Question Answering Technologies Behind IBM Watson

Project Lab Report

Stefan Bauregger, Timo Gerecht, Daniel Theiß, Ute Winchenbach

October 15, 2015

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Abstract

This lab report describes the results of one project implemented for the lab on *Question Answering Technologies Behind IBM Watson* that was held at the Technische Universität Darmstadt by Prof. Dr. Chris Biemann, Seid Muhie Yimam, Steffen Remus and supervised by Uli Fahrer. The presented application employs the IBM Watson QA service on a corpus of ”The Simpsons” documents. Natural language questions are answered by Watson and further processed into sentence and multiple word long answers. These answers are either presented on their own or used as input for an user-generated Quiz application. An Android-App and a Web application are presented as user interfaces for this system.

Contents

1. Introduction **3**
1.1. Component Overview . . . 3
1.1.1. Task Responsibilities . . . 3

2. Answer Processing Pipeline **5**
2.1. Components . . . 5
2.1.1. Sentence Selection . . . 5
2.1.2. Named Entity Extraction . . . 5
2.1.3. Quiz Answer Generation . . . 7
2.2. Design Choices . . . 8
2.3. Evaluation . . . 8

3. Quiz **11**
3.1. Single Player Game . . . 11
3.2. Multi Player Game . . . 11
3.3. Quiz Questions . . . 11
3.3.1. Review . . . 11
3.3.2. Acceptance Criteria . . . 12
3.4. Highscore . . . 12

4. User Interface (UI) **13**
4.1. Android Application . . . 13
4.1.1. Tools . . . 13
4.1.2. Components . . . 13
4.1.3. Design Choices . . . 16
4.2. Webinterface . . . 17
4.2.1. Tools . . . 17
4.2.2. Components . . . 17
4.2.3. Design Choices . . . 19

5. NLP Server **24**
5.1. Language Processing . . . 24

6. Quiz Backend **25**
6.1. Database . . . 25
6.2. REST API . . . 25
6.3. Java-Client . . . 28

7. Corpus **29**
7.1. Contents . . . 29
7.2. Training data . . . 29

8. Imagefinder **30**
8.1. allImagesQuery . . . 30
8.2. imagesOnArticleQuery . . . 30
8.3. thumbImageQuery . . . 30

A. Training Data **31**

Bibliography **32**

1 Introduction

This lab report describes an application that uses the *IBM Watson QA service* [IBM15] as a black-box technology to answer and further process natural language questions about the television series "The Simpsons". The resulting system is able to provide sentence or multiple word long answers that are used as direct answers or input into a Quiz Game. The Quiz Game is based on user-generated content that is assisted by automatically generated quiz answers. It offers single and two player local multi-player play. The application is implemented as a mobile application and a web application. This report offers insight into the application's core components, their underlying design, used tools and provided sources and gives user guidance.

1.1 Component Overview

The presented system consists of four main components: the *User Interface*, the *Answer Processing Pipeline*, the *NLP Server* and the *Quiz Backend*. Furthermore, it uses the *IBM Watson QA service* [IBM15] as a fifth external component. An overview of the interaction of the different components is given in Figure 1.1. More details on their implementation are provided in the next chapters.
The *IBM Watson* component is an IBM Watson Private Instance that was trained on a corpus of "The Simpsons" documents with 1000 question-answer pairs via the Watson Experience Manager. The corpus contains 1678 episode, character and location documents from the Simpsons Wikia [Wik15a]. It is accessed through RESTful API via a Java wrapper that was implemented by Uli Fahrer [Fah15a]. The IBM Watson module receives natural language questions and provides paragraph long answers from the collection of Simpsons documents.
These answers are further processed into sentence or multiple word long answers by the *Answer Processing Pipeline*. This module selects shorter text excerpts that are most likely to answer the given question. For this purpose, it analyses question as well as answer text features that are annotated by the *Natural Language Processing (NLP) Server*. This server uses the Stanford Core NLP Toolkit [MSB+14] for e.g. text segmentation, Part-of-Speech and Named Entity annotation.
The initial natural language questions are received from the *User Interface*, which also displays and further uses the resulting answers. An Android app and a web application are implemented as user interfaces. They offer direct answers to question input or use the generated answers in a Quiz Game. The Quiz Game stores on and retrieves user and quiz question information from the *Quiz Backend Server*, a MySQL database with PHP frontend.

1.1.1 Task Responsibilities

Android Application:	Stefan Bauregger
NLP Server:	Daniel Theiß
Quiz Backend:	Daniel Theiß
Answer Processing Pipeline:	Timo Gerecht and Ute Winchenbach
Corpus Management and Watson Training:	All
Web Application:	All

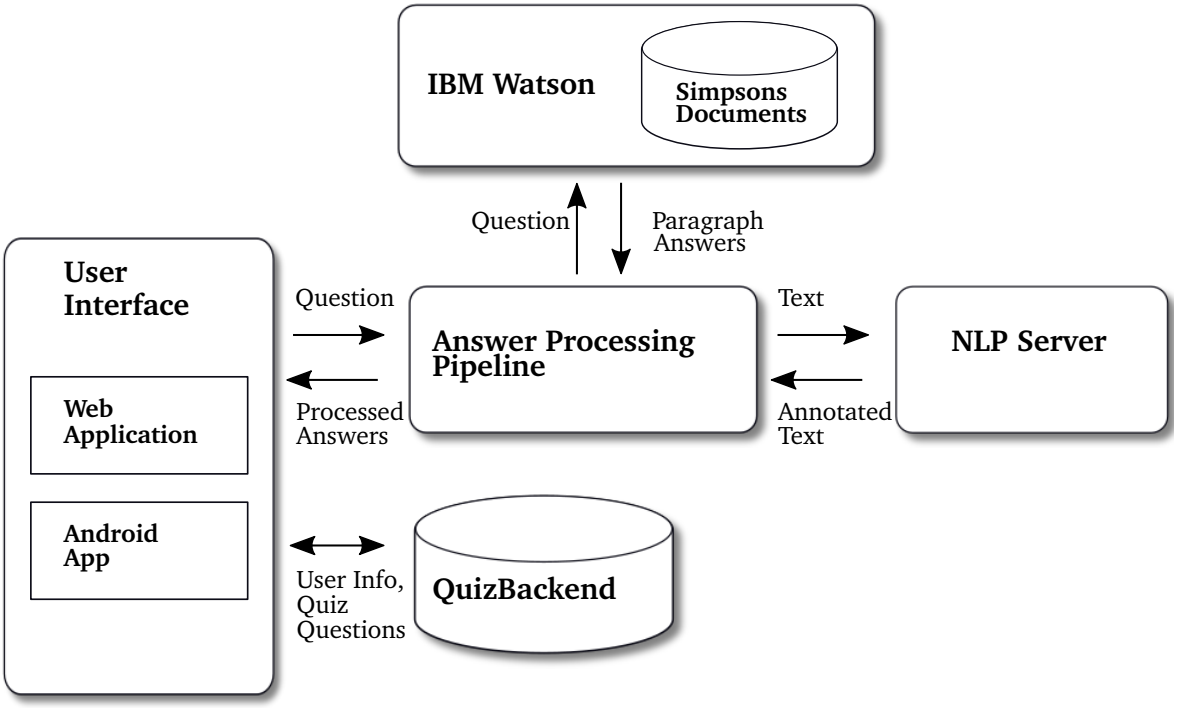


Figure 1.1.: Component overview.

