers. Furthermore, the batch was run to test the post-processing of the annotated data. For each finished batch an analysis and voting process was conducted with the Python script explained in Section 3.5.1.

<sup>26</sup>USA, India, Canada, United Kingdom and Australia.

In total, 5 batches were conducted on the Mechanical Turk platform, the first test batch and 4 batches with 1,020 samples per batch. In the analysis of the batches no irregularities regarding the workers annotations could be observed. The average time per question was 49.9 seconds in the first batch and it decreased to 22-32 seconds in the following batches.

### 3.5.4 Dataset statistics

This section provides statistics about the final dataset as a result of the human sequence annotation task on comparative questions. The data of the 5 batches was collected and post processed by means of a Python data collector script. The post-processing steps for the generation of the final dataset include:

1. Loading the sequence annotation data batches.
2. Merging the annotation data with the results of the human classification task in order to include already known information.
3. Exporting the non-comparative flagged samples.
4. Excluding samples that have more than two non-comparative votes.
5. Reducing the dataset to samples that have only two comparative-objects (Obj1 and Obj2).
6. Generating statistics.
7. Excluding the gold labelled samples from the data.
8. Exporting the dataset with three different formats:
  - (a) Line-separated: Each token is in one line with its tab-separated sequence label. The samples are split by an empty line.
  - (b) Multi-info: Each line holds information about one token. The original question-ID, the token, the POS-tag and the sequence label. Samples are not separated, but they can be identified by the same question-ID.
  - (c) Flair-format: As a convenience method, the data is split up into train, development and test sets for machine learning.

All these steps are mandatory in the post-processing, except for the exclusion of the gold labelled data (Step 7). Step 3 exports the non-comparative flagged samples that are used to refine the comparative classification dataset in Section 3.4. In total, 428 samples were tagged as non-comparative by at least one worker. Steps 4 and 5 exclude samples that are not helpful for the final dataset. Step 4 excludes samples that have a majority vote on non-comparative. In Step 5, samples that do not have two comparative objects are excluded. This might happen because the voting was inconclusive or because the workers did not tag a second object. Furthermore, samples that have more than two objects are excluded so that the data fits the definition of a comparative question. These measures reduce the dataset to 3,998 samples. With these samples the following statistics are generated. 278 (6.9%) out of the 3,998 samples have one vote for non-comparative. 3,440 (81%)

of the samples have two objects on which all three workers agreed. 1,001(23%) samples have an aspect on which all three workers agreed and 2,454 (57%) samples have an aspect on which the majority agreed. Table 3.33 shows the distribution of samples that only have comparative objects, objects and aspects and both combined with shared objects. It can be seen that comparative questions with objects and aspects is the most common configuration with 44%, followed by sentences that only contain two objects (32%).

OBJ1&2	Aspect	Shared	Counts	Percentage (%)
True	False	False	1298	32
True	False	True	301	8
True	True	False	1755	44
True	True	True	644	16

Table 3.33: The table shows the number of samples by the labels that are contained in the sample.

In order to rate the worker’s annotation quality, 420 (10%) samples of the data were randomly picked and manually annotated. This way gold labels for the sequence-tagging dataset were received. 25 out of the 420 samples are non-comparative and 10 do not have exactly two objects. This results in 385 gold labelled samples. Table 3.34 shows the evaluation of the main task with the gold labelled data. For the comparative objects the workers achieved a high F1 score of 0.88 and 0.89. The comparative aspect has an average F1 score of 0.6 with a higher precision of 0.72. The shared objects turn out to have the lowest F1 score of 0.49 and only reach a precision of 0.42. The micro F1 score, calculated over all classes, is 0.8. These metrics further indicate that it is easier for the workers to identify the comparative objects than the aspects. The shared objects might take a special position because they are not only more complex from a language point of view, but also less represented in the data (24%).

	Precision (%)	Recall (%)	F1 score
OBJ-1	87	90	0.88
OBJ-2	86	93	0.89
SHARED	42	60	0.49
ASPECT	72	52	0.60
micro avg	81	80	0.80

Table 3.34: The table shows the measures of the workers’ performance from the sequence-tagging task, which was evaluated with 420 gold-labelled samples.

The next chapter will use this sequence-tagging dataset and the classification dataset from Section 3.4, which were created with the help of human workers, in machine-learning classification and sequence labelling tasks.

## Chapter 4

# Experiments with comparative question datasets

This chapter describes the experiments that were conducted with the two comparative question datasets, which were created in Chapter 3. Both datasets serve a different purpose. The first dataset is a set of sentences collected from QA web-platforms and provides labels for classifying comparative questions. The experiments on classifying comparative questions with the help of machine learning and neural networks is described in Section 4.1. The second dataset is a set of comparative questions and provides labels for comparative objects and aspects. The experiments of extracting the objects and aspects from comparative questions is described in Section 4.2. The results of both experiments will be discussed in Section 4.3 and both experiments will be joined together in a comparative extraction pipeline. The pipeline will be used in a web app with a user interface and an API in Section 4.4. The aim of the experiments is to show the benefit of the datasets created and to provide a base for future research and optimization. This chapter does not aim to propose state-of-the-art implementations of the corresponding research problems, but it will aim to provide a good baseline for research with the datasets.

### 4.1 Classification of comparative questions

This section describes the experiments with the data of the first crowd sourcing task (see Section 3.4). The goal is to provide a machine classifier for comparative questions. In the experiments, feature-based machine learning and neural machine learning will be utilized to fulfil the task. Section 4.1.1 provides a description of the different tools, methods and the program structures used in the experiments. Section 4.1.2 provides information about the training of the classifiers, the parameters and the data that was utilized. Furthermore, Section 4.1.2 presents the results of the classification experiments.

### 4.1.1 Experimental setup

In order to conduct the experiments, two platforms were used. The first platform is a local instance of Python with Jupyter Notebook<sup>1</sup>. Jupyter Notebook is a web application that allows to collaborate on documents and program code. The tool is structured by cells of program code and their results. It supports the interactive execution of the separate code blocks within one file. This allows a better handling of program changes during the development. For example, the processing of data can be repeatedly changed and executed while the heavy and time-intensive loading of data is only executed once. The second platform is Google Colab<sup>2</sup>, a cloud-based instance of Jupyter Notebook. Google Colab can be used without a setup and provides free access to computing resources such as powerful Graphics Processing Units (GPU). The local instance of Jupyter Notebook is utilized for the machine-learning algorithms that are feature-based. Google Colab is utilized for the computationally and resource-intensive neural machine learning.

#### 4.1.1.1 Feature-base machine learning

The local environment is used for conducting experiments with a feature-based machine-learning classification of the comparative samples. To perform the experiments in the local environment, the tools of scikit-learn<sup>3</sup> are utilized. Scikit-learn is an open-source machine-learning library for Python and it provides various implementations of supervised classification algorithms, for example, the support-vector machine (SVM) model or the logistic regression model. Scikit-learn also provides a feature called “pipeline”, that serves the purpose to sequentially assemble different machine learning steps. This sequential assembly allows parts of the pipeline to be exchanged or run with different parameters. For example, the classification algorithm at the end of the pipeline can be easily exchanged. This scikit-learn feature is utilized to implement a classification script, which can evaluate multiple classifier algorithms and feature sets in a row for the same data. The script can be divided into the following functional parts:

1. Loading the data, the pre-processor tool (spaCy) and the Global Vectors for Word Representation (GloVe)<sup>4</sup> embeddings.
2. Pre-processing the data with spaCy.
3. Stratified n-fold splitting of the data into train and test folds.
4. Selection of the fold combination.

---

<sup>1</sup><https://jupyter.org/>

<sup>2</sup><https://colab.research.google.com>

<sup>3</sup><https://scikit-learn.org/>

<sup>4</sup><https://nlp.stanford.edu/projects/glove/>



5. Definition of transformers for the data in order to generate features from the text. For example, the term frequency–inverse document frequency (tf-idf) vectorizer.
6. Concatenating the results of the different features in a scikit-learn feature union.
7. Setting up the scikit-learn pipeline with the feature union and the different classifiers.
8. Execution of the pipeline with three options:
  - (a) Random grid search with cross-validation over a set of parameters (not implemented for all classifiers).
  - (b) Exhaustive grid search with cross-validation over a set of parameters.
  - (c) Running all models with one parameter set and prediction against the test fold.
9. Calculating the mean F1 score for each model over all folds.

Overall, the script provides a way to perform a nested cross-validation in order to find a reliable method for classifying the comparative questions. In general, the set of open variables is very high in this scenario, since the pre-processing decisions, the feature selection and the parameters of the features, as well as the classification algorithms and their hyper-parameters are open for optimization and selection. The script implements various options for all these variables. The experiments do not attempt to explore every possible variable option, but to focus at some point on promising models and parameters.

Step 1 of the script focuses on loading all resources, like the spaCy model for the English language and the GloVe word vectors (6B.50d) that were trained on Wikipedia and the Gigaword<sup>5</sup> corpus. The second step deals with pre-processing the data. In this step, the sentences are lowercased, tokenized and part-of-speech tags (POS), as well as detailed-part-of-speech tags (DTAG) are generated with spaCy. Additionally, the inflected forms of tokens can be grouped together (lemmatized), the stop words and the punctuation can be removed and the pronouns can be replaced by the “-PRON-” identifier. The separate information is saved for each sentence and also combined as “token + POS”, for example, “canVERB”. Step 3 performs the split of the data into train and test sets. In this step, the scikit-learn function “StratifiedKFold” is utilized, in order to shuffle the data and to generate 5 folds, while preserving the percentage of samples for each class. Step 4 marks the start of the outer loop of the cross validation by selecting one fold as test set and the other folds as trainings set.

Step 5 defines the various transformers as separate scikit-learn pipeline objects. Three groups of transformers are implemented: tf-idf transformers, n-gram transformers and

---

<sup>5</sup><https://catalog.ldc.upenn.edu/LDC2011T07>

transformers for embeddings. The tf-idf and the n-gram transformers use scikit-learn functions and can be parametrized on execution of the pipeline. As for the tf-idf two vectorizers are implemented: one uses the processed text as input and the other one uses the combined “token + POS”. For the n-gram transformers, a count vectorizer is applied to count the frequency of a word or token. The transformers are set up for the processed text, the tokens, the POS tags and the DTAGs as inputs. Unigrams (single words) are counted by default. This parameter can be changed on execution to arbitrary values for the n-gram size. The GloVe embeddings have dedicated vectorizers, which generate the mean or the sum of the GloVe word vector representations of a sentence. Step 6 concatenates the results of the selected features. This union of features can be fed into an estimator. The estimators (classifiers) are set up in Step 7. Various algorithms provided by scikit-learn are utilized as classifiers. Some of these have already proved to generate good results in the research of Panchenko et al. [49] and Bondarenko et al. [51]. Table 4.1 shows the classifiers that are utilized in the local environment. All models are feature-based machine-learning classifiers, except for the multi-layer perceptron (MLP) which is a neural network. The support-vector machine (SVM) is by default operated with a linear kernel and the SGD classifier works by default with a linear SVM.

Type	Implementation (Source)
Support-vector machine (SVM)	Support Vector Classifier (scikit-learn)
Logistic regression	Logistic Regression Classifier(scikit-learn)
Random forest	Random Forest Classifier (scikit-learn)
Extra Trees (Etree)	Extra Trees Classifier (scikit-learn)
Stochastic gradient descent (SGD)	SGD Classifier (scikit-learn)
Naïve Bayes (NB)	Bernoulli NB (scikit-learn)
XGBoost	XGBoost (XGBClassifier)
Multi-layer perceptron (MLP)	MLPClassifier (scikit-learn)

Table 4.1: The table shows the classifiers utilized in the local environment with their implementation source. A list of all ML model sources can be found in the appendix in Table 5.

Step 8 is the inner part of the nested cross validation. In this step, the grid search for parameters are performed. The grid search is executed on the training data and it splits the data into sub-folds in order to find the best parameters by cross-validation. To find the parameters, an exhaustive grid search can be used for SVM, random forest, logistic regression, SGD and Etree. Random grid search functions are implemented for SVM, Random Forrest, Etree and MLP. This completes the inner part of the cross validation. The parameters found by the grid search can be set for each classifier to be trained on

the complete training data and evaluated against the test fold. The evaluation F1 scores are saved for each classifier and each test fold. The mean F1 score for each classifier is calculated in Step 9. Although it does not provide the best possible results for the data, because the parameters found only apply to one fold, the nested cross validation allows to indicate good features and classifiers for the task.

#### 4.1.1.2 Neural machine learning

The Google cloud platform Colab is utilized to conduct the compute heavy experiments that use artificial neural networks. The approach to neural networks is twofold. One approach executes a fine-tuning of a BERT model [52] and the other approach uses the Flair framework (Version 0.4.5). Flair is an open-source natural-language-processing (NLP) framework [80]<sup>6</sup>. The framework uses the open-source machine-learning framework PyTorch<sup>7</sup>. Flair was first introduced by Zalando Research<sup>8</sup>. Flair enables the the user to perform sequence labelling and text classification on text by means of the framework's state-of-the-art NLP models. Moreover, Flair acts as a text embedding library by supporting the use and the combination of various word and document embeddings like GloVe, Flair or ALBERT embeddings. The ALBERT model is the state-of-the-art model for tasks like question answering on the SQuAD dataset and reading comprehension on the RACE dataset (see related work in Section 2.1). Language models like ALBERT can be accessed through the Huggingface Transformers<sup>9</sup> library, which provides pretrained models for more than 100 languages. The Huggingface Transformers enable the second approach on neural networks, a direct fine-tuning of a BERT model. With this method, a pretrained model can be loaded and fine-tuned for the task of classifying comparative sentences.

In order to work with Google Colab, three steps are necessary, which are not needed in a local environment. Firstly, hardware instances need to be reserved. The CPU and the system memory resources are automatically allocated with the execution of the script. GPU resources must be reserved in program code and then assigned to PyTorch. Figure 4.1 shows the program code and the result of the GPU allocation. Secondly, software packages that are necessary for the project, but not set up by default, need to be installed on the hardware instance. For the classification project, the Flair and the Huggingface Transformers packages need to be installed. Thirdly, a connection to Google Drive should be initiated. Drive is used to store the Colab files and acts as a long-term storage for Colab. Through the Drive connection, the input datasets can be loaded and the final models can be saved. These three steps are necessary every time the hardware instance

---

<sup>6</sup><https://github.com/flairNLP/flair>

<sup>7</sup><https://pytorch.org/>

<sup>8</sup><https://research.zalando.com/>

<sup>9</sup><https://github.com/huggingface/transformers>

```

14 # If there's a GPU available...
15 if torch.cuda.is_available():
16     # Tell PyTorch to use the GPU.
17     device = torch.device("cuda")
18     print('There are %d GPU(s) available.' % torch.cuda.device_count())
19     print('We will use the GPU:', torch.cuda.get_device_name(0))
20 # If not...
21 else:
22     print('No GPU available, using the CPU instead.')
23     device = torch.device("cpu")

Found GPU at: /device:GPU:0
There are 1 GPU(s) available.
We will use the GPU: Tesla P100-PCIE-16GB

```

Figure 4.1: The figure shows the allocation of resources in Google Colab. In this example the system is assigned a Tesla P100 GPU.

is initiated. As long as the session is not terminated, multiple executions of the script require these three steps only once.

Using Flair for text classification is fairly easy. The first two steps load the dataset and generate a label dictionary (see Figure 4.2). In this example, the classification dataset with dedicated gold label test data and additional 6.000 non-comparative samples is loaded. For the comparative question data, the classification is binary and, therefore, the label dictionary only contains “comparative” and “non-comparative” as labels.

Figure 4.3 shows the initialization of the word embedding. Flair supports to load “classic” word embeddings like GloVe and it also provides various embeddings for different languages created by its developers. Trough the Huggingface Transformers library, Flair supports transformer-based architectures like ALBERT. In the example (see Figure 4.3), the Flair multi-language embeddings and the large ALBERT embeddings are utilized. All

```

1 # 1. load corpus
2 columns = {0: 'label', 1: 'text'}
3 # this is the folder in which train, test and dev files reside
4 data_folder = './'
5 # init a corpus using column format, data folder and the names of the train, dev and test files
6 corpus: Corpus = CSVClassificationCorpus(data_folder, columns, skip_header=True, delimiter='\t')
7 print(corpus)
8 # 2. create the label dictionary
9 label_dict = corpus.make_label_dictionary()

2020-05-11 07:18:39,694 Reading data from .
Corpus: 12985 train + 2291 dev + 2110 test sentences
2020-05-11 07:18:39,876 Computing label dictionary. Progress:
2020-05-11 07:18:48,210 [b'__label__comparative', b'__label__non_comparative']

```

Figure 4.2: The figure shows the code to load the text corpus and to generate a label dictionary with Flair in Google Colab.

```

1 3. #initialize embeddings
2 glove_embedding = WordEmbeddings('glove')
3 flair_forward_embedding = FlairEmbeddings('multi-forward')
4 flair_backward_embedding = FlairEmbeddings('multi-backward')
5 bert_embedding = BertEmbeddings(bert_model_or_path="albert-large-v2")

```

Figure 4.3: The figure shows the code to initialize several embeddings with Flair in Google Colab.

embeddings are automatically downloaded on the first execution of the script.

The loaded embeddings can be combined to build a document embedding. This way, one embedding for the entire text is generated out of the word embeddings. In the example of Figure 4.4, a document embedding that trains a recurrent neural network (RNN) over the word embeddings is utilized. This example uses ALBERT and GloVe embeddings. This easy way of loading and combining different embeddings allows the evaluation of multiple combinations of embeddings. With the document embedding it is possible to create a classifier (see Figure 4.4 Line 4).

```

1 # 4. initialize document embedding by passing list of word embeddings
2 document_embeddings = DocumentRNNEmbeddings([glove_embedding, bert_embedding], rnn_type='LSTM')
3 # 5. create the text classifier
4 classifier = TextClassifier(document_embeddings, label_dictionary=label_dict, multi_label=False)

```

Figure 4.4: The figure shows the code to initialize document embeddings out of several word embeddings and to create the Flair classifier.

As a last step, the training class needs to be initialised with the classifier and the corpus. With the definition of hyper-parameters, like the learning rate, the training can get started. The training and the final evaluation against the test set are done automatically. Once the training is finished, the results and the best model can be saved in Google Drive.

With Flair the embeddings of transformers like BERT can be utilized. In order to directly

```

1 # 6. initialize the text classifier trainer
2 trainer = ModelTrainer(classifier, corpus)
3 # 7. start the training
4 trainer.train('./',
5             learning_rate=0.05,
6             mini_batch_size=32,
7             anneal_factor=0.5,
8             patience=5,
9             max_epochs=150,
10            embeddings_storage_mode='gpu')

```

Figure 4.5: The figure shows the code to execute the model training with Flair.

train a BERT model, the Huggingface Transformers library can be utilized. As training a complete transformer is computational expensive, pretrained BERT models are available, which then are fine-tuned for the task at hand. Chris McCormick and Nick Ryan published a tutorial on how to fine-tune BERT with PyTorch [81]. The tutorial shows how to fine-tune a model for sentence classification on an open dataset with the Huggingface library and PyTorch. This is exactly what is needed for the comparative classification task. Therefore, the tutorial and the code is used to create the fine-tuning for the comparative question classification task. The code of McCormick et al. is extended to load and fit the comparative data.

### 4.1.2 Training and results

The experiments of classifying comparative questions are split into 4 parts. The first part was conducted in an very early stage in order to get a baseline for the task. The other three series of experiments are built upon each other. The first series had the aim to evaluate the data pre-processing steps. The second series of experiments was conducted to find the best fitting classification algorithm. And the third series of experiments was dedicated to analyse the impact of the input data on the classification results.

The development and training of classifiers was started during the first human annotation task. The first baseline was generated after the fourth human annotation batch was run. This means that there were only 3,240 samples in the dataset. At that point, the machine-learning script was only implemented partially. The first training was done with a linear SVM and tf-idf as a single feature on the text. This setup reached an F1-Score of 0.646 and will serve as the baseline of the classification task.

Once the human annotations were completed and the local machine-learning script was implemented as described in Section 4.1.1, the first series of experiments began. The goal of the first series was to evaluate the data pre-processing features. There are three pre-processing options that were evaluated. Firstly, to exclude or keep stop words. Secondly, to exclude or keep punctuation. And thirdly, to convert all pronouns to “-PRON-” or leave them as original text. Table 4.2 shows the F1 scores for some selected combinations. Except for the logistic regression (logReg) the best results can be achieved by keeping stop words in the text, removing punctuation and converting all pronouns into the “-PRON-” variable. This pre-processing configuration was utilized for the following machine-learning experiments.

With the pre-processing in place, the second series of experiments was started. The aim

Model	Stop words	Punctuation	-PRON-	F1 score
SVM	Yes	Yes	No	0.6075
logReg	Yes	Yes	No	0.5991
MLP	Yes	Yes	No	0.5974
SVM	Yes	No	No	0.6126
logReg	Yes	No	No	0.6035
MLP	Yes	No	No	0.6010
SVM	Yes	No	Yes	<b>0.6127</b>
logReg	Yes	No	Yes	0.6044
MLP	Yes	No	Yes	<b>0.6074</b>
SVM	No	No	Yes	0.6067
logReg	No	No	Yes	<b>0.6154</b>
MLP	No	No	Yes	0.6042

Table 4.2: The table shows combination of pre-processing features and the corresponding F1 score on a selection of models. “Yes” and “No” indicate if the feature is a part of the data.

was to find the best fitting classification model for the task. To evaluate the machine-learning classifiers, grid searches were run for various models. Exhaustive grid searches were run for SVM, random forest, logistic regression, SGD and Etree. For SVM, random forest, Etree and MLP, random grid searches were run. The search spaces can be found in the appendix in Listings 1 - 6. Each search used three parameter types that could vary. Firstly, the model specific hyper parameters were searched, for example, the kernel of an SVM. Secondly, single features were turned on or off, for example, the n-grams of the POS-tags. Thirdly, the parameters of the features were varied, for example, the n-gram range and the maximum considered of n-grams. Because grid searches are extremely resource and time intensive, not all parameter combinations could be tested at the same time and the number of maximum iterations for the random grid search ranged between 100 and 250. The grid searches indicate a good performance for a linear SVM and SGD with hinge loss (SVM), as well as logistic regression with n-gram features of the tokens, the POS-tags and the DTAGs. The best parameters of each model were used to train the models and test against five revolving test folds. The input data for classification was the complete human annotated data without further post-processing (AllData). The test folds contained around 2,000 samples. The F1 test scores of the models were recorded for each fold. The mean F1 score was calculated after completion of the experiment. Table 4.3 shows the mean macro F1 scores of the models utilized in the first series of experiments <sup>10</sup>.

<sup>10</sup>The training parameters for each model can be found in the appendix in Listing 7

SGD has the highest score (0.7936), directly followed by the linear SVM (0.7903). This is logical, as the SGD is trained with a linear SVM. The logistic regression and MLP models also achieve good results. In general, all evaluated models exceed the baseline by far.

Model	SGD	SVM	Log. Regression	MLP	XGBoos	Bernulli NB	Rand. Forrest	Etree
F1 score	0.7936	0.7903	0.7895	0.7803	0.7752	0.7537	0.7438	0.7199

Table 4.3: The table shows the mean F1 scores of the machine-learning models from the first series of experiments.

To evaluate the neural machine learning, the Flair framework was utilized. The experiment utilized three stacked embeddings: GloVe Word embeddings, Flair “news-forward-fast” and Flair “news-backward-fast”. Both flair embeddings support the English language and were trained on the 1-billion-word corpus<sup>11</sup>. Based on the word embeddings, document embeddings were generated by an RNN (DocumentRNNEmbeddings) with a hidden layer size of 512. The training was conducted with the values shown in Table 4.4 Experiment 2. The training used the complete human annotated data without further post-processing (AllData). Both, the development set and test set hold 15% of the data (1,515 samples), which leaves 7,073 samples for the training. It took 9 hours to train the model on CPU and a macro F1 score of 0.7922 was reached, placing it between the SGD and SVM classifier. The same setup was used with GloVe and ALBERT embeddings (albert-base-v2) instead of the GloVe and Flair embeddings. Only the learning rate was changed to 0.05 (see Table 4.4 Experiment 3). The model took 6 hours for training and reached a macro F1 score of 0.8104. Therefore, the ALBERT embeddings exceeded the performance of the Flair embedding by 0.018 points. Changing the document embedding RNN type from gated recurrent unit (GRU) to long short-term memory (LSTM) and training on GPU increased the macro F1 score to 0.8405 and reduced the training time to one hour (see Table 4.4 Experiment 4). In the fifth experiment, the fine-tuning of a BERT model was carried out (see Table 4.4). A BERT model (bert-base-uncased), pretrained for sequence classification, was utilized for the experiment. The model was trained for 3 epochs and operated with the same input data as the experiments before. The fine-tuned model achieves a macro F1 score of 0.8452, which is a slightly better result (+0.0047) than the one from the ALBERT embeddings in Flair.

The last series of experiments had the purpose to study the impact of the dataset on the classification performance and to identify the best post-processed dataset for the classifica-

<sup>11</sup><https://github.com/ciprian-chelba/1-billion-word-language-modeling-benchmark>



#	Dataset	Framework	Embeddings / Model	Document embedding	Learning rate	F1 score (macro)
2	AllData	Flair CPU	GloVe news-forward-fast news-backward-fast	GRU	0.1	0.7922
3	AllData	Flair CPU	GloVe <i>albert-base-v2</i>	GRU	<i>0.05</i>	0.8104
4	AllData	Flair GPU	GloVe albert-base-v2	<i>LSTM</i>	0.05	0.8405
5	AllData	<i>Huggingface Transformers</i>	BERT-base-uncased	-	0.00002	<b>0.8452</b>

Table 4.4: The table shows the macro F1 scores and the configuration of the experiments with neural networks.

tion. Due to the simplicity of the Flair framework and the good results with the ALBERT embeddings, it was decided to proceed with the setup from Experiment 4. Another point in favour of using Flair are the already integrated methods that save and load trained models and the available methods that predict single sentences with the models. These will be useful to fulfil the thesis goal of providing a classification of comparative sentences to systems like the Comparative Argumentative Machine. For this series of experiments the embeddings were changed to the ALBERT large embeddings (albert-large-v2). The hyper parameters did not change in the series of experiments, only the dataset was changed. The first experiment of the series established a baseline for the ALBERT large embeddings with the untouched complete dataset (AllData). The macro F1 score for the sixth experiment is 0.8267 and, therefore, 0.014 points lower than in Experiment 4 (see Table 4.5). For the following experiments the data samples that have gold labels were excluded from the training data. This left 9,330 samples for the training and for the development set. The development set was chosen to contain 15-16% of the data. The test set contained the 2,111 samples that were manually gold labelled. Table 4.5 shows the configurations and the results from the experiments. In Experiment 7, only the test data is excluded, which results in a macro F1 score of 0.8621. This is the highest value so far, but it can also be determined that the F1 score for the comparative samples decreased to 0.813. Experiments 8 and 9 show that the post-processing methods of relabelling comparatives, which were made against open groups, and the revote with data from the second human annotation task increase the macro F1 performance and the performance on comparative samples. Experiment 10 combines both post-processing methods in the data, which results in another increase of the F1 scores. The experiments 11 to 14 extend the dataset with non-comparative data samples, which were filtered out in the data-mining process. Both, the experiment with 1,500 and the one with 3,000 samples increase the macro F1 score, as well as the F1 score for comparatives. The experiment with 6,000 additional

non-comparative samples has slightly worse values than the one with 3,000 extra samples. Experiment 14, which adds 9,000 non-comparative samples, could not be conducted due to GPU memory restrictions. The training of this experiment exceeded 16GB of memory.