2 Answer Processing Pipeline

This chapter gives an overview of the Answer Processing Pipeline, which further processes IBM Watson answers to obtain shorter answers. The underlying Sentence Selection, Named Entity Extraction and Quiz Answer Generation are discussed. Moreover, different scoring functions that are used in Sentence and Named Entity Selection are evaluated.

2.1 Components

The Answer Processing Pipeline takes questions and paragraph long answers as input from the IBM Watson component. Question and answers are then annotated by the NLP Server (cf. Chapter 5). These annotations are processed by the Answer Processing Pipeline into sentence long answers. This is done by selecting the sentences that are the most similar to the given question or show desired properties. This sentence selection is made with the use of a scoring function. Shorter answers that are one or more words long and consist of named entities are extracted after the sentence selection. Here, another scoring function is used which considers the previous sentence score and the presence of named entities in the sentence. The collection of short answers can be expanded by the Quiz Answer Generation and then used as input for the Quiz Game. The Answer Processing Pipeline can be executed from the *QuizPipeline* class via `executePipeline()` or `executeQAPipeline()` and returns only extracted answers or extracted and generated ones. Figure 2.1 illustrates the pipeline execution and intermediate results.

2.1.1 Sentence Selection

The Sentence Selection is based on selecting sentences that are the most similar to the question or show certain properties. For this, the question and answer texts are segmented into sentences. A further pre-processing step of stopword removal is optional.
The similarity is then analysed by comparing words, token n-grams or character n-grams that exist in the question text and the answer text. Here, the NLP server (cf. Chapter 5) annotations for Part-of-Speech (POS), tokens and named entities are used. The occurences of n-grams or annotated words are counted (*NGramFreqCounter* class and its subclasses) in question and answer texts and then compared to each other. The comparison is done via a distance measure (classes that implement *NGramDistEvaluator*). Two distance measures are used, a summed differences measure (*NGramDifference* class) and a Simple N-Gram Profile Intersection (SPI) [FSGK06] which is used as a dissimilarity measure (*SpiNGramSim* class). The resulting distance score can be used in a combination with other distance scores. This is done in the *NGramSentenceAnswerExtractor* class that can be configured with a distance measure and multiple n-gram or word counters. The distance scores are transformed into a similarity score by subtracting it from one.
In addition to the similarity analysis some simple feature extractors (*AnswerFeatureExtraction* class) are implemented that determine if an answer shows certain properties. Here, metadata that is returned by IBM Watson is also used. It is examined if an answer uses synonyms of words that are part of the question. Moreover, it can be determined if an answer uses types of named entities that suit the type of question (e.g. the question might be a person type like “Who is Homer’s wife?”, then it is tested if the answer contains named entities of type PERSON). These feature extractors also result in scores that can be combined with the similarity scores to give an overall scoring function. The different scores are weighted and combined with a confidence score. This confidence score is returned from Watson and gives an indication on how suitable the parent text paragraph is as an answer. Finally, the answer sentence with the highest score is the sentence that most likely answers the given question. An evaluation of different scoring functions is done (cf. Section 2.3) and the final implementation uses the best performing one. The final score combines character bigram differences and character trigram SPI with the Watson confidence factor and a boolean feature that matches contained Named Entity types to the question type.

$$Score = 0.1 \cdot NEType\text{-}Score + 0.2 \cdot (1 - Char\text{-}Bigram\text{-}Dist) + 0.3 \cdot (1 - Char\text{-}Trigram\text{-}SPI) + 0.5 \cdot Confidence$$

2.1.2 Named Entity Extraction

The selection of a shorter answer which is only a few words long from a given sentence is based on Named Entity Extraction (NER).

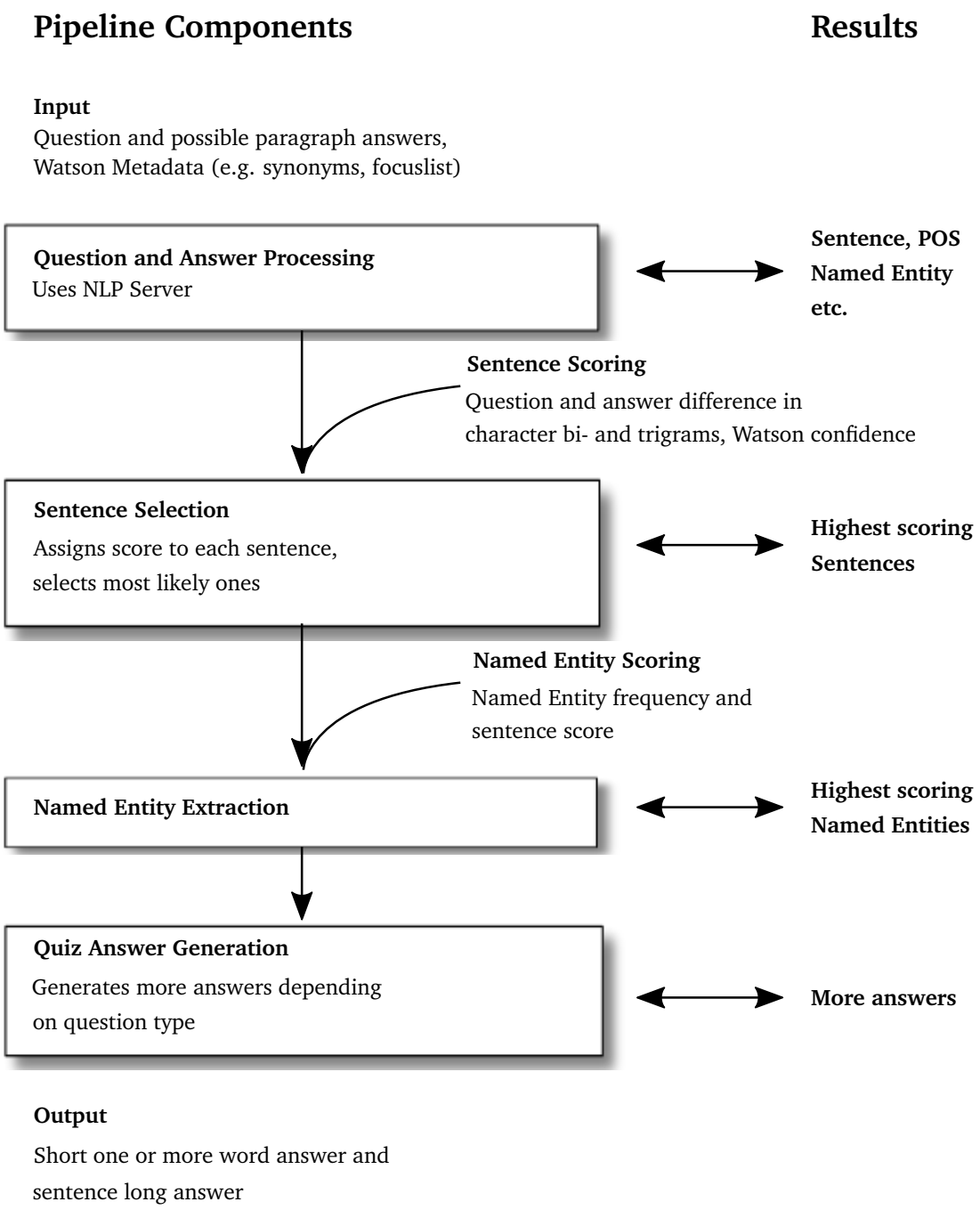


Figure 2.1.: Answer Processing Pipeline.