#	Dataset	Train+Dev Samples	Test Samples	F1 score (macro)	F1 score (comp)	F1 score (non-comp)
6	AllData	8,590	1,516	0.8267	0.8299	0.8435
7	AllData ExtraTest	9,330	2,111	0.8621	0.813	0.9112
8	AllData ExtraTest NoOpenGroup	9,330	2,111	0.8642	0.8164	0.912
9	AllData ExtraTest Revote	9,278	2,111	0.8672	0.8207	0.9136
10	AllData ExtraTest NoOpenGroup Revote	9,278	2,111	0.8731	0.8273	0.9188
11	AllData ExtraTest NoOpenGroup Revote NonCompData1.5k	10,778	2,111	0.8742	0.8294	0.9189
12	AllData ExtraTest NoOpenGroup Revote NonCompData3k	12,278	2,111	<b>0.8760</b>	<b>0.8316</b>	<b>0.9204</b>
13	AllData ExtraTest NoOpenGroup Revote NonCompData6k	15,278	2,111	0.8753	0.8311	0.9195
14	AllData ExtraTest NoOpenGroup Revote NonCompData9k	18,278	2,111	-	-	-

Table 4.5: The table shows the F1 scores using the different classification datasets. Experiment 14, which adds 9,000 non-comparative samples, could not be conducted due to GPU memory restrictions. The parameters were not changed in these experiments. Parameters (Framework: Flair with DocumentRNNEmbeddings (LSTM). Embeddings: GloVe + ALBERT-large-v2. Training: Batchsize 32, learning rate 0.05.)

## 4.2 Extracting comparative objects and aspects

This section describes the experiments with the data annotated in the second crowd sourcing task (see Section 3.5). The goal of the experiments is to provide a sequence classifier for comparative sentences. In the experiments, feature-based machine learning and neural networks are used to classify comparative objects and comparative aspects. Section 4.2.1 provides a description of the different tools, methods and the program structures utilized in the experiments. Section 4.2.2 provides information on the training of the classifiers, the parameters and the data that was used. Furthermore, Section 4.2.2 presents the results of the classification experiments.

### 4.2.1 Experimental setup

Similar to the comparative question classification experiments, the experiments on sequence classification are conducted on two platforms. For the feature-based machine-learning experiments, a local Python environment with Jupyter Notebooks is set up. For the neural machine learning, Google Colab is utilized again as a cloud instance based on Jupyter Notebooks. The local environment utilizes the classifiers and tools provided by the scikit-learn library. The ML classifiers are implemented in a Python script, which can be separated into four functional parts.

1. Loading the multi-info dataset. The dataset contains the tokens, POS-tags and the sequence labels.
2. Generating features with the tokens and the additional information available.
3. Splitting the data into a train and a test set.
4. Running machine-learning models with a set of features and a prediction against the test set.

The first step loads the multi-info dataset and groups the words with the corresponding POS-tag and the comparative sequence tag together as a Python tuple. The tuples that belong to the same sentence are structured in a Python list. The second step generates features out of these lists of tuples. Listing 4.1 shows the pseudocode for the generation of features out of one token. There are three types of features: features based on the current token, features generated from tokens previous to the current one and features generated from the token following the current one. The features are: the lower cased word itself, the POS-tag, a boolean indicating if the token is a digit and a boolean indicating if the token is a titlecased word (Lines 8-11). Further to these features, the endings of the word can be extracted (Lines 14-17). The same features are extracted for following and previous words.

This generates a regional context for each token in a sentence. Step three splits the data into a train set and a test set. The test set is 15% the size of the input data. The last step allows to run various sequence classifiers on the data. A SGD model with linear SVM, a Multinomial Naïve Bayes model, a passive-aggressive classifier model, a Perceptron model and a conditional random fields (CRF) model are implemented in the script. A list of all ML model sources can be found in the appendix in Table 5.

```

1| def word2features(sent, i):
2|     word = sent[i][0]
3|     postag = sent[i][1]
4|
5|     ###Word Features
6|     features = {
7|         'bias': 1.0,
8|         'word.lower()': word.lower(),
9|         'word.istitle()': word.istitle(),
10|        'word.isdigit()': word.isdigit(),
11|        'postag': postag,
12|    }
13|
14|    if len(word) > 3:
15|        features.update({'wordend[-3:]': word[-3:]})
16|    if len(word) > 2:
17|        features.update({'wordend[-2:]': word[-2:]})
18|
19|    ###previous words features
20|    if i > 0:
21|        word1 = sent[i-1][0]
22|        postag1 = sent[i-1][1]
23|        features.update({
24|            '-1:word.lower()': word1.lower(),
25|            '-1:word.istitle()': word1.istitle(),
26|            '-1:word.isdigit()': word1.isdigit(),
27|            '-1:postag': postag1,
28|        })
29|
30|    ###following words features
31|    if i < len(sent)-1:
32|        word1 = sent[i+1][0]
33|        postag1 = sent[i+1][1]
34|        features.update({
35|            '+1:word.lower()': word1.lower(),
36|            '+1:word.istitle()': word1.istitle(),
37|            '+1:word.isdigit()': word1.isdigit(),
38|            '+1:postag': postag1,
39|        })

```

Listing 4.1: Pseudocode of the feature generation for sequence classification with feature-based machine learning.

The computationally heavy experiments with neural networks are conducted with the Flair NLP framework on the cloud-platform Colab. The script utilized in the classification experiments with Flair (see Section 4.1.1) can be modified and applied for this sequence-tagging task. The initialization of the hardware resources and the loading of the data remains the same. After loading the data, a dictionary of the tags needs to be generated (see Figure 4.6).

```

1 # 2. what tag do we want to predict?
2 tag_type = 'comp'
3 # 3. make the tag dictionary from the corpus
4 tag_dictionary = corpus.make_tag_dictionary(tag_type=tag_type)

```

Dictionary with 8 tags: <unk>, O, OBJ-1, OBJ-2, SHARED, ASPECT, <START>, <STOP>

Figure 4.6: The figure shows the code to generate a tag dictionary used by the Flair framework.

In comparison to the classification task, the step of loading the embeddings changes in a way that the separate embeddings are only initialized and then stacked upon each other (see Figure 4.7). The script allows to choose the embeddings by commenting in the desired embeddings. For this step, Flair supports “classical” word and character embeddings, as well as transformer-based architectures like ALBERT through the Huggingface Transformers library.

```

1 # 4. initialize embeddings
2 embedding_types: List[TokenEmbeddings] = [
3     #comment in the lines to use the embeddings
4     WordEmbeddings('glove'),
5     CharacterEmbeddings(),
6     BertEmbeddings(bert_model_or_path="albert-large-v2"),
7     #FlairEmbeddings('news-forward'),
8     #FlairEmbeddings('news-backward'),
9 ]
10 embeddings: StackedEmbeddings = StackedEmbeddings(embeddings=embedding_types)

```

Figure 4.7: The figure shows the code to initialize several embeddings as a stacked embedding with Flair.

The major difference of the script is the creation of a sequence tagger, which is handed over to the model trainer. The sequence tagger takes the previously initialized stacked embeddings and the tag dictionary as an input. All other parts of the script, including the model trainer, do not need to be changed. The model trainer will train the sequence tagger and generate a model that can be saved to Google Drive for a future usage.

```

1 #5. create the sequence tagger
2 tagger: SequenceTagger = SequenceTagger(hidden_size=256,
3                                         embeddings=embeddings,
4                                         tag_dictionary=tag_dictionary,
5                                         tag_type=tag_type,
6                                         use_crf=True)

```

Figure 4.8: The figure shows the code to create a sequence tagger with the stacked embeddings in Flair.

### 4.2.2 Training and results

The experiments on the extraction of comparative objects and aspects are split into four parts. The first part is the general evaluation of the feature-based ML models. The second part of the experiments optimizes the combination of features for a linear SGD model within the hardware resource limits. The third part of the experiments evaluates various parameter and embedding combinations for the neural machine learning. The last part of the experiments uses a dedicated test set to evaluate selected models in Flair.

#	Dataset	Samples	Model	Memory use	F1 score (micro)
1	Multi-info	2,568	Perceptron	25GB	0.59
2	Multi-info	2,568	Multinomial Naïve Bayes	25GB	0.59
3	Multi-info	2,568	Passive Aggressive Classifier	25GB	0.64
4	Multi-info	2,568	SGD with linear SVM	25GB	<b>0.65</b>
5	Multi-info	3,998	CRF	2GB	<b>0.68</b>

Table 4.6: The table shows the micro F1 scores and the configuration of the first series of experiments with feature-based machine-learning models.

The first part of the experiments utilized feature-based machine learning to tag sequences. All models were run with the same data and a train/test split of 85% to 15%. The features generated in the second step of the script (see Section 4.2.1) resulted in a high memory usage. The local environment had a maximum of 25GB RAM available for the training. Training with all features and the whole dataset exceeded this mark. Therefore the size of the dataset had to be reduced to 2,568 samples (2,182 training / 385 test). Table 4.6 shows the F1 results of the models for the different tags and the micro F1 score. The SGD with linear SVM has the highest micro F1 score (0.65). In contrast to these four models, the CRF model could be trained with all features on the complete dataset. For this model a random grid search with cross validation was used to find the best parameters on a development set of 600 samples. After the grid search the model was tested against a set of 600 samples (15%) and achieved a micro F1 score of 0.68. Because the sizes of the training and of the test sets are different for the CRF and the SGD model, their results are not necessarily directly comparable. In order to overcome these differences in the size of the datasets, the features were varied in the second part of the experiments. The aim was to maximize the data size and to generate competitive micro F1 results. Table 4.7 shows the feature variation for the linear SGD, the resulting micro F1 scores and the models memory footprint. The memory footprint was used as an indicator for the reduction of features or the possibility to extend the dataset size.

#	Samples	Feature 1	Feature 2	Memory use	F1 score (micro)
4	2,568	Word (local, -3/+3)	POS/endings/title/digit	25GB	0.65
6	2,568	Word (local, -3/+3)	POS/title/digit	24GB	0.62
7	2,568	Word (local, -3/+3)	POS/(local title/digit)	23.9GB	0.62
8	2,568	Word (local)	POS/title/digit	4GB	0.63
9	2,568	Word (local, +1)	POS/title/digit	7.5GB	0.58
10	2,568	Word (local, -1/+1)	POS/title/digit	11.1GB	0.64
11	2,568	Word (local, -2/+2)	POS/title/digit	17.7GB	0.65
12	3,998	Word (local)	POS/title/digit	7.9GB	0.61
13	3,998	Word (local, +1)	POS/title/digit	15GB	0.61
14	3,998	Word (local, -1/+1)	POS/title/digit	22.7GB	0.62
15	3,136	Word (local, -2/+2)	POS/title/digit	24.5GB	0.63

Table 4.7: The table shows the micro F1 scores and the feature configurations of the second series of experiments with a linear SGD model. Experiment four is the same as in Table 4.6. Feature 1 describes the usage of the token and the neighbouring words. The abbreviation “Word (local -3/+3)” means that the word of the token itself and the three previous and following words are used as a feature. Feature 2 describes the usage of the additional information for the local and the three previous and following words.

Experiments 6-8 show that a reduction of the additional information had only a low impact on the memory footprint. Removing the word endings and reducing the titlecased and digit flags to the local token reduced the F1 score to 0.62, while the memory footprint was reduced only by 1GB. Removing neighbouring words as a feature (Experiment 8) had a big impact on the memory footprint and resulted in the smallest model (4GB). Experiments 9-11 show that the memory footprint was increased up to 17.7GB when adding a maximum of two previous and following words as features. Experiment 11 achieved the same F1 score with less features than Experiment 4. Experiments 12-15 utilized the feature combinations from Experiments 8-11, with the maximum dataset size possible. For the Experiments 12-14 it was possible to use the complete dataset without hitting the hardware memory limit. Experiment 15 enabled the use of 3,136 samples. As a result of the second series of experiments, the Experiments 12-14 can be compared to Experiment 5. The dataset sizes are the same, however they do not overcome the result of the CRF model from Experiment 5.

The third series of experiments evaluates various embeddings utilized with the Flair sequence tagger in the cloud environment Colab. Colab provides hardware resources, such as GPUs, for the computationally-intensive neural machine learning. All experiments were

conducted with the full dataset (3,998 samples). 70% of the data was used for the training and 15% was hold out for both the development and the test set. In Flair, a bidirectional LSTM model with optional conditional random fields was utilized as a sequence tagger.

#	Model	Embedding 1	Embedding 2	Embedding 3	F1 score (micro)
16	Bi-LSTM-CRF	GloVe	-	-	0.6972
17	Bi-LSTM-CRF	GloVe	Flair-fwd/bwd	-	0.7504
18	Bi-LSTM-CRF	GloVe	Char-Embedding	Flair-Fwd/Bwd	0.7697
19	Bi-LSTM	GloVe	Char-Embedding	Flair-Fwd/Bwd	0.7365
20	Bi-LSTM-CRF	GloVe	Char-Embedding	DistilBert-base-cased	0.7855
21	Bi-LSTM-CRF	GloVe	DistilBert-base-cased	-	0.7846
22	Bi-LSTM-CRF	GloVe	Bert-large-cased	-	0.8033
23	Bi-LSTM-CRF	GloVe	Albert-base-v2	-	0.8089
24	Bi-LSTM-CRF	GloVe	Albert-large-v2	-	0.7929
25	Bi-LSTM-CRF	GloVe	Char-Embedding	Albert-large-v2	0.7936

Table 4.8: The table shows the micro F1 scores and the embedding configurations of the experiments with neural networks in Flair.

Table 4.8 shows the results of the experiments with different embeddings combinations in Flair. Experiment 16 utilized GloVe embeddings and it performed 0.017 points better than the CRF model from Experiment 5. Experiments 17-18 added the English Flair-forward and Flair-backward embeddings, as well as character embeddings. Both changes increased the micro F1 score. Deactivating the CRF feature of the sequence tagger resulted in a decrease of the F1-Score (Experiment 19). Experiments 20-25 explored various combinations of BERT embeddings in different sizes and types. The best micro F1 score was achieved by a combination of GloVe and ALBERT (base-v2) embeddings with a F1 score of 0.8089. The Configurations 22-25 generated the leading results on the full dataset. In the fourth series of the experiments, these four configurations are trained on the full dataset (15% development set) and tested against the dedicated gold-labelled sequence data. Table 4.9 shows the results of this training with dedicated F1 scores for each of the tags. Experiment 26 has the overall best results for classifying the sequence tags, with a micro F1 score of 0.8054 against the manually-labelled test set. This experiment reaches the highest scores for the OBJ-1 tag and aspects of the comparison. The F1 score for the OBJ-2 tag and the shared objects is exceeded by Experiment 27.