For the extraction we use our NLP Server (cf. Chapter 5) which we have extended to support Entities like *Homer*, *Amber* or *Snowball II*. We also expand the underlying models with our own Entity types DISTANCE, WEIGHT, COLOR and add some special types to combine different Named Entities (NE).
The Named Entities Extraction is able to compose Entities from different NER types.
Examples of a composite of multiple entities are:

- Homer Simpson: PERSON PERSON -> PERSON
- Ned Flanders' House : PERSON PERSON LOCATION -> LOCATION/ORGANIZATION
- 744 Evergreen Terrace : NUMBER LOCATION LOCATION -> LOCATION/ORGANIZATION
- 30 Meter : NUMBER DISTANCE -> DISTANCE
- 400 Pounds : NUMBER WEIGHT -> WEIGHT
- Moe's Tavern : PERSON ORGANIZATION -> LOCATION/ORGANIZATION
- Comic Book Guy : MISC MISC MISC -> PERSON
- 12th Avenue: ORDINAL LOCATION -> LOCATION/ORGANIZATION

Altogether we extract the following Named Entities from the selected sentence depending on a given question:

- **Person**, **Location/Organization**
- **Date**, **Duration**
- **Number**, **Money**
- **Weight**
- **Distance**
- **Color**

The extraction of different NEs from the same sentence with concatenated entities depends on the question type. If we ask for a Person, the answer extraction from the sentence "The address of Ned Flanders' house is 744 Evergreen Terrace." results in "Ned Flanders". If we ask for a Location the results are "Ned Flanders' house" and "744 Evergreen Terrace". A question about a Number results in "744".
After we have extracted the Named Entities, we compute their score consisting of the frequency and the sentence score, which are weighted with 10 percent and 90 percent respectively. These weights are chosen after an evaluation of different combinations of NE frequency and sentence score (cf. Section 2.3). The Named Entities with the highest score are considered the most suitable answers to the given question.

$$Score = 0.9 \cdot sentence\text{-}score + 0.1 \cdot occurrence\ of\ NE$$

2.1.3 Quiz Answer Generation

The Quiz Answer Generation is based on the Named Entity Extraction and always returns a certain number of short answers to a given question. In the first step of our Quiz Answer Generation we determine the question type. For this we use the obtained Watson Answer.
The first stage of processing in the IBM Watson system is to perform a detailed analysis of the question. [LPM+12]
Watson detects critical elements of the question, including:

1. **the focus:** the part of the question that is a reference to the answer
2. **lexical answer types (LAT):** terms in the question that indicate what type of entity is being asked for
3. **qClass:** a classification of the question into one or more of several broad types

We have access to this analysis by Watson via the Watson Answering response which includes this information in a qClass List, Focus List and LAT List. The qClass List provides information whether the question is about a Number or Date. Using the Focus List, we know whether it is a ”Why, When, Who, Whose...”-question. ”Who, Whose”-questions imply ”Person” as a question type. The LAT List gives information according to who or what is asked. So we can differentiate the question type based on keywords that appear in the LAT list. For example, if ”episode” or ”season” is in the list, our question type is ”Episode” and ”Season”. We also use the LAT list to distinguish between female, male persons and animal NEs. If we cannot find a question type based on the three Watson lists, we compare key words in the whole question text.
The Quiz Answer Generation executes the Answer Processing Pipeline for sentence answers and extracts shorter answers if the question type allows it. It is for example not possible to get a question type for a ”Why”-question that would allow to specify a shorter answer.
Based on the specific question type we extract matching composed Named Entities. A special case are the question types ”Episode” and ”Season”, because no Named Entity processing is used here. Instead we use the highest scoring sentence that results from the Sentence Selection and is part of an episode description. The season and episode information can then be found in the title of the document that this sentence originates from.
After we have extracted our quiz answers we generate more alternative answers if necessary. These generated answers are based on random numbers for question types like ”Number”, ”Money” or ”Percentage” and names from article lists of the Simpsons Wiki [Wik15a] for other question types.

2.2 Design Choices

- The question type determination was designed to be expandable for different types.
- The similarity analysis was designed to be expandable for different types of n-grams or annotated words and configurable with different distance measures.

2.3 Evaluation

The sentence selection of the Answer Processing Pipeline is evaluated with regard to different scoring functions. The input for these tests consists of the first three Watson answers that are each of paragraph length. The test data consists of 100 question-answer pairs that contain 48 training questions and 52 unseen ones. The questions vary in type (e.g. person, location, reason). The answers that are given by the application are considered correct if they contain a certain word or sequence of words (e.g. an answer which contains “Smithers” like “Waylon Joseph Smithers, Jr. (better known as "Mr. Smithers" or simply "Smithers") is Mr. Burns’ personal assistant, executive, and self-proclaimed best friend.” is considered correct for the question “Who is Mr Burns’ assistant?”). As evaluation metric the rate of correct answers to all questions is calculated. This is done for the case that the correct answer is found as the first returned answer. Moreover, it is done for the cases that it is found in the first two results (first answer and one alternative), the first three and so on. Several methods are tried in different combinations with varying weighting factors for the scoring function. As pre-processing methods stopword removal and case insensitivity are tested. The methods are:

- **Confidence**: The Confidence Factor that is returned by IBM Watson
- **Char-[n]gram-Dist** or **Token-[n]gram-Dist**: Distance in character or token n-grams
- **Char-[n]gram-Overlap** or **Token-[n]gram-Overlap**: SPI in character or token n-grams
- **NE-Overlap**: SPI in Named Entities
- **NEType-Score**: The presence of Named Entity Types (e.g. LOCATION, PERSON, NUMBER)
- **Synonym-Score**: The presence of synonyms of question words that are found in the answer
- **Relative-Synonym-Score**: The rate of synonyms of question words that are found in the answer

The results of the evaluation can be found in Table 2.2. The table shows the tested scoring functions with their used pre-processing methods and their respective accuracy when only the first or up to four alternatives are considered.
The results show that the employed methods can improve the selection of correct answers when compared to Watson answers on their own (Confidence with a score of 49%). The scoring functions correctly identify relevant answers for 42 to 53 percent of the tests. This increases to results between 68 and 77 percent if more alternatives are considered.

The best results are achieved with a combination of Confidence, character bi- and trigrams and a boolean feature for contained NE types. However, most methods do not show large differences when compared to each other.
Character trigrams seem to perform well in the evaluation. However, their performance can be slightly improved when more methods are added to the scoring function. Token or specific annotation distances do not score as high but their performance can be improved by adding more methods to the scoring function. The inclusion of pre-processing methods does not noticeably improve the results.
The evaluation results indicate that the distance measure SPI (referred to as Overlap) might perform better than the summed distance measure when tested on their own. On the other hand, when both measures are combined with character n-gram features, they score well (a score of 53 % for the final scoring-function of *0.4 Confidence+ 0.3 Char-Trigram-Overlap+ 0.2 Char-Bigram-Dist+ 0.1 NEType-Score*).
The Named Entity Extraction that offers shorter answers is also evaluated. The most suitable named entity answer is found through another scoring function that combines NE frequency and the previously discussed Sentence Selection score. Evaluation results of differently weighed scoring functions can be found in Table 2.1. The NE occurrence weight describes how much the count of relevant named entities is emphasized. The Sentence Selection weight puts emphasis on the score for the sentence that contains the current named entity. The tested weights for the NE occurrence are 0.06, 0.08, 0.1, 0.12 and 0.14, the respective Sentence Selection weights are calculated by subtraction these from one. Correctness checks and evaluation metrics are the same as for the Sentence Selection evaluation. The number of processed Watson answers is five and the test data contains 100 question-answer pairs. The test data contains most of the previous test questions with replacements for questions, which do not allow named entities as answers (e.g. ”Why”-questions).
The results show values between 56 and 59 % of correct answers that are given as the first answer. These values increase with the number of considered alternatives. The evaluation scores also increase at the beginning with bigger NE occurrence weights but start to decrease again after the best performing weight of 0.1. This might be explained with a bias towards overall very frequent words (like ”Homer”) that has to be compensated by the Sentence Selection score in the scoring function.

NE occurrence weight	Sentence Selection weight	Correct/All	1 alternative	2 alt.	3 alt.	4 alt.
0.06	0.94	0.58	0.62	0.66	0.67	0.68
0.08	0.92	0.59	0.64	0.68	0.68	0.69
0.10	0.90	0.59	0.64	0.68	0.69	0.69
0.12	0.88	0.56	0.61	0.66	0.66	0.67
0.14	0.86	0.56	0.64	0.65	0.66	0.66

Table 2.1.: Named Entity extraction evaluation for differently weighed scoring functions.

Scoring function	Pre-Processing	Correct/All	1 alternative	2 alt.	3 alt.	4 alt.
Confidence	Case	0.49	0.59	0.65	0.68	0.70
0.5 Confidence+ 0.5 Char-Bigram-Dist	Case	0.45	0.61	0.68	0.71	0.72
0.5 Confidence+ 0.5 Char-Trigram-Overlap	Case	0.51	0.62	0.67	0.74	0.77
0.5 Confidence+ 0.5 Token-Unigram-Dist	Case	0.48	0.61	0.64	0.69	0.73
0.5 Confidence+ 0.5 Token-Unigram-Overlap	Case	0.49	0.59	0.66	0.70	0.72
0.5 Confidence+ 0.5 Char-Trigram-Overlap	Case, Stopwords Removal	0.49	0.61	0.67	0.71	0.74
0.5 Confidence+ 0.5 Token-Unigram-Overlap	Case, Stopwords Removal	0.43	0.64	0.67	0.69	0.73
0.5 Confidence+ 0.5 NE-Overlap	Case	0.42	0.51	0.57	0.63	0.68
0.5 Confidence+ 0.3 Char-Trigram-Overlap+ 0.2 Token-Unigram-Overlap	Case, Token: Stopwords Removal	0.48	0.65	0.68	0.73	0.76
0.5 Confidence+ 0.3 Char-Trigram-Overlap+ 0.2 NE-Overlap	Case	0.47	0.63	0.64	0.68	0.73
0.5 Confidence+ 0.3 Char-Trigram-Overlap+ 0.2 NEType-Score	Case	0.50	0.66	0.70	0.72	0.75
0.5 Confidence+ 0.3 Char-Trigram-Overlap+ 0.2 Relative-Synonym-Score	Case	0.47	0.64	0.66	0.72	0.75
0.5 Confidence+ 0.3 Char-Trigram-Overlap+ 0.1 NEType-Score+ 0.1 Synonym-Score	Case	0.46	0.62	0.67	0.70	0.72
0.5 Confidence+ 0.3 Char-Trigram-Overlap+ 0.2 Char-Bigram-Dist	Case	0.51	0.65	0.70	0.73	0.75
0.4 Confidence+ 0.3 Char-Trigram-Overlap+ 0.1 NEType-Score+ 0.2 Char-Bigram-Dist	Case	0.53	0.66	0.70	0.72	0.77

Table 2.2.: Sentence selection evaluation for different scoring functions.

3 Quiz

The following sections explain the rules of the Simpsons Quiz.

3.1 Single Player Game

A single player game contains 9 questions of ascending difficulty (always three questions of easy, medium and hard difficulty). Each correct answer scores 10 points, wrong answers score 0 points. The question has to be answered within 20 seconds. If the timelimit is over, the question cannot be answered and the player scores 0 points.

3.2 Multi Player Game

Like the single player game, a multi player game contains 9 questions of ascending difficulty. A multi player game is played by two players locally. A question is answered in several stages:

1. **Buzzing:** Each player has the possibility to press a buzzer button. The player who presses his buzzer first gets the chance to answer the question. If no buzzer was pressed during 10 seconds, the buzzers are deactivated and the question can't be answered.

2. **Answering:** The player, who pressed his buzzer first can now answer the question. If he answers the question correctly, he scores 20 points. If he fails to answer correctly, he scores -10 points. He also scores -10 points if the question isn't answered within 15 seconds.

3. **Second Chance:** If the first player failed to answer the question correctly, the second player gets the chance to answer it correctly. He gets no negative score for wrong answers, but a score of 15 points for a correct answer. The timelimit for the second chance amounts to 5 seconds.

After 9 questions the user with the highest score wins. If both players scored the same amount of points, the game ends in a draw.

3.3 Quiz Questions

A quiz question consists of a question, one correct answer and three wrong answers (distractors). Each user can create and review questions for the game. The following sections describe the reviewing process and the acceptance criteria for questions to appear in the Single Player and Multi Player Quiz Games.

3.3.1 Review

When reviewing a question, the user has to answer the following questions:

- The answer marked as correct was the correct answer to the question.
- The distractors (wrong answers) were wisely chosen.
- The question was formulated well.
- How difficult is the question to answer?

The difficulty can be evaluated by an integer value between 1 (very easy) and 9 (very hard).
Each question can be reviewed once per user. A user cannot review a question he created by himself.

3.3.2 Acceptance Criteria

A question appears in the Single and Multi Player Quiz Game if it matches the following criteria:

- The question has been reviewed a minimum of three times.
- At least 80% of the reviewers stated, that the answer marked as correct was the correct answer to the question.
- At least 60% of the reviewers stated, that the distractors were wisely chosen.
- At least 50% of the reviewers stated, that the question was formulated well.

The acceptance criteria guarantee correctness of the question (highest priority), meaningfulness of the possible answers and correct phrasing of the question (lowest priority, since this is quite subjective).
The difficulty of a question is determined by calculating the average difficulty from the reviews. The question is evaluated as easy if the average difficulty lies between 1 and 3.5, as medium for difficulty values between 3.5 and 6.5 and hard for difficulty values above 6.5.

3.4 Highscore

The quiz game contains a highscore, that shows the 10 most active users in creating and reviewing questions. For each created and accepted question a user scores 5 points and 1 point for each review. The 10 users with the highest score are displayed in the highscore.
The highscore offers an incentive for users to create and review questions.

4 User Interface (UI)

4.1 Android Application

4.1.1 Tools

The Android application was created using Android Studio 1.4 which is based on IntelliJ IDEA [Goo15].

4.1.2 Components

The Android application consists of several views. Views are created by Activities, which each resemble one screen of the app. In Figure 4.1 the activities of the SimpsonsQA application are displayed. The arrows represent possible ways of navigating through the views. The following sections contain rough explanations of the activities and the corresponding workflows.

LoginRegisterActivity

- Create a new account
- Login to an existing account
- Automatic redirection to Startpage, if resumable session exists

StartPageActivity

This activity contains the main screen of the app (cf. Figure 4.2a), consisting of ImageButtons, that enable the user to start a QuizGame, to add new Questions, to review new Questions, to enter the Highscore View or to open a Help Dialog. Via the menu from the action bar, the QA mode can be started.

ReviewActivity

The ReviewActivity (cf. Figure 4.4c) enables the user to review questions that have been created recently. After creation the activity loads new questions ordered by the times they have been reviewed (fewest first). After answering a question, the user can fill out a review form, that contains the following questions (cf. Section 3.3.1):

- The green marked answer was the correct answer to the question.
- The distractors (wrong answers) were wisely chosen.
- The question was formulated well.
- How difficult is the question to answer?