#	Embeddings	F1 score (micro)	F1 score (OBJ-1)	F1 score (OBJ-2)	F1 score (ASPECT)	F1 score (SHARED)
26	GloVe + Bert-large-cased	0.8054	0.8863	0.8936	0.613	0.3429
27	GloVe + Albert-base-v2	0.802	0.8653	0.895	0.6076	0.4135
28	GloVe + Albert-large-v2	0.7857	0.8463	0.8856	0.596	0.3137
29	GloVe + Char-Embedding + Albert-large-v2	0.7888	0.8566	0.8885	0.5908	0.3396

Table 4.9: The table shows the F1 scores and the embedding configurations of the Bi-LSTM-CRF sequence tagger in Flair. The model was tested against a dedicated gold labelled test set of 385 samples.

### 4.3 Discussion of results

The experiments on classifying comparative questions show that it is possible with the proposed datasets to classify comparatives with a good F1 score. The baseline F1 score of 0.646 is exceeded by the feature-based machine learning, as well as by the neural machine learning. With a macro F1 score of 0.7936, the linear SGD model reaches the highest score of the feature-based machine-learning models. In comparison to the human performance of 0.8983 (macro F1)<sup>12</sup>, this is a notable result, which is only exceeded by more complex BERT transformer networks. This result should only be considered as an indicator of the performance because the test and the training data differ between the experiments. In addition to the results achieved, experiments with feature-based machine-learning methods also have a higher potential. The long run-times of the hyper-parameter and feature grid searches allowed only a partial exploration of the parameter spaces. An extended analysis might increase the performance of the models.

Model	F1 score (macro)	F1 score (comp)	F1 score (non-comp)
Flair LSTM with ALBERT large embeddings	0.8760	0.8316	0.9204
Human performance	0.8983	0.8912	0.9052

Table 4.10: The table shows the performance of the best Flair-based NN with ALBERT embeddings compared to the human performance on classifying comparative questions.

Flair Framework provides an easy-to-use approach to neural machine learning with various possibilities of embeddings to choose from. The Flair classification provides the best results with the ALBERT embeddings on the dedicated test data (macro F1 score 0.8760). The score is only 0.022 points lower than the human performance of 0.8983 (macro F1). Table 4.10 compares the performance of best neural network and the human performance for the classes. Furthermore, the experiments of the third series (see Table 4.5) show that the results benefit from the improvements to the human annotated classification data. Extending the amount of non-comparative data is beneficial up to 3,000 samples. With 6,000 additional non-comparative samples, the F1 scores slightly drop, which may be related to the high imbalance of the classes (4,000 comparatives to 11,000 non-comparatives). The Flair implementation with ALBERT embeddings provides a good foundation for further research. The replacement of the embeddings, for example, a change to ALBERT large embeddings or other derivatives of BERT, has the potential of future performance increases. Furthermore, a parameter optimization through a grid search might increase the performance of the model.

<sup>12</sup>The human performance was evaluated by manually labelling 10% of the human labelled data (see Section 3.4).

The experiments on the identification of comparative objects and aspects show how intensive the training is in terms of resources. The feature-based machine-learning models reach the system memory limit with their complex and extensive features. Series two (see Table 4.7) of the experiments shows that a reduction of the features results in a decrease of the micro F1 score. Only the feature-reduced Experiment 11 can reach a micro F1 score of 0.65, which is also reached when training with all features. Anyhow, the feature set can not be trained with the complete dataset, as this exceeded the memory limit. The other experiments with reduced features are not able to reach the F1 score of the CRF model (Experiment 5). The CRF model is the only model that had its parameters optimized through a random grid search with cross validation. On the one hand, this increases the reliability of the results of the CRF model, but on the other hand, it also means that there might still be potential for the other feature-based models that was not exhausted. However, the grid searches have high training times and are computationally expensive. The integration of new features, for example, the word shape or the utilization of word embeddings, might be beneficial for the feature-based classifiers and their resource needs. Recognizing that a simple Bi-LSTM-CRF model that solely uses GloVe embeddings exceeds the performance of the CRF model, provides an argument against any further exploration of feature-based machine-learning models in this context. Series three and four of the experiments show that using ALBERT and BERT embeddings with a Bi-LSTM-CRF model results in the best F1 scores. A combination of BERT and GloVe embeddings achieves the best overall micro F1 result of 0.8054 on a manually-labelled test set (Experiment 26). This model reaches the human micro F1 score (see Table 4.11). Comparing the results for each tag shows that the F1 scores for the comparative objects and the aspect of the comparison exceed human performance. Solely the classification of the shared objects has a higher magnitude difference with 0.15 points. It is likely that the model lacks support for this tag, as only 24% of samples have a SHARED tag and humans have only a 42% accuracy on SHARED tags. In comparison to the SHARED tag, 100%

Model	F1 score (micro)	F1 score (OBJ1)	F1 score (OBJ2)	F1 score (ASPECT)	F1 score (SHARED)
Flair Bi-LSTM-CRF with GloVe and BERT large cased embeddings	0.8054	0.8863	0.8936	0.613	0.3429
Human performance	0.8049	0.8839	0.8937	0.6022	0.4943

Table 4.11: The table shows the performance of the best Flair based NN with BERT embeddings compared to the human performance on the sequence classification of comparative objects (OBJ), shared objects (SHARED) and comparative aspects (ASPECT) in comparative questions.

of the samples have two comparative objects and 60% of the samples have an ASPECT tag. Although the BERT model has the overall highest F1 score, ALBERT models should be considered depending on the use case. The ALBERT model from Experiment 27 is 5.5 times smaller than the BERT model (2 GB). Furthermore, the experiments were limited to “large” BERT and ALBERT models because bigger models exceeded the GPU memory limit. A research with less limitations in terms of hardware could increase the results. Moreover, a resource-intensive parameter optimization was not performed with the neural ML models. This provides another point of improvement for future research.

## 4.4 Comparative classification web app

This section describes the practical use of the models, presented in the previous sections, in a “classification pipeline”. The pipeline incorporates the two main goals of the thesis. Firstly, to classify if a question is a comparative question. And secondly, to classify each word of a sentence to find comparative objects and comparative aspects. The pipelines features are accessible through a RESTful Application Programming Interface, which is made available through a Flask<sup>13</sup> micro service. Flask is a micro web framework that includes only the most necessary web functions by default. For example, there is no database layer integrated by default. The framework can be extended by third-party libraries and packages. For the classification pipeline, Flask is extended by the Flask-RESTX<sup>14</sup> package, which supports building REST APIs.

The pipeline is divided into four main functions, which are processed one after the other. The first function tokenizes the sentence with the spaCy tokenizer in the same way as described in the experiments in Section 4.1.1. The second function generates a classification for the sentence with a classifier from the Flair framework. The function predicts a binary label (comparative / non-comparative) and supplies a confidence for that prediction. The third part of the pipeline predicts a sequence tag for each token of the sentence with a sequence classifier from Flair framework. For each token the function returns the three highest ranking labels with their corresponding confidence. The sequence tags are predicted even if the comparative classification of the sentence was negative. The last function of the pipeline accumulates the data and organizes and aggregates. The data package is send in the JavaScript Object Notation (JSON)<sup>15</sup> format through the API. In order to be functional, the two classification functions need to be provided with models that were trained in Flair on the comparative datasets. The deployment of the pipeline on a local Windows 10 machine and on a remote Linux server showed that Flair models, trained on each of the systems, are not interchangeable. The models contain hard coded paths to temporary files, which can not be found on computers with another operating system than the one the model was trained on.

The API has two endpoints to interact with the pipeline. The first endpoint handles the classification and extraction of a sentence (path: /api/compq/extract). A client can send a sentence to the endpoint (via HTTP POST<sup>16</sup>) and will receive the analysis of the sentence as an answer. The second endpoint creates an analysis with an example sentence (path:

---

<sup>13</sup><https://flask.palletsprojects.com>

<sup>14</sup><https://flask-restx.readthedocs.io>

<sup>15</sup>JSON: A human-readable open standard file format to transmit attribute-value pairs of data.

<sup>16</sup>Hypertext Transfer Protocol (HTTP) POST method to send data to a server.

Field name	Function	Example
sentence	The analysed sentence	“Can men run faster than women?”
label	Binary label of the sentence: “comparative” or “non-comparative”	“comparative”
score	The confidence for the label	0.9478
tokens	A list of tokens from the sentence with three tags and scores	(men, OBJ-1, 0.99)
OBJ1	The first comparative object as continuous text	“men”
OBJ2	The second comparative object as continuous text	“women”
SHARED	The shared object as continuous text	“ ”
ASPECT	The aspect as continuous text	“run faster”
annotated Sentence	The sentence with inline tags	“Can men (OBJ-1) run (ASPECT) faster (ASPECT) than women (OBJ-2)?”
htmlOut	Formatted HTML output with separation and highlighting of the tokens	“Can <span class="OBJ-1"> men</span> <span class="ASPECT"> run</span> <span> faster</span> <span> than women</span> <span> (OBJ-2)?</span> ”

Table 4.12: The table shows the data fields of a response from the extraction pipeline API. The answer has the same structure for both API endpoints (/extract and /example). The example column shows the response for the query “Can men run faster than women?”.



/api/compq/example). A client can invoke the endpoint (via HTTP GET<sup>17</sup>) without any further specification and will receive the analysis of a randomly picked example sentence as an answer. The response has the same structure for both API endpoints (/extract and /example). Table 4.12 shows the response fields of the endpoints. The most important fields are the label and score(confidence), as well as the list of tokens with their assigned labels and confidences. The remaining fields are convenience fields that provide a faster access to the results.

Providing the API enables systems to easily get access to the analysis of comparative sentences without the need of further programming and without the direct interaction with the trained classifiers. In order to enable access to the classification functionality for human users, a web front end was developed. Together with the API as a server back end, the front end assembles a small web app. The front end uses the JavaScript web framework React<sup>18</sup> to provide the user interface. Figure 4.9 shows the interface of the web app. The user can enter a question in the input field or request an example at the

<sup>17</sup>Hypertext Transfer Protocol (HTTP) GET method to request data from a server.

<sup>18</sup><https://reactjs.org/>

top of the page. The result is presented at the bottom half of the page. The results are structured into four tiles. The “Analysis” tile shows the sentence and the comparative label, as well as the confidence for the label. The tokens, which are comparative objects and aspects are marked with a colour. The legend for the colours is provided in the bottom right tile. The top right tile shows the extractions for the found comparative objects and aspects as a contentious text. In the case that no aspect or shared object is found, the fields are not displayed. The bottom left tile gives a detailed view for each token. The tokens are displayed in separate boxes, which can be extended with a mouse click. After the extension of a box (see Figure 4.9 Token 7), the interface shows up to three tags with their corresponding confidence. The tags are sorted by their confidence. The tag on top (in bold) is the label that was assigned to the token by the system. In the case that the confidence for a tag is lower than 0.01% the tag is not displayed.

 **CompQ Extractor Demo** 

## Comparative Question Extractor

This demo application can classify comparative sentences and extract their comparative objects and aspects. You can type a sentence below or start with an example.

Do you have a comparative question?

**Analyze Sentence** **Show an Example**

### Analysis

Your Sentence is **comparative** with a confidence of 98.55%

What is the difference between **Cola** and **Pepsi**

### Extractions

Object 1: *Cola*

Object 2: *Pepsi*

### Token Labels

Please click on the token to see the labels and their confidence.

Token 0: *What*

Token 1: *is*

Token 2: *the*

Token 3: *difference*

Token 4: *between*

Token 5: *Cola*

Token 6: *and*

Token 7: *Pepsi*

Label: **OBJ-2 (99.52%)**  
Label: SHARED (0.09%)  
Label: OBJ-1 (0.03%)

### Legend

**Object 1**

**Object 2**

**Shared Object**

**Aspect**

Figure 4.9: The figure shows the interface of the comparative classification web app. A user can query a sentence to the system and gets a detailed analysis on the comparative classification, as well as the comparative objects and aspects.



## Chapter 5

# Conclusion

The linguistic-based taxonomy for comparative questions proposed in Chapter 3 has enabled the creation of two open-domain datasets, which have been utilized to successfully classify comparative questions and extract their comparative objects and their comparative aspects.

An analysis of the linguistic background on comparatives and questions has shown that comparative questions can be generated with a set of textbook rules, as well as with special words and phrases. The results of this analysis and the project requirements have laid the foundation of the definition of a comparative question (see Section 3.1.4). Hence, the definition complies with the needs of this thesis and is not universally valid. It requires the question to be written in the English language and to hold exactly two comparative objects. The definition allows the question to contain multiple sentences, spelling mistakes or incomplete punctuation and does not limit the domain of the question or its comparative entities. This way, the definition ensures that systems like the Comparative Argumentative Machine, specialized on providing fact supported answers for comparisons with two entities, can handle all valid comparisons while keeping the domain of the data open. Based on the definition and the linguistic background of comparative questions, a linguistic-based taxonomy for comparative questions has been proposed in Section 3.2. The taxonomy has been developed to be suitable for the data mining of comparative questions. It is organized in a hierarchical tree-like structure that reassembles a more detailed generation of a comparative question with every level of depth.

The taxonomy has enabled the data mining for comparative questions out of 9.6 million open-domain sentences from the platforms Reddit and Yahoo Answers. A total of 245,000 possible comparative samples have been found through the data-mining process. In the first crowd-sourced human annotation task (see Section 3.4), 10,380 samples have been classified under the classes “comparative” and “not comparative”. In a second human

annotation task, 4,260 comparative labelled sentences have been provided with sequence tags for comparative objects and comparative aspects (see Section 3.5). As a result, two novel open-domain datasets have been created in the English language for the classification of comparative questions and the identification of comparative objects and comparative aspects.

Experiments with the datasets have shown that good results, close to the human performance, can be reached by means of neural networks that use ALBERT embeddings. In the classification task, the model reaches a macro F1 score of 0.876, 0.022 points below the human performance. The identification of comparative objects and aspects reaches the human performance with a micro F1 score of 0.8054. The results of the experiments and the human performance reveal how difficult comparative questions are. In 34% of the samples, only two out of three human annotators agreed on a classification for a sentence. In the process of identifying the comparative objects and aspects, the human annotators only had a micro average precision of 81% over all classes. Shared comparative objects were the most difficult class for human annotators with a precision of only 42%. This highlights the importance of good quality data and high-precision human annotations, in order to be able to train reliable classifiers.

The comparative classification web app, proposed in Section 4.4, brings together all the information of the thesis. The classification models are utilized in an extraction pipeline, providing a simple way for users to classify sentences and to extract the comparative objects and aspects through a website. For systems like CAM, the web app allows the access to an API to query comparative classification analyses. With the created datasets and the performed experiments, the thesis comes one step closer to comprehend people's natural-language questions in form of comparative questions.

## Chapter 6

# Future work

There are several aspects that could be proposed for future research. Some can improve or change the datasets, others might optimize the classification results of the machine-learning models.

Human annotation has shown that it is hard for humans to agree on one label for some sentences. To reach a higher classification precision, difficult sentences could be classified again under a new human annotation task. The sentences for the task could be selected by choosing those that do not have an absolute agreement vote. Another option would be to select sentences in which the machine classifier has a low confidence. Widening the definition of a comparative question could also be beneficial for the annotation quality. Allowing more than two comparative objects as well as comparisons against open-group entities would make the classification more natural for the human annotators. But this measure would come at the price that question answering systems would need to be able to handle comparisons with more than two objects and especially comparisons against groups like one entity versus all others.

With the existing classifiers it is possible to extend the dataset. A higher amount of labelled data could be collected by classifying the samples, which were not annotated yet, from the filtered dataset. The set contains 235,000 samples that could be used for this data increment. Furthermore, Reddit has been proven to be a good resource for open-domain comparative questions. Question-centred subreddits (see Section 3.3.4) could be exploited for more comparative samples. The content of information in the datasets can be extended by annotating the attributes of the comparative aspects. The datasets contain 2,400 samples, which hold a comparative aspect. A human annotation with skilled workers would be needed to annotate these attributes. Furthermore, gathering the best answers to the samples in the dataset would extend the quantity of information in the datasets and allow the creation of new tasks.

The experiments could be performed by more advanced hardware in order to optimize their classification results. This would especially help to enable the use of larger embeddings for both experiments. In the case of the feature-based ML for the sequence tagging, an operating system with more than 25GB free memory would be able to execute the experiments with more text features. Moreover, implementing a parameter optimization (for example a grid search) for the Flair neural networks could improve the classifiers' results. Apart from the optimizations, the comparative classification model could be tested on out-of-scope data. Bondarenko et al. [51] proposed a comparative dataset in the English language without reference classification scores.

These are just a few examples of what future research could incorporate to improve the dataset and the classification tasks. The publication of the datasets under a Creative Commons license would provide the research community with new research opportunities. Argumentative question answering systems (e.g., CAM) could be extended and enhanced and the research on classifying comparative questions could advance further.

# Bibliography

## Article references

- [13] M. Schildwächter, A. Bondarenko, J. Zenker, M. Hagen, C. Biemann, and A. Panchenko, “Answering comparative questions: Better than ten-blue-links?” In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, L. Azzopardi, M. Halvey, I. Ruthven, H. Joho, V. Murdock, and P. Qvarfordt, Eds., ACM, 2019, pp. 361–365, ISBN: 9781450360258.
- [14] B. F. Green, A. K. Wolf, C. Chomsky, and K. Laughery, “Baseball,” in *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference on - IRE-AIEE-ACM '61 (Western)*, W. F. Bauer, Ed., ACM Press, 1961, p. 219.
- [15] W. A. Woods, “Progress in natural language understanding,” in *Proceedings of the June 4-8, 1973, national computer conference and exposition on - AFIPS '73*, ACM Press, 1973, p. 441.
- [16] B. Katz, “Using english for indexing and retrieving,” in *Artificial Intelligence at MIT Expanding Frontiers*, MIT Press, 1991, pp. 134–165, ISBN: 0262231506.
- [17] B. Katz, “Annotating the world wide web using natural language,” in *Computer-Assisted Information Searching on Internet*, ser. RIAO '97, Centre de hautes études internationales d’informatique documentaire, 1997, pp. 136–155.
- [18] B. Katz, S. Felshin, D. Yuret, A. Ibrahim, J. Lin, G. Marton, A. Jerome McFarland, and B. Temelkuran, “Omnibase: Uniform access to heterogeneous data for question answering,” in *Natural Language Processing and Information Systems*, B. Andersson, M. Bergholtz, and P. Johannesson, Eds., Springer Berlin Heidelberg, 2002, pp. 230–234, ISBN: 978-3-540-36271-5.
- [21] D. A. Ferrucci, “Introduction to “this is watson”,” *IBM Journal of Research and Development*, vol. 56, no. 3.4, 1:1–1:15, 2012.

## BIBLIOGRAPHY

---

- [24] A. Lally, J. M. Prager, M. C. McCord, B. K. Boguraev, S. Patwardhan, J. Fan, P. Fodor, and J. Chu-Carroll, “Question analysis: How watson reads a clue,” *IBM Journal of Research and Development*, vol. 56, no. 3.4, 2:1–2:14, 2012.
- [25] J. Chu-Carroll, J. Fan, B. K. Boguraev, D. Carmel, D. Sheinwald, and C. Welty, “Finding needles in the haystack: Search and candidate generation,” *IBM Journal of Research and Development*, vol. 56, no. 3.4, 6:1–6:12, 2012.
- [26] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100,000+ questions for machine comprehension of text,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2016, pp. 2383–2392. [Online]. Available: <https://www.aclweb.org/anthology/D16-1264>.
- [27] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Association for Computational Linguistics, 2018, pp. 784–789. [Online]. Available: <https://www.aclweb.org/anthology/P18-2124>.
- [28] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, “Race: Large-scale reading comprehension dataset from examinations,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2017, pp. 785–794. [Online]. Available: <https://www.aclweb.org/anthology/D17-1082>.
- [29] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang, *Ms marco: A human generated machine reading comprehension dataset*, 2016. [Online]. Available: <http://arxiv.org/pdf/1611.09268v3>.
- [32] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, *Albert: A lite bert for self-supervised learning of language representations*, 2019. [Online]. Available: <http://arxiv.org/pdf/1909.11942v4>.
- [34] P. Qi, X. Lin, L. Mehr, Z. Wang, and C. D. Manning, *Answering complex open-domain questions through iterative query generation*, 2019. [Online]. Available: <http://arxiv.org/pdf/1910.07000v1>.
- [35] A. Broder, “A taxonomy of web search,” *ACM SIGIR Forum*, vol. 36, no. 2, p. 3, 2002.
- [36] D. E. Rose and D. Levinson, “Understanding user goals in web search,” in *Proceedings of the 13th International Conference on World Wide Web*, ser. WWW ’04,

- 
- Association for Computing Machinery, 2004, pp. 13–19, ISBN: 158113844X. [Online]. Available: <https://doi.org/10.1145/988672.988675>.
- [37] B. J. Jansen, D. L. Booth, and A. Spink, “Determining the informational, navigational, and transactional intent of web queries,” *Information Processing & Management*, vol. 44, no. 3, pp. 1251–1266, 2008.
- [40] W. G. Lehnert, “The process of question answering,” PhD thesis, Yale University, USA, 1977.
- [41] A. C. Graesser, K. Lang, and D. Horgan, “A taxonomy for question generation,” *Questioning Exchange*, vol. 2, pp. 3–15, 1988.
- [42] A. Graesser, V. Rus, and Z. Cai, *Question classification schemes*, 2008.
- [43] T. W. Lauer and E. Peacock, “An analysis of comparison questions in the context of auditing,” *Discourse Processes*, vol. 13, no. 3, pp. 349–361, 1990.
- [44] N. Jindal and B. Liu, “Identifying comparative sentences in text documents,” in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’06, Association for Computing Machinery, 2006, pp. 244–251, ISBN: 1595933697. [Online]. Available: <https://doi.org/10.1145/1148170.1148215>.
- [45] N. Jindal and B. Liu, “Mining comparative sentences and relations,” in *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, ser. AAAI’06, AAAI Press, 2006, pp. 1331–1336, ISBN: 9781577352815.
- [46] A. Jain and P. Pantel, “Identifying comparable entities on the web,” in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, ser. CIKM ’09, Association for Computing Machinery, 2009, pp. 1661–1664, ISBN: 9781605585123. [Online]. Available: <https://doi.org/10.1145/1645953.1646198>.
- [47] A. Jain and P. Pantel, “How do they compare? automatic identification of comparable entities on the web,” in *2011 IEEE International Conference on Information Reuse Integration*, 2011, pp. 228–233.
- [48] S. Li, C. Lin, Y. Song, and Z. Li, “Comparable entity mining from comparative questions,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1498–1509, 2013.
- [49] A. Panchenko, A. Bondarenko, M. Franzek, M. Hagen, and C. Biemann, “Categorizing comparative sentences,” in *Proceedings of the 6th Workshop on Argument Mining*, Association for Computational Linguistics, 2019, pp. 136–145. [Online]. Available: <https://www.aclweb.org/anthology/W19-4516>.

## BIBLIOGRAPHY

---

- [50] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, “Supervised learning of universal sentence representations from natural language inference data,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2017, pp. 670–680. [Online]. Available: <https://www.aclweb.org/anthology/D17-1070>.
- [51] A. Bondarenko, P. Braslavski, M. Völske, R. Aly, M. Fröbe, A. Panchenko, C. Biemann, B. Stein, and M. Hagen, “Comparative web search questions,” in *Proceedings of the 13th International Conference on Web Search and Data Mining*, ser. WSDM ’20, Association for Computing Machinery, 2020, pp. 52–60, ISBN: 9781450368223. [Online]. Available: <https://doi.org/10.1145/3336191.3371848>.
- [52] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “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, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: <https://www.aclweb.org/anthology/N19-1423>.
- [67] G. Pass, A. Chowdhury, and C. Torgeson, “A picture of search,” in *Proceedings of the 1st International Conference on Scalable Information Systems*, ser. InfoScale ’06, Association for Computing Machinery, 2006, 1–es, ISBN: 1595934286. [Online]. Available: <https://doi.org/10.1145/1146847.1146848>.
- [69] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov, “Natural questions: A benchmark for question answering research,” *Transactions of the Association for Computational Linguistics*, vol. 7, no. 15, pp. 453–466, 2019.
- [80] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, “Flair: An easy-to-use framework for state-of-the-art nlp,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, Association for Computational Linguistics, 2019, pp. 54–59. [Online]. Available: <https://www.aclweb.org/anthology/N19-4010>.

## Book references

- [39] T. W. Lauer, E. Peacock, and A. C. Graesser, Eds., *Questions and information systems*. Hillsdale, N.J: L. Erlbaum, 1992, ISBN: 978-0805810189.
- [53] B. Aarts, *Oxford modern English grammar*. Oxford: Oxford University, 2011, ISBN: 978-0-19-953319-0.



---

## Online references

- [1] T. Barnett. (2016). The zettabyte era officially begins. Cisco Systems, Ed., [Online]. Available: <https://blogs.cisco.com/sp/the-zettabyte-era-officially-begins-how-much-is-that> (visited on 02/17/2020).
- [2] Cisco Systems. (2020). Vni forecast highlights: 2022 forecast highlights. Cisco Systems, Ed., [Online]. Available: [https://www.cisco.com/c/m/en\\_us/solutions/service-provider/vni-forecast-highlights.html](https://www.cisco.com/c/m/en_us/solutions/service-provider/vni-forecast-highlights.html) (visited on 02/17/2020).
- [3] InternetLiveStats.com. (2020). Total number of websites. InternetLiveStats.com, Ed., [Online]. Available: <https://www.internetlivestats.com/total-number-of-websites/> (visited on 02/17/2020).
- [4] Danny Sullivan. (2016). Google now handles at least 2 trillion searches per year. searchengineland.com, Ed., [Online]. Available: <https://searchengineland.com/google-now-handles-2-999-trillion-searches-per-year-250247> (visited on 01/14/2020).
- [5] Rand Fishkin. (2017). The state of searcher behavior revealed through 23 remarkable statistics. moz.com, Ed., [Online]. Available: <https://moz.com/blog/state-of-searcher-behavior-revealed> (visited on 01/14/2020).
- [6] Amit Singhal. (2012). Introducing the knowledge graph: Things, not strings. Google Blog, Ed., [Online]. Available: <https://search.googleblog.com/2012/05/introducing-knowledge-graph-things-not.html> (visited on 01/14/2020).
- [7] James Vincent. (2016). Google boasts about how good its ai is. TheVerge.com, Ed., [Online]. Available: <https://www.theverge.com/2016/10/4/13122406/google-phone-event-stats> (visited on 01/14/2020).
- [8] Moz Inc. (2020). Google serp feature graph. moz.com, Ed., [Online]. Available: <https://moz.com/mozcast/features> (visited on 01/14/2020).
- [9] Synchrony Financial. (2016). Fifth annual major purchase consumer study. Synchrony Financial., Ed., [Online]. Available: [https://www.synchrony.com/download/2016\\_Major\\_Purchase\\_Study\\_White\\_Paper.pdf](https://www.synchrony.com/download/2016_Major_Purchase_Study_White_Paper.pdf) (visited on 02/17/2020).
- [10] Alexa.com. (2020). Competitive analysis, marketing mix and traffic for cnet.com. Alexa.com, Ed., [Online]. Available: <https://www.alexa.com/siteinfo/cnet.com> (visited on 01/14/2020).
- [11] Quora.com. (2019). How does quora work? Quora.com, Ed., [Online]. Available: <https://www.quora.com/How-does-Quora-work> (visited on 01/14/2020).