After the review has been submitted or if no review was submitted, the user can choose to load the next question to review.

QuizGameActivity

The QuizGameActivity starts a single player quiz game containing 9 questions. The rules of the quiz game are explained in Section 3.1. A screenshot of the quiz game can be seen in Figure 4.2c. After finishing the quiz, the user can choose to return to the Startpage or directly start a new game.

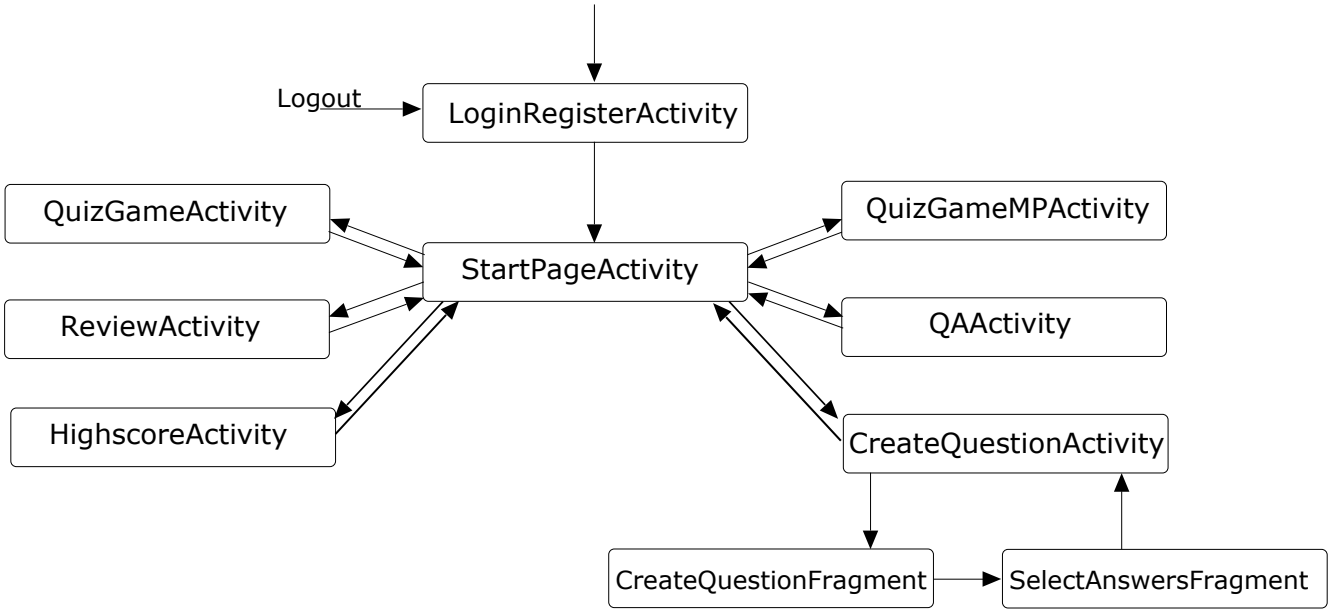
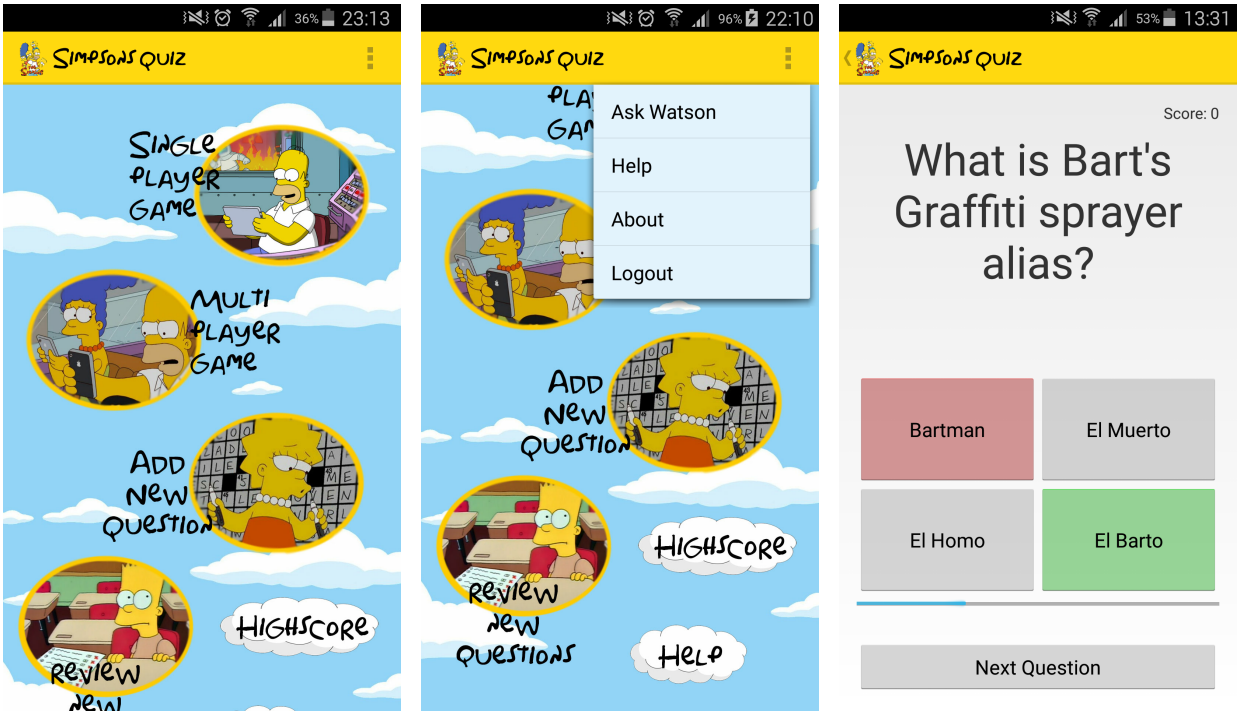
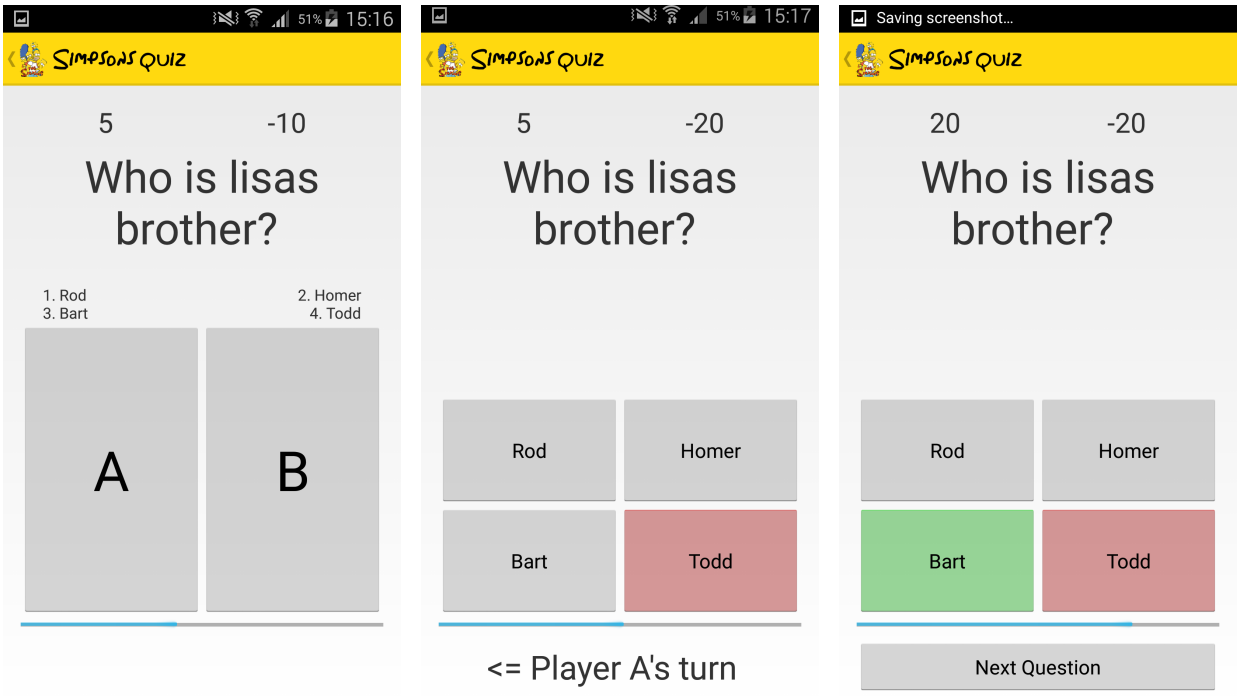