## BIBLIOGRAPHY

---

- [12] Language Technology Group. (2018). *Acqua: Argumentation in comparative question answering*, [Online]. Available: <https://www.inf.uni-hamburg.de/en/inst/ab/lt/research/acqua.html>.
- [19] Wolfram Alpha. (2020). *Wolfram alpha: About*. Wolframalpha.com, Ed., [Online]. Available: <https://www.wolframalpha.com/about/> (visited on 01/20/2020).
- [20] Wolfram Alpha. (2020). *Wolfram alpha: Frequently asked questions*. Wolframalpha.com, Ed., [Online]. Available: <https://www.wolframalpha.com/faqs/> (visited on 01/20/2020).
- [22] J. Best. (2013). *Ibm watson: The inside story of how the jeopardy-winning supercomputer was born, and what it wants to do next*. Tech Republic, Ed., [Online]. Available: <https://www.techrepublic.com/article/ibm-watson-the-inside-story-of-how-the-jeopardy-winning-supercomputer-was-born-and-what-it-wants-to-do-next/> (visited on 01/22/2020).
- [23] S. Shippy. (2011). *Questions asked to watson on jeopardy*. Quora.com, Ed., [Online]. Available: <https://www.quora.com/What-questions-were-asked-in-the-Jeopardy-episode-involving-Watson> (visited on 01/23/2020).
- [30] S. Kumar. (2018). *Ai outperforms humans in question answering*. medium.com, Ed., [Online]. Available: <https://medium.com/the-new-nlp/ai-outperforms-humans-in-question-answering-70554f51136b> (visited on 01/24/2020).
- [31] S. Pham. (2018). *Computers are getting better than humans at reading*, [Online]. Available: <https://money.cnn.com/2018/01/15/technology/reading-robot-alibaba-microsoft-stanford/index.html> (visited on 01/25/2020).
- [33] P. Qi. (2019). *Answering complex open-domain questions at scale*, [Online]. Available: <http://qipeng.me/blog/answering-complex-open-domain-questions-at-scale.html> (visited on 01/25/2020).
- [38] T. Soulo. (2020). *Top google searches (as of 2020)*. ahrefs.org, Ed., [Online]. Available: <https://ahrefs.com/blog/top-google-searches/> (visited on 03/23/2020).
- [54] P. Barger. (2019). *Adjectives and adverbs: Comparative and superlative forms – complete lists*. patternbasedwriting.com, Ed., [Online]. Available: [https://patternbasedwriting.com/elementary\\_writing\\_success/adjectives-adverbs-comparative-superlative-complete-lists](https://patternbasedwriting.com/elementary_writing_success/adjectives-adverbs-comparative-superlative-complete-lists) (visited on 12/11/2019).
- [55] British Council. (2020). *Comparative and superlative adjectives*. British Council, Ed., [Online]. Available: <https://learnenglish.britishcouncil.org/english-grammar-reference/comparative-and-superlative-adjectives> (visited on 03/23/2020).

- 
- [56] Cambridge Dictionary. (2020). Comparison adjectives. Cambridge University Press, Ed., [Online]. Available: <https://dictionary.cambridge.org/de/grammatik/britisch-grammatik/comparison-adjectives-bigger-biggest-more-interesting> (visited on 05/14/2020).
- [57] British Council. (2020). Comparative and superlative adverbs. British Council, Ed., [Online]. Available: <https://learnenglish.britishcouncil.org/english-grammar-reference/comparative-and-superlative-adverbs> (visited on 03/23/2020).
- [58] Cambridge Dictionary. (2020). Comparison adverbs. Cambridge University Press, Ed., [Online]. Available: <https://dictionary.cambridge.org/de/grammatik/britisch-grammatik/comparison-adverbs-worse-more-easily> (visited on 05/14/2020).
- [59] Cambridge Dictionary. (2020). Prepositions. Cambridge University Press, Ed., [Online]. Available: <https://dictionary.cambridge.org/de/grammatik/britisch-grammatik/prepositions> (visited on 05/15/2020).
- [60] Macmillan Dictionary. (2020). Ways of comparing things: Thesaurus. Macmillan Education, Ed., [Online]. Available: <https://www.macmillandictionary.com/thesaurus-category/british/ways-of-comparing-things> (visited on 12/11/2019).
- [61] 7ESL.com. (2019). Transition words and phrases: Useful list, types and examples. 7ESL.com, Ed., [Online]. Available: <https://7esl.com/transition-words/> (visited on 05/15/2020).
- [62] Oxford University Press. (2019). Definition: Comparison. Lexico.com and Oxford University Press, Eds., [Online]. Available: <https://www.lexico.com/definition/comparison> (visited on 05/15/2020).
- [63] Oxford University Press. (2019). Definition: Question. Lexico.com and Oxford University Press, Eds., [Online]. Available: <https://www.lexico.com/definition/question> (visited on 05/15/2020).
- [64] Cambridge Dictionary. (2020). Clause types. Cambridge University Press, Ed., [Online]. Available: <https://dictionary.cambridge.org/de/grammatik/britisch-grammatik/clause-types> (visited on 05/16/2020).
- [65] Cambridge Dictionary. (2020). Question words. Cambridge University Press, Ed., [Online]. Available: <https://dictionary.cambridge.org/de/grammatik/britisch-grammatik/question-words> (visited on 05/16/2020).
- [66] Cambridge Dictionary. (2020). Verb types. Cambridge University Press, Ed., [Online]. Available: <https://dictionary.cambridge.org/de/grammatik/britisch-grammatik/verbs-types> (visited on 05/16/2020).

## BIBLIOGRAPHY

---

- [68] M. Barbaro and T. Zeller. (2006). A face is exposed for aol searcher no. 4417749. nytimes.com, Ed., [Online]. Available: <https://www.nytimes.com/2006/08/09/technology/09aol.html> (visited on 05/23/2020).
- [70] Quora. (2017). Quora duplicate question pairs dataset, [Online]. Available: <https://www.kaggle.com/c/quora-question-pairs> (visited on 08/20/2019).
- [71] Quora. (2018). Quora insincere question dataset, [Online]. Available: <https://www.kaggle.com/c/quora-insincere-questions-classification> (visited on 08/20/2019).
- [72] Yahoo. (2007). Yahoo! search logs: L18, [Online]. Available: <https://webscope.sandbox.yahoo.com/catalog.php?datatype=1> (visited on 08/20/2019).
- [73] Yahoo. (2007). Yahoo! answers dataset: L6, [Online]. Available: <https://webscope.sandbox.yahoo.com/catalog.php?datatype=1> (visited on 08/20/2019).
- [74] Yelp. (2019). Yelp open dataset, [Online]. Available: <https://www.yelp.com/dataset> (visited on 08/20/2019).
- [75] Reddit. (2020). Reddit r/askreddit, [Online]. Available: <https://www.reddit.com/r/AskReddit/> (visited on 06/27/2020).
- [76] Subredditstats. (2020). Subredditstats for r/askreddit, [Online]. Available: <https://subredditstats.com/r/AskReddit> (visited on 05/27/2020).
- [77] Reddit. (2020). Reddit api wiki, [Online]. Available: <https://github.com/reddit-archive/reddit/wiki/API> (visited on 08/07/2020).
- [78] J. M. Baumgartner. (2020). Pushshift reddit api documentation, [Online]. Available: <https://github.com/pushshift/api> (visited on 08/07/2020).
- [79] J. M. Baumgartner. (2020). Pushshift reddit faq, [Online]. Available: [https://www.reddit.com/r/pushshift/comments/bcxguf/new\\_to\\_pushshift\\_read\\_this\\_faq/](https://www.reddit.com/r/pushshift/comments/bcxguf/new_to_pushshift_read_this_faq/) (visited on 08/07/2020).
- [81] C. McCormick and N. Ryan. (2019). Bert fine-tuning tutorial with pytorch, [Online]. Available: <http://www.mccormickml.com> (visited on 08/07/2020).

# List of Figures

1.1	Number of existing websites . . . . .	1
1.2	Google’s knowledge panels . . . . .	2
1.3	Comparative Argumentative Machine (CAM) . . . . .	5
2.1	Wolfram Alpha . . . . .	9
3.1	Examples of rule-based comparisons . . . . .	27
3.2	First two levels of the Comparative Question Taxonomy . . . . .	36
3.3	Third level of the Comparative Question Taxonomy . . . . .	37
3.4	Decision pipeline for comparative question filtering . . . . .	59
3.5	MTurk: How it works . . . . .	68
3.6	MTurk sentence classification template . . . . .	70
3.7	MTurk sequence-tagging template . . . . .	85
3.8	MTurk sequence tagging gold labels . . . . .	90
4.1	Colab Resource Allocation . . . . .	102
4.2	Flair load corpus . . . . .	102
4.3	Flair initialize embeddings . . . . .	103
4.4	Flair initialize classifier . . . . .	103
4.5	Flair initialize trainer . . . . .	103
4.6	Flair sequence tagger tag dictionary . . . . .	111
4.7	Flair sequence tagger initialize embeddings . . . . .	111
4.8	Flair create sequence tagger . . . . .	111
4.9	Comparative classification web app . . . . .	122

LIST OF FIGURES

---

1	MTurk classification template instruction . . . . .	XXVIII
2	MTurk classification template instruction . . . . .	XXIX

# List of Tables

1.1	Examples of comparative questions . . . . .	3
2.1	Examples of the participation of Watson in Jeopardy! . . . . .	10
2.2	Psychological classes of questions by Lauer et al. . . . .	14
2.3	Taxonomy of questions by Graesser et al. . . . .	15
2.4	Taxonomy of comparative questions by Lauer et al. . . . .	16
3.1	Inflection of adjectives . . . . .	24
3.2	Inflection of adjectives (superlative) . . . . .	25
3.3	Irregular adjectives . . . . .	25
3.4	Suffix Adverbs . . . . .	27
3.5	Comparative phrases and words . . . . .	29
3.6	Open questions with wh-words . . . . .	30
3.7	Closed questions with auxiliary verbs . . . . .	31
3.8	Example sentences for the definition of comparative questions . . . . .	33
3.9	Examples from the AOL dataset . . . . .	41
3.10	Examples from the GNQ dataset . . . . .	42
3.11	Examples from the MS MARCO dataset . . . . .	43
3.12	Examples from the Quora duplicate question dataset . . . . .	45
3.13	Examples from the Quora Insincere Question dataset . . . . .	46
3.14	Examples from the Yahoo! Answers dataset . . . . .	49
3.15	Examples from the Reddit r/AskReddit . . . . .	51
3.16	Statistics on the data sources . . . . .	52
3.17	Fetches Reddit fields . . . . .	55

## LIST OF TABLES

---

3.18	Exclusion rules after manual evaluation . . . . .	60
3.19	Exclusion of phrases after manual evaluation . . . . .	61
3.20	Class distribution after manual classification . . . . .	62
3.21	Exclusion rules after manual classification . . . . .	62
3.22	Class distribution after manual reclassification . . . . .	63
3.23	Statistics: Filtered datasets . . . . .	64
3.24	Precisions after filtering the source data . . . . .	65
3.25	Statistics: Full data source filtering . . . . .	66
3.26	MTurk pilot classification samples . . . . .	74
3.27	MTurk additional pilot classification samples . . . . .	75
3.28	MTurk classification pilot distribution of workers . . . . .	75
3.29	MTurk classification pilot sample correctness . . . . .	76
3.30	Classification vote categories . . . . .	78
3.31	Comparative questions classification datasets . . . . .	82
3.32	MTurk sequence pilot distribution of workers . . . . .	92
3.33	Sequence tagging: dataset statistics . . . . .	95
3.34	Sequence tagging: worker performance . . . . .	95
4.1	Feature-based ML: classifiers . . . . .	100
4.2	ML pre-processing combinations . . . . .	105
4.3	Feature-based ML: classification scores . . . . .	106
4.4	Neural ML: classification scores . . . . .	107
4.5	Classification scores with different datasets . . . . .	108
4.6	Feature-based ML: sequence-tagging scores . . . . .	112
4.7	SGD: sequence-tagging scores . . . . .	113
4.8	Neural ML: sequence-tagging scores . . . . .	114
4.9	Neural ML: sequence-tagging scores 2 . . . . .	115
4.10	Comparison of NN classification to human performance . . . . .	116
4.11	Comparison of NN sequence tagging to human performance . . . . .	117
4.12	Extraction Pipeline API response . . . . .	120



---

1	Comparative phrases and words . . . . .	XXIII
2	Comparative phrases and words . . . . .	XXIV
3	MTruk Calculation . . . . .	XXV
4	Classification Dataset taxonomy distribution . . . . .	XXVI
5	ML model source list . . . . .	XXVII



# Listings

3.1	Pseudocode of the Reddit scraper script with an optional collection of the highest ranked answers. . . . .	54
3.2	Pseudocode of the Reddit data reader script. . . . .	56
3.3	Pseudocode of the Yahoo data reader script. . . . .	57
3.4	Pseudocode of the MTurk data preparation script for the human classification project. . . . .	72
3.5	Pseudocode of the text pre-processing functions for the human sequence-tagging project. The code loads the spaCy NLP library and redefines its tokenizer rules. . . . .	87
4.1	Pseudocode of the feature generation for sequence classification with feature-based machine learning. . . . .	110
1	Pseudocode of the random grid search parameter space for the SVM model. XXX	
2	Pseudocode of the random grid search parameter space for the Random Forrest model. . . . .	XXX
3	Pseudocode of the random grid search parameter space for the MLP model. XXXI	
4	Pseudocode of the random grid search parameter space for the Etree model. XXXI	
5	Pseudocode of the grid search parameter space for the SGD model. . . . .	XXXII
6	Pseudocode of the grid search parameter space for the Logistic Regression model. . . . .	XXXII
7	Pseudocode of the training parameters for the the feature-base ML models (see Section 4.1.2). . . . .	XXXII



# List of abbreviations

**ACQuA** Argumentation in Comparative Question Answering

**AI** artificial intelligence

**ALBERT** A Lite BERT

**API** Application Programming Interface

**AWS** Amazon Web Services

**BERT** Bidirectional Encoder Representations from Transformers

**CAM** Comparative Argumentative Machine

**CNN** convolutional neural network

**CRF** conditional random fields

**CSR** Class Sequential Rule

**DTAG** detailed-part-of-speech tag

**eli5** Reddit r/explainlikeimfive

**Etree** Extra Trees

**FOR** false omission rate

**GloVe** Global Vectors for Word Representation

**GNQ** Google Natural Question

**GPU** Graphics Processing Units

**GRU** gated recurrent unit

**HIT** Human Intelligence Task

**HTTP** Hypertext Transfer Protocol

**IEP** indicative extraction pattern

**IMDB** Internet Movie Database

## List of abbreviations

---

**JSON** JavaScript Object Notation

**KB** knowledge base

**LSR** Label Sequential Rule

**LSTM** long short-term memory

**MLP** multi-layer perceptron

**MS MARCO** MAchine Reading COmprehension

**MS** Microsoft

**MTurk** Mechanical Turk

**NB** Naïve Bayes

**NER** named entity recognition

**NLP** natural language processing

**NLU** natural language understanding

**NSQ** Reddit r/NoStupidQuestions

**POS** part-of-speech

**POS tagger** part-of-speech tagger

**PRAW** Python Reddit API Wrapper

**PSAW** Pushshift Python API Wrapper

**QA** question answering

**QAS** question answering systems

**RACE** ReAding Comprehension Dataset

**REST** Representational State Transfer

**RNN** recurrent neural network

**SaaS** Software as a Service

**SGD** stochastic gradient descent

**SQuAD** Stanford Question Answering Dataset

**START** SynTactic Analysis using Reversible Transformations

**SVM** support-vector machine

**tf-idf** term frequency–inverse document frequency

**XML** Extensible Markup Language

# Appendices





Structure	Type	Useful for a questions?	Example
after	Preposition	?	
against	Preposition	Yes, as a abbreviation	Muhammad Ali against Joe Frazier
ahead of	Preposition	Yes	Apple's technology is ahead of Google's.
alike	Adjective	Yes	The cities Istanbul and Athens are alike.
all-time	Adjective	No, part of adjectives	
Alongside	Preposition	No, not comparative?	
as much	Adjective	No, part of adjectives (as .. as)	
beside	Preposition	Yes	Gold looks better beside Silver.
besides	Preposition	Yes	Besides Gold, Sliver is a expensive metal.
between	Preposition	Yes	One can tell the difference between Gold and Silver.
by comparison (with)	Phrase	Yes	Athens is small by comparison with Rome.
close to	Adjective	Yes	Apple's technology is close to Google's.
Compared to	Phrase	Yes	Athens is small compared to Rome.
Compared with	Phrase	Yes	Gold is expensive compared with Silver.
different from	Phrase	Yes	Istanbul is differnt from Athens.
different to	Phrase	Yes	Gold is different to Silver.
dissimilar to	Phrase	Yes	Istanbul is dissimilar to Athens.
equal to	Adjective	Yes	Gold is equal to Platin.
even	Adjective	No, part of adjectives	
in (marked/sharp/stark/striking) contrast to	Phrase	Yes	In contrast to Gold, Silver is cheap.
in comparison	Phrase	?	In comparison, Silver is cheaper than gold.
in comparison to	Phrase	Yes	In comparison to Gold, Silver is cheap.
in comparison with	Phrase	Yes	Gold is expensive in comparison with Silver.

Table 1: Full list of lexical items (phrases and words) used in comparisons (A-I).

Structure	Type	Useful for a questions?	Example
in contradistinction to something	Phrase	?	
in contrast to someone	Phrase	Yes	In contrast to Trump, Obama traveled less.
in relation to	Phrase	Yes	Prices for Gold are high in relation to Silver.
it's one thing to..., it's another/a different thing to	Phrase	No	
just as	Adjective	No, part of adjectives (as .. as)	
just as... so (too)	Phrase	No	
just like	Preposition	Yes	The cities Istanbul and Athens are just like each other.
like	Preposition	Yes	Istanbul is like Athens.
near to	Preposition	Yes	Apple's technology is near to Google's.
next to	Phrase	Yes	Next to Google, Apple is shitty. (Could be biased in questions)
not like	Preposition	Yes	Istanbul is not like Athens.
not only ... but also	Phrase	No, not comparative?	
not the same as	Phrase	Yes	Gold is not the same as Silver.
or	Conjunction	Yes	Who won, Muhammad Ali or Joe Frazier?
over against	Phrase	Yes	Google should be preferred over against Apple.
people like someone/like that	Phrase	No, not comparative?	
related	Adjective	Yes	Gold is related to Silver.
relative to	Phrase	Yes	?
relative	Adjective	?	
Relatively speaking	Phrase	No, not comparative?	
seen against (something)	Phrase	Yes	Seen against Gold, Silver is cheap.
similar to	Preposition	Yes	Istanbul is similar to Athens.
the same	Pronoun	Yes	Gold and Aurum are the same.
the same as	Phrase	Yes	Google is the same as Apple.
the... the...	Phrase	No, part of adjectives	
unequal to	Adjective	Yes	Gold is unequal to Silver.
unlike	Adjective	Yes	Istanbul is unlike Athens.
unrelated	Adjective	Yes	Gold is unrelated to Silver.
versus	Preposition	Yes	Muhammad Ali fought versus Joe Frazier in 1971.
vs.	Preposition	Yes, as a abbreviation	Muhammad Ali vs Joe Frazier
Whereas	Conjunction	No? Implying Answer?	Amazon is a company, whereas Athens is a city.
while	Conjunction	No? Implying Answer?	Amazon is a company, while Athens is a city.

Table 2: Full list of lexical items (phrases and words) used in comparisons (I-Z).

MTurk cost calculation	
Estimated Classification correctness rate	50%
Queries to classify	11,000
Positive Comparatives	5,500
Classification:	
Reward per HIT:	\$ 0.220
Amazon fee	20%
Samples per HIT	20
Assignees per HIT:	3
Cost Classification	\$ 435.6
Sequence Tagging:	
Reward per HIT:	\$ 0.250
Amazon fee	20%
Samples per HIT	10
Assignees per HIT:	3
Cost Sequence Tagging	\$495.0
Costs for Pilots	
	\$60
Sum	\$990.6
Budget	\$ 1,000

Table 3: MTurk cost calculation to determine how many samples can be annotated with the budgeted.

---

Class	Counts	Percentage (%)
110	154	1.52
111	114	1.12
112	250	2.47
113	6	0.059
120	251	2.48
121	174	1.72
122	27	0.26
123	2	0.019
130	363	3.59
131	178	1.76
132	25	0.24
133	2	0.019
140	856	8.47
141	851	8.42
210	416	4.11
211	287	2.83
212	254	2.51
213	19	0.18
220	540	5.34
221	401	3.96
222	63	0.62
223	2	0.019
230	853	8.44
231	495	4.89
232	29	0.28
233	7	0.069
240	2484	24.57
241	1003	9.92

---

Table 4: Distribution of the classification dataset samples in the comparative question taxonomy. All samples (positive and negative) are part of this distribution.

---

Model Implementation	Source
Support Vector Classifier (scikit-learn)	<i>SKLEARN</i> /sklearn.svm.SVC.html
Logistic Regression Classifier (scikit-learn)	<i>SKLEARN</i> /sklearn.linear_model.LogisticRegression.html
Random Forest Classifier (scikit-learn)	<i>SKLEARN</i> /sklearn.ensemble.RandomForestClassifier.html
Extra Trees Classifier (scikit-learn)	<i>SKLEARN</i> /sklearn.ensemble.ExtraTreesClassifier.html
SGD Classifier (scikit-learn)	<i>SKLEARN</i> /sklearn.linear_model.SGDClassifier.html
Bernoulli NB (scikit-learn)	<i>SKLEARN</i> /sklearn.naive_bayes.BernoulliNB.html
XGBoost (XGBClassifier)	<a href="https://xgboost.readthedocs.io">https://xgboost.readthedocs.io</a>
MLPClassifier (scikit-learn)	<i>SKLEARN</i> /sklearn.linear_model.Perceptron.html
Multinomial Naïve Bayes (scikit-learn)	<i>SKLEARN</i> /sklearn.naive_bayes.MultinomialNB.html
Passive-aggressive Classifier (scikit-learn)	<i>SKLEARN</i> /sklearn.linear_model.PassiveAggressiveClassifier.html
Conditional Random Fields (CRF) model	<a href="https://sklearn-crfsuite.readthedocs.io">https://sklearn-crfsuite.readthedocs.io</a>

Table 5: The table shows the models used in this thesis with their implementation source. Note: The “SKLEARN” variable stands for the standard scikit-learn website link: “<https://scikit-learn.org/stable/modules/generated/>”

---

## Instructions



### General

The goal of this task is to find comparative questions. A comparative questions, is a question in which **two** entities (people, objects, companies, other comparable things, ...) or situations are compared to each other. The texts used here are taken from different sources such as Reddit and Yahoo Questions.

*Make sure to have a look at the examples, as not every linguistic comparison is considered a to be tagged with the class "Comparative Question"!*

### First

In the first step you have to read and classify fifteen sentences into the two categories:

- Comparative Question
- Not comparative or not a question

If you are not sure in which category the text belongs, you can select "I am not sure".

### Second

The first time you work on this task you need to provide us with some information about yourself like your age, your country and if you are a native English speaker.

### (Optional Third)

If you have any general comment about this HIT, tell us also in the comment box.

### Bonus

If your answers match with **90%** of the other worker's answers, the reward of the HIT will be **doubled!** The bonus is calculated after the HITs are completed by other workers and might take more than **TWO** days to be paid.

Close

Figure 1: The figure shows the overlay with instructions provided to the workers for sentence classification on MTurk.

## Definition

A comparative questions, is a **question** in which **two** entities/subjects/situations (people, objects, companies, other comparable things, ...) are **compared to each other**.

## Examples

#	Example	Classification
1.	<b>Are oak trees taller than maple trees?</b> This is a comparative question in which oak trees are compared to maple trees.	<b>Comparative Question</b>
2.	<b>Why are Samsung TV's brighter than LGs?</b> This is a comparative question where TV's from Samsung are compared to TV's from LG with a focus on the aspect of brightness.	<b>Comparative Question</b>
3.	<b>Is Gold expensive in comparison with Silver?</b> This is a comparative question in which Gold is compared to Silver with a focus on the costs.	<b>Comparative Question</b>
4.	<b>Why is natural sugar better than added sugar? What's different in the chemical construction?</b> This is a comparative question in which "natural sugar" is compared to "added sugar". Even though there is a second sentence without a comparison, we consider the whole text comparative!	<b>Comparative Question</b>
5.	<b>Is one better than the other?</b> This is <b>not</b> considered a comparative question as, there are no entities compared in this sentence.	<b>Not comparative</b> or not a question
6.	<b>Madrid is different from New York.</b> This is <b>not</b> considered a comparative question, as this is not a question but a statement comparing two cities.	<b>Not comparative</b> or not a question
7.	<b>Is it okay to date someone nine years younger than me?</b> This is <b>not</b> considered a comparative question, as there is no comparison of comparable entities.	<b>Not comparative</b> or not a question
8.	<b>Why is the tide in some places so much higher than in others?</b> This is by our definition <b>not</b> considered a comparative question, as there are no actual entities compared in this sentence. The sentence compares "some places" to "other places". We do <b>not</b> consider this use of pronouns as comparative.	<b>Not comparative</b> or not a question
9.	<b>Which company builds the best cars, Audi, BMW or Ford?</b> This is by our definition <b>not</b> considered a comparative question, as there are more then two entities compared to each other.	<b>Not comparative</b> or not a question
10.	<b>What makes apple airpods different from other wireless earbud makers?</b> There is a clear question and comparative, but apple airpods are compared to the group of all the other wireless earbud makers, except apple. So there is no clear second entity which apple airpods are compared to.	<b>Not comparative</b> or not a question
11.	<b>Why do children learn languages faster than adults ?</b> This is a comparative question in which children are compared to adults regarding the aspect of learning languages.	<b>Comparative Question</b>
12.	<b>Why is it better to bake than fry chicken?</b> This is a comparative question in which baked chicken is compared to fried chicken.	<b>Comparative Question</b>

Figure 2: The figure shows the overlay with examples provided to the workers for sentence classification on MTurk.

```

1| modelName = "SVM Linear"
2| gamma = np.power(10, np.arange(-5, -1, dtype=float))
3| C = [0.001, 0.01, 0.1, 0.5, 1, 2, 5, 10]
4| random_grid = [
5|     {'classifier__kernel': ['rbf'],
6|      'classifier__gamma': gamma,
7|      'classifier__C': C},
8|     {'classifier__kernel': ['sigmoid'],
9|      'classifier__gamma': gamma,
10|      'classifier__C': C},
11|     {'classifier__kernel': ['linear'],
12|      'classifier__C': C},
13|     {
14|         'features__bowTokenNGram': [bowTokenNGram, 'drop'],
15|         'features__bowTokenNGram__vect__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2),
16|                                                         (1,3), (1,4), (1,5)],
17|         'features__bowTokenNGram__vect__max_features' : [100,200,300,400,500,1000, None],
18|         'features__bowPOSNGram': [bowPOSNGram, 'drop'],
19|         'features__bowPOSNGram__vect__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2), (1,3),
20|                                                         (1,4), (1,5)],
21|         'features__bowPOSNGram__vect__max_features' : [100,200,300,400,500,1000, None],
22|         'features__bowDTAGNGram': [bowDTAGNGram, 'drop'],
23|         'features__bowDTAGNGram__vect__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2), (1,3),
24|                                                         (1,4), (1,5)],
25|         'features__bowDTAGNGram__vect__max_features' : [100,200,300,400,500,1000, None],
26|         'features__tfidfTokens': [tfidfTokens, 'drop'],
27|         'features__tfidfTokens__tfidf__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2), (1,3),
28|                                                         (1,4), (1,5)],
29|         'features__tfidfTokens__tfidf__max_features' : [100,200,300,400,500,1000, None],
30|         'features__tfidfTokensPOS': [tfidfTokensPOS, 'drop'],
31|         'features__tfidfTokensPOS__tfidf__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2),
32|                                                         (1,3), (1,4), (1,5)],
33|         'features__tfidfTokensPOS__tfidf__max_features' :
34|             [10,20,30,40,50,100,200,300,400,500,1000, None],
35|     }
36| ]