Figure 4.1.: The components of the Android application



(a) The Startpage **(b)** The Startpage with Menu **(c)** The single player game

Figure 4.2.: Views of the Android application



(a) Buzzing **(b)** Answering **(c)** Second Chance

Figure 4.3.: The stages of a multiplayer game

QuizGameMPActivity

The QuizGameMPActivity starts a two player quiz game containing 9 questions of ascending difficulty. The rules of the multi player quiz game are explained in Section 3.2. Both users play on a single mobile device. The screen shows the score of each user and displays who has a turn. The users are labeled as Player A and Player B. As explained in the rules a question is answered in several stages as displayed in Figure 4.3.
After 9 questions the user with the higher score wins. The users can choose to return to the Startpage or to start a new game.

CreateQuestionActivity

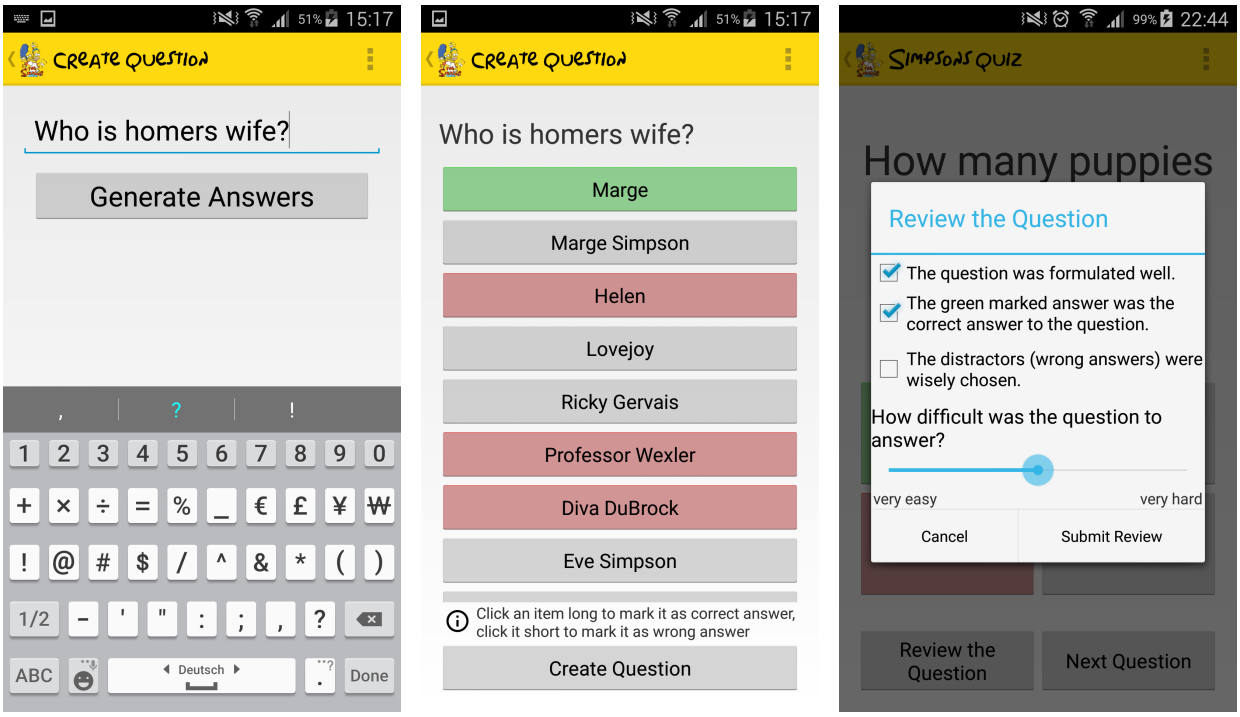
The CreateQuestionActivity enables the user to enter a Simpsons-related question and automatically generate predefined answers, from which one correct answer and three distractors can be chosen. The Activity contains two Fragments, which are displayed in Figure 4.4):

CreateQuestionFragment
This Fragment contains a TextView, where the question can be entered (cf. Figure 4.4a). After entering a question and clicking the "Generate Answers" Button, the answers are generated via the QuizPipeline (cf. section 2.1) and the next Fragment is loaded:

SelectAnswersFragment
This Fragment shows several generated answers (cf. Figure 4.4b). Via a long click on one answer, the answer can be selected as correct answer, a short click on one answer selectes the answer as distractor. An info label reminds the user, that 1 correct answer and 3 distractors have to be chosen to succesfully create a question. The user also has the possibility to add extra answers. After the question has been created and submitted, the user can choose to return to the Startpage or create a new question.

QAActivity

The QAActivity (cf. Figure 4.5a) contains a TextView where the user can enter Simpsons-related questions. After clicking the "Ask Watson" Button, an answer to the question is generated via the QuizPipeline (cf. Section 2.1). The answer is



(a) Enter the question **(b)** The possible answers **(c)** The review

Figure 4.4.: Question Creation and Review

enriched with the sentence from the corpus (cf. section 2.1.1), from which the answer was extracted. If possible an image illustrating the answer is loaded via the ImageFinder (see chapter 8). To display the answer-score a detail-mode can be chosen from the menu. If the detail-mode is active, the score and a web-link to the page from the corpus are added to the answer. A "Next Answer" Button enables the user to display more generated answers with descending scores.

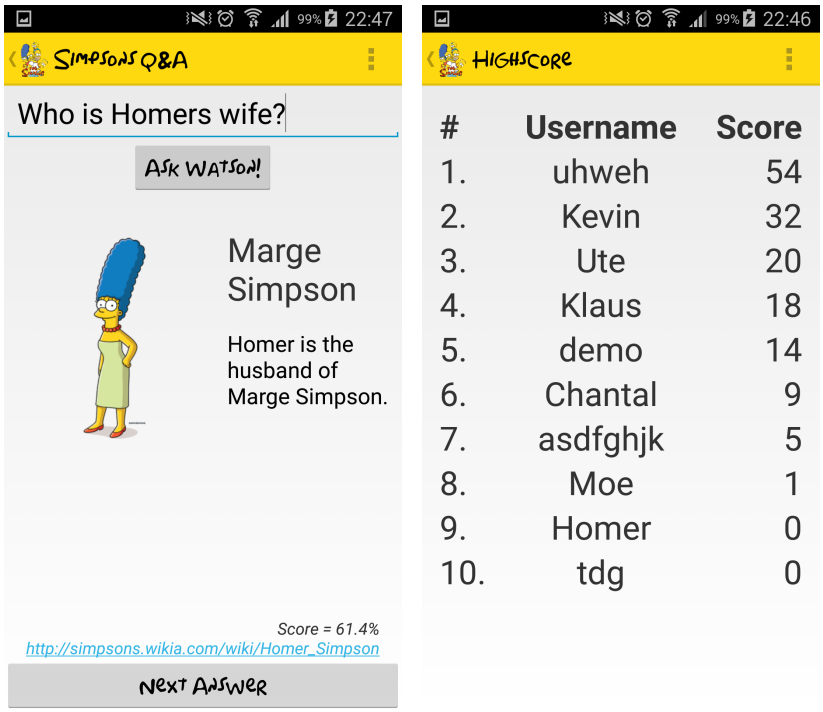
HighscoreActivity

The HighscoreActivity (cf. Figure 4.5b) shows the 10 users with the highest score according to Section 3.4.

4.1.3 Design Choices

- The ui was designed to be simple and intuitively usable.
- The theme contains Simpsons-style colors like yellow and light blue.
- The quiz game was designed to resemble known quiz games like Quizduell[1].
- The Review and CreateQuestion workflows are each divided in two parts to simplify the usage on a mobile system:
  - **Review:** First the question is displayed as it would be during a quiz game, second the user can review the question based on the given criteria.
  - **CreateQuestion:** First the user can enter the question, in a second step, the user can choose the possible answers for the question using touch optimized input.
- To navigate between views or in menus, the android built in features like the back button can be used.

[1] http://www.quizduell-game.de/



(a) The Question Answering View **(b)** The Highscore

Figure 4.5.: Question Answering View and Highscore View

4.2 Webinterface

The webinterface offers the same functionality as the Android application in a web browser.

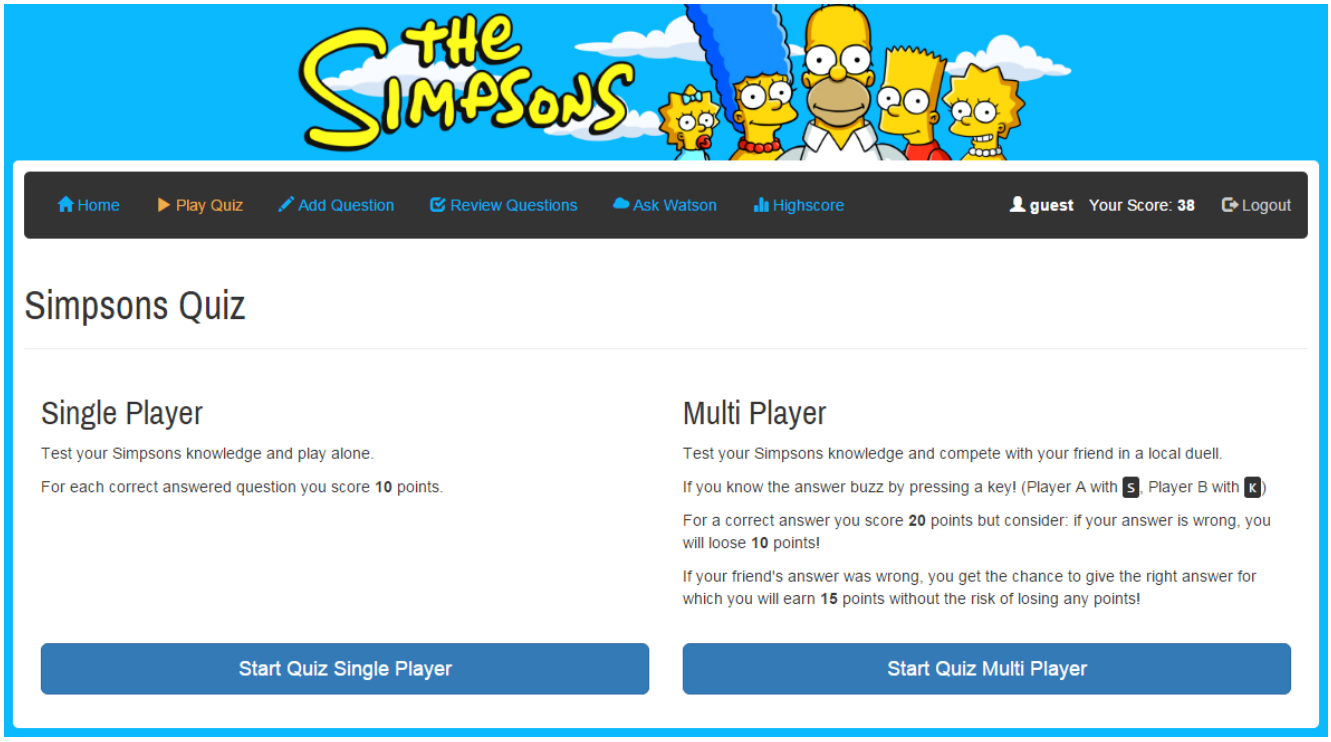
4.2.1 Tools

The webinterface was created using the Play Framework [ZT15], which is an open source web application framework based on Java and Scala. The design was realised with the Bootstrap Framework[2], for client-side functionality the jQuery JavaScript Library[3] was used. The application is based on a webinterface developed by Uli Fahrer [Fah15b].

4.2.2 Components

The webinterface contains several views, that resemble the views of the Android application concerning functionality and design. The user first encounters the Login page that gives a short introduction to the application and offers registration, login, as well as a guest account for testing the service. After signing in, the user can choose on the main dashboard between the different services and read short explanations about them.
Similar to the Android app, the pure QA-service (cf. Figure 4.9a) allows the user to enter natural language questions. The returned answers contain short answers if possible, a sentence long explanation and an accompanying image. The calculated answer score, as well as a reference to the original document are given. The highest scoring answer is shown first to the user but further answers can be revealed by clicking a button.
The Quiz Game can be accessed via an introductory page (cf. Figure 4.6a) that explains its rules and lets the user decide between entering a single or multi player game. Both game modes offer a clear interface that gives information on current player-score, remaining time and correct and wrong answers. Questions are asked one at a time, answered by mouse-click and iterated by clicking a next-button (cf Figure 4.6b). At the end of both game modes a short message with the winning game-score is given. The multi-player mode implements the buzzing mechanic through key presses during a time interval that is illustrated by a progress bar (cf Figure 4.7). Buzzing and answering rules are the same as for the Android App (cf. Section 3.2). The player's turn is illustrated by highlighting "Player A" or "Player B" and displaying a message (cf Figure 4.8).

[2] http://getbootstrap.com/
[3] https://jquery.com/



(a) The Quiz Startpage

(b) A Single Player Quiz Game

Figure 4.6.: Quiz Startpage and Single Player Game

Similar to the Android app, the webinterface implements the concept of user-generated quiz questions, which are created and then reviewed to improve content quality. The view that allows the users to create their own quiz questions starts with an input field for the question and shows possible answers after the “Ask”-button is pressed (cf. Figure 4.10a). After this, the user can select their desired answer options by button-press. The selection is then highlighted and can be submitted when it is complete (cf Figure 4.10b). The review page (cf. Figure 4.9b) shows the examined question and answer options again and lets the user decide on given criteria (cf. Section 3.3.2) or skip to another question. Review and creation are rewarded by points (cf. Section 3.4) and a highscore table can be accessed. Moreover, the individual user score is shown next to the user name in the menu-header on every page.

4.2.3 Design Choices

The views of the webinterface were designed according to the design of the Android application with respect to the possibilities of a bigger screen and the controls with keyboard and mouse. The same header and style of menu is shown on every page for coherence of style. The menu is accessible from every page but kept unobtrusive to ensure usability.

Figure 4.7.: A Multi Player Quiz Game

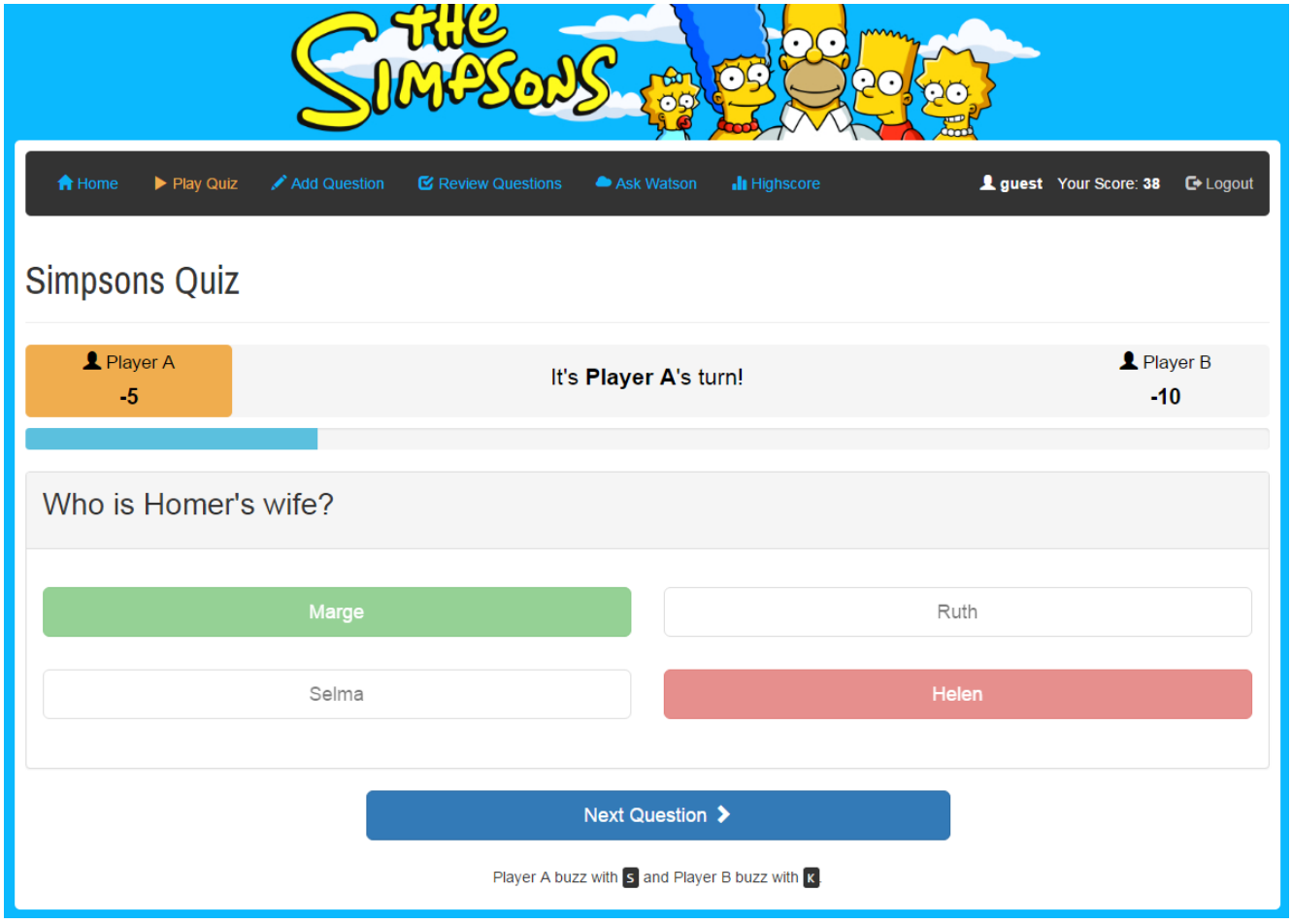
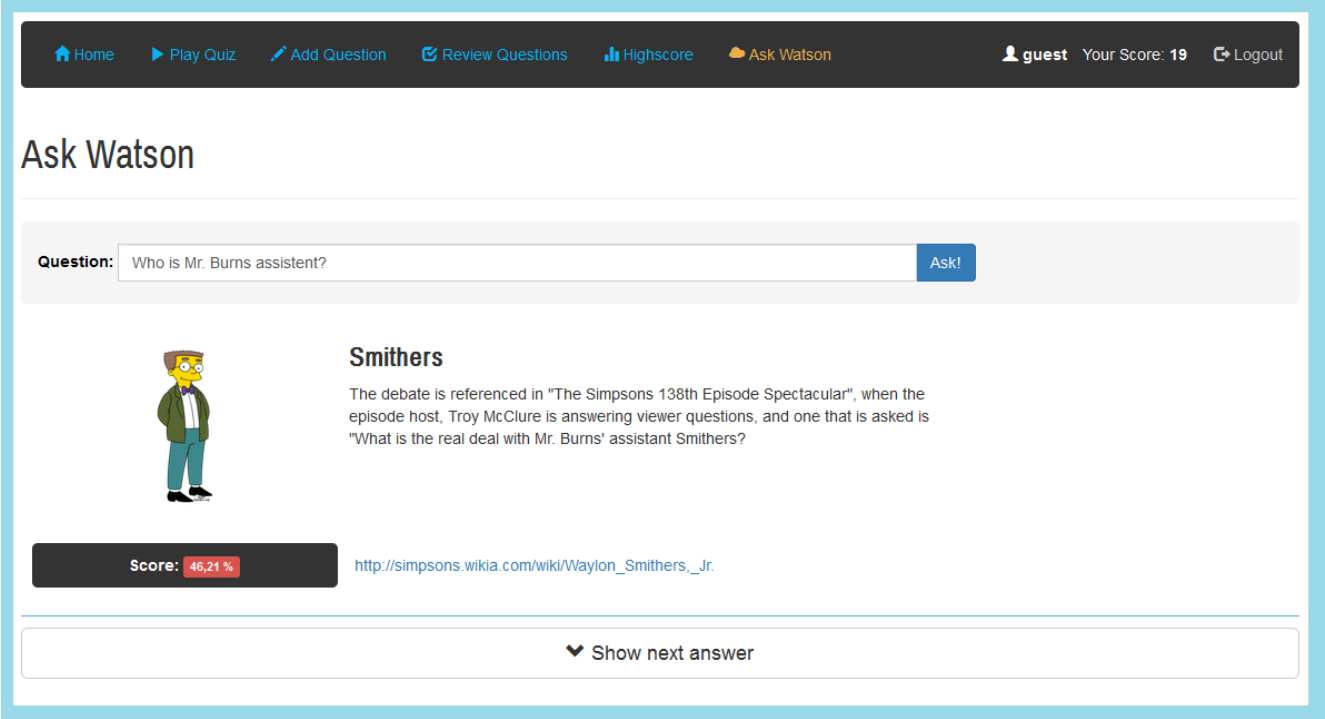