```

Listing 1: Pseudocode of the random grid search parameter space for the SVM model.

```

1| modelName = "Random Forrest"
2| n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1000, num = 10)]
3| max_features = ['auto', 'sqrt']
4| max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
5| max_depth.append(None)
6| min_samples_split = [2, 5, 10, 12]
7| min_samples_leaf = [1, 2, 4]
8| bootstrap = [True, False]
9| random_grid = {
10|     'classifier__n_estimators': n_estimators,
11|     'classifier__max_features': max_features,
12|     'classifier__max_depth': max_depth,
13|     'classifier__min_samples_split': min_samples_split,
14|     'classifier__min_samples_leaf': min_samples_leaf,
15|     'classifier__bootstrap': bootstrap,
16|     'features__bowTokenNGram': [bowTokenNGram, 'drop'],
17|     'features__bowTokenNGram__vect__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2), (1,3),
18|                                                         (1,4), (1,5)],
19|     'features__bowTokenNGram__vect__max_features' : [100,200,300,400,500,1000, None],
20|     'features__bowPOSNGram': [bowPOSNGram, 'drop'],
21|     'features__bowPOSNGram__vect__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2), (1,3),
22|                                                         (1,4), (1,5)],
23|     'features__bowPOSNGram__vect__max_features' : [100,200,300,400,500,1000, None],
24|     'features__bowDTAGNGram': [bowDTAGNGram, 'drop'],
25|     'features__bowDTAGNGram__vect__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2), (1,3),
26|                                                         (1,4), (1,5)],
27|     'features__bowDTAGNGram__vect__max_features' : [100,200,300,400,500,1000, None],

```



```

25| 'features__tfidfTokens': [tfidfTokens, 'drop'],
26| 'features__tfidfTokens__tfidf__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2), (1,3)
    | , (1,4), (1,5)],
27| 'features__tfidfTokens__tfidf__max_features': [100,200,300,400,500,1000, None],
28| 'features__tfidfTokensPOS': [tfidfTokensPOS, 'drop'],
29| 'features__tfidfTokensPOS__tfidf__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2)
    | , (1,3), (1,4), (1,5)],
30| 'features__tfidfTokensPOS__tfidf__max_features':
    | [10,20,30,40,50,100,200,300,400,500,1000, None],
31| }

```

Listing 2: Pseudocode of the random grid search parameter space for the Random Forrest model.

```

1| modelName = "mlp"
2| random_grid = {
3|   'classifier__hidden_layer_sizes': [(50,50,50), (50,100,50), (100,)],
4|   'classifier__activation': ['tanh', 'relu'],
5|   'classifier__solver': ['sgd', 'adam'],
6|   'classifier__alpha': [0.0001, 0.05],
7|   'classifier__learning_rate': ['constant', 'adaptive'],
8|   'features__bowTokenNGram': [bowTokenNGram, 'drop'],
9|   'features__bowTokenNGram__vect__ngram_range' : [(1,4)],
10|  'features__bowTokenNGram__vect__max_features': [100,200,300,400,500,1000, None],
11|  'features__bowPOSNGram': [bowPOSNGram, 'drop'],
12|  'features__bowPOSNGram__vect__ngram_range' : [(3,3)],
13|  'features__bowPOSNGram__vect__max_features': [100,200,300,400,500,1000, None],
14|  'features__bowDTAGNGram': [bowDTAGNGram, 'drop'],
15|  'features__bowDTAGNGram__vect__ngram_range' : [(2,2)],
16|  'features__bowDTAGNGram__vect__max_features': [100,200,300,400,500,1000, None],
17|  'features__tfidfTokens': [tfidfTokens, 'drop'],
18|  'features__tfidfTokens__tfidf__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2), (1,3)
    | , (1,4), (1,5)],
19|  'features__tfidfTokens__tfidf__max_features': [100,200,300,400,500,1000, None],
20|  'features__tfidfTokensPOS': [tfidfTokensPOS, 'drop'],
21|  'features__tfidfTokensPOS__tfidf__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2)
    | , (1,3), (1,4), (1,5)],
22|  'features__tfidfTokensPOS__tfidf__max_features':
    | [10,20,30,40,50,100,200,300,400,500,1000, None],
23| }

```

Listing 3: Pseudocode of the random grid search parameter space for the MLP model.

```

1| modelName = "etree"
2| n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
3| max_features = ['auto', 'sqrt']
4| max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
5| max_depth.append(None)
6| min_samples_split = [2, 5, 10, 12]
7| min_samples_leaf = [1, 2, 4]
8| bootstrap = [True, False]
9| random_grid = {
10|   'classifier__n_estimators': n_estimators,
11|   'classifier__max_features': max_features,
12|   'classifier__max_depth': max_depth,
13|   'classifier__min_samples_split': min_samples_split,
14|   'classifier__min_samples_leaf': min_samples_leaf,
15|   'classifier__bootstrap': bootstrap,
16|   'features__bowTokenNGram': [bowTokenNGram, 'drop'],
17|   'features__bowTokenNGram__vect__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2)
    | , (1,3), (1,4), (1,5)],
18|   'features__bowTokenNGram__vect__max_features': [100,200,300,400,500,1000, None],
19|   'features__bowPOSNGram': [bowPOSNGram, 'drop'],
20|   'features__bowPOSNGram__vect__ngram_range' : [(1,1), (2,2), (3,3), (4,4), (5,5), (1,2)
    | , (1,3), (1,4), (1,5)],

```

```

21| 'features__bowPOSNGram__vect__max_features': [100,200,300,400,500,1000,None],
22| 'features__bowDTAGNGram': [bowDTAGNGram,'drop'],
23|   'features__bowDTAGNGram__vect__ngram_range': [(1,1),(2,2),(3,3),(4,4),(5,5),(1,2)
24|     ,(1,3),(1,4),(1,5)],
25| 'features__bowDTAGNGram__vect__max_features': [100,200,300,400,500,1000,None],
26| 'features__tfidfTokens': [tfidfTokens,'drop'],
27|   'features__tfidfTokens__tfidf__ngram_range': [(1,1),(2,2),(3,3),(4,4),(5,5),(1,2)
28|     ,(1,3),(1,4),(1,5)],
29| 'features__tfidfTokens__tfidf__max_features': [100,200,300,400,500,1000,None],
30| 'features__tfidfTokensPOS': [tfidfTokensPOS,'drop'],
31|   'features__tfidfTokensPOS__tfidf__ngram_range': [(1,1),(2,2),(3,3),(4,4),(5,5),(1,2)
32|     ,(1,3),(1,4),(1,5)],
33| 'features__tfidfTokensPOS__tfidf__max_features':
34|   [10,20,30,40,50,100,200,300,400,500,1000,None],
35| }

```

Listing 4: Pseudocode of the random grid search parameter space for the Etree model.

```

1| modelName = "SGD"
2| modelParameters = {
3|   "classifier__loss" : ["hinge","log"],
4|   "classifier__alpha" : [0.001, 0.01, 0.1, 0.5, 1, 2,5, 10],
5|   "classifier__penalty" : ["l2","l1"],
6|   'features__bowTokenNGram__vect__ngram_range' : [(1,1),(2,2),(3,3),(4,4),(5,5),(1,2),(1,3)
7|     ,(1,4),(1,5)],
8|   'features__bowTokenNGram__vect__max_features': [100,200,300,400,500,1000,None],
9|   'features__bowPOSNGram': [bowPOSNGram,'drop'],
10|   'features__bowPOSNGram__vect__ngram_range' : [(1,1),(2,2),(3,3),(4,4),(5,5),(1,2),(1,3)
11|     ,(1,4),(1,5)],
12|   'features__bowPOSNGram__vect__max_features': [100,200,300,400,500,1000,None],
13|   'features__bowDTAGNGram': [bowDTAGNGram,'drop'],
14|   'features__bowDTAGNGram__vect__ngram_range' : [(1,1),(2,2),(3,3),(4,4),(5,5),(1,2),(1,3)
15|     ,(1,4),(1,5)],
16|   'features__bowDTAGNGram__vect__max_features': [100,200,300,400,500,1000,None],
17| }

```

Listing 5: Pseudocode of the grid search parameter space for the SGD model.

```

1| modelName = "Logistic Reg"
2| C = [0.001, 0.01, 0.1, 0.5, 1, 2,5, 10]
3| modelParameters = {
4|   'classifier__C': C
5|   'classifier__max_iter': [10,50,100,200,300,400,500],
6|   'features__bowTokenNGram__vect__ngram_range' : [(1,1),(2,2),(3,3),(4,4),(5,5),(1,2),(1,3)
7|     ,(1,4),(1,5)],
8|   'features__bowTokenNGram__vect__max_features': [100,200,300,400,500,1000,None],
9|   'features__bowPOSNGram': [bowPOSNGram,'drop'],
10|   'features__bowPOSNGram__vect__ngram_range' : [(1,1),(2,2),(3,3),(4,4),(5,5),(1,2),(1,3)
11|     ,(1,4),(1,5)],
12|   'features__bowPOSNGram__vect__max_features': [100,200,300,400,500,1000,None],
13|   'features__bowDTAGNGram': [bowDTAGNGram,'drop'],
14|   'features__bowDTAGNGram__vect__ngram_range' : [(1,1),(2,2),(3,3),(4,4),(5,5),(1,2),(1,3)
15|     ,(1,4),(1,5)],
16|   'features__bowDTAGNGram__vect__max_features': [100,200,300,400,500,1000,None],
17| }

```

Listing 6: Pseudocode of the grid search parameter space for the Logistic Regression model.

```

1| svcParameters = {
2|   'classifier__kernel': 'linear',
3|   'classifier__C': 0.01,
4|   'features__bowTokenNGram__vect__ngram_range' : (1,4),

```

```

5 |     'features__bowPOSNGram__vect__ngram_range' : (1,2),
6 |     'features__bowDTAGNGram__vect__ngram_range' : (1,5),
7 |     'features__bowTokenNGram__vect__max_features' : None,
8 |     'features__bowPOSNGram__vect__max_features' : 300,
9 |     'features__bowDTAGNGram__vect__max_features' : 200,}
10 |
11 | randForestParameters = {
12 |     'classifier__n_estimators': 900,
13 |     'classifier__max_features': 'auto',
14 |     'classifier__max_depth': 70,
15 |     'classifier__min_samples_split': 10,
16 |     'classifier__min_samples_leaf': 1,
17 |     'classifier__bootstrap': True,
18 |     'classifier__random_state': 42,
19 |     'features__bowTokenNGram__vect__ngram_range' : (1,5),
20 |     'features__bowPOSNGram__vect__ngram_range' : (1,4),
21 |     'features__bowDTAGNGram__vect__ngram_range' : (1,1)}
22 |
23 | logRegParameters = {
24 |     'classifier__max_iter': 500,
25 |     'classifier__random_state': 42,
26 |     'features__bowTokenNGram__vect__ngram_range' : (1,5),
27 |     'features__bowPOSNGram__vect__ngram_range' : (1,4),
28 |     'features__bowDTAGNGram__vect__ngram_range' : (1,1),}
29 |
30 | sgdParameters = {
31 |     "classifier__loss" : "hinge",
32 |     "classifier__alpha" : 0.01,
33 |     "classifier__penalty" : "l2",
34 |     'features__bowTokenNGram__vect__ngram_range' : (1,5),
35 |     'features__bowPOSNGram__vect__ngram_range' : (1,4),
36 |     'features__bowDTAGNGram__vect__ngram_range' : (1,1)}
37 |
38 | bernNBParameters = {
39 |     'features__bowTokenNGram__vect__ngram_range' : (1,5),
40 |     'features__bowPOSNGram__vect__ngram_range' : (1,4),
41 |     'features__bowDTAGNGram__vect__ngram_range' : (1,1)}
42 |
43 | etreeParameters = {
44 |     'classifier__n_estimators': 200,
45 |     'classifier__min_samples_split': 5,
46 |     'classifier__min_samples_leaf': 2,
47 |     'classifier__max_features': 'sqrt',
48 |     'classifier__max_depth': 60,
49 |     'classifier__bootstrap': False,
50 |     'features__bowTokenNGram__vect__ngram_range' : (1,5),
51 |     'features__bowPOSNGram__vect__ngram_range' : (1,4),
52 |     'features__bowDTAGNGram__vect__ngram_range' : (1,1)}
53 |
54 | xgbParameters = {
55 |     'features__bowTokenNGram__vect__ngram_range' : (1,5),
56 |     'features__bowPOSNGram__vect__ngram_range' : (1,4),
57 |     'features__bowDTAGNGram__vect__ngram_range' : (1,1)}
58 |
59 | mlpParameters = {
60 |     'classifier__max_iter': 500,
61 |     'classifier__solver': 'sgd',
62 |     'classifier__learning_rate': 'adaptive',
63 |     'classifier__hidden_layer_sizes': (50, 50, 50),
64 |     'classifier__alpha': 0.001,
65 |     'classifier__activation': 'relu',
66 |     'classifier__random_state': 42,
67 |     'features__bowTokenNGram__vect__ngram_range': (1, 4),
68 |     'features__bowPOSNGram__vect__ngram_range': (3, 3),
69 |     'features__bowDTAGNGram__vect__ngram_range': (3, 3)}

```

Listing 7: Pseudocode of the training parameters for the the feature-base ML models (see Section 4.1.2).



# Versicherung an Eides statt

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudien-  
gang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel -  
insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen - benutzt habe.  
Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind  
als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht  
in einem anderen eingereicht habe und die eingereichte schriftliche Fassung der auf dem  
elektronischen Speichermedium entspricht.

Hamburg, den September 11, 2020

Steffen Stahlhacke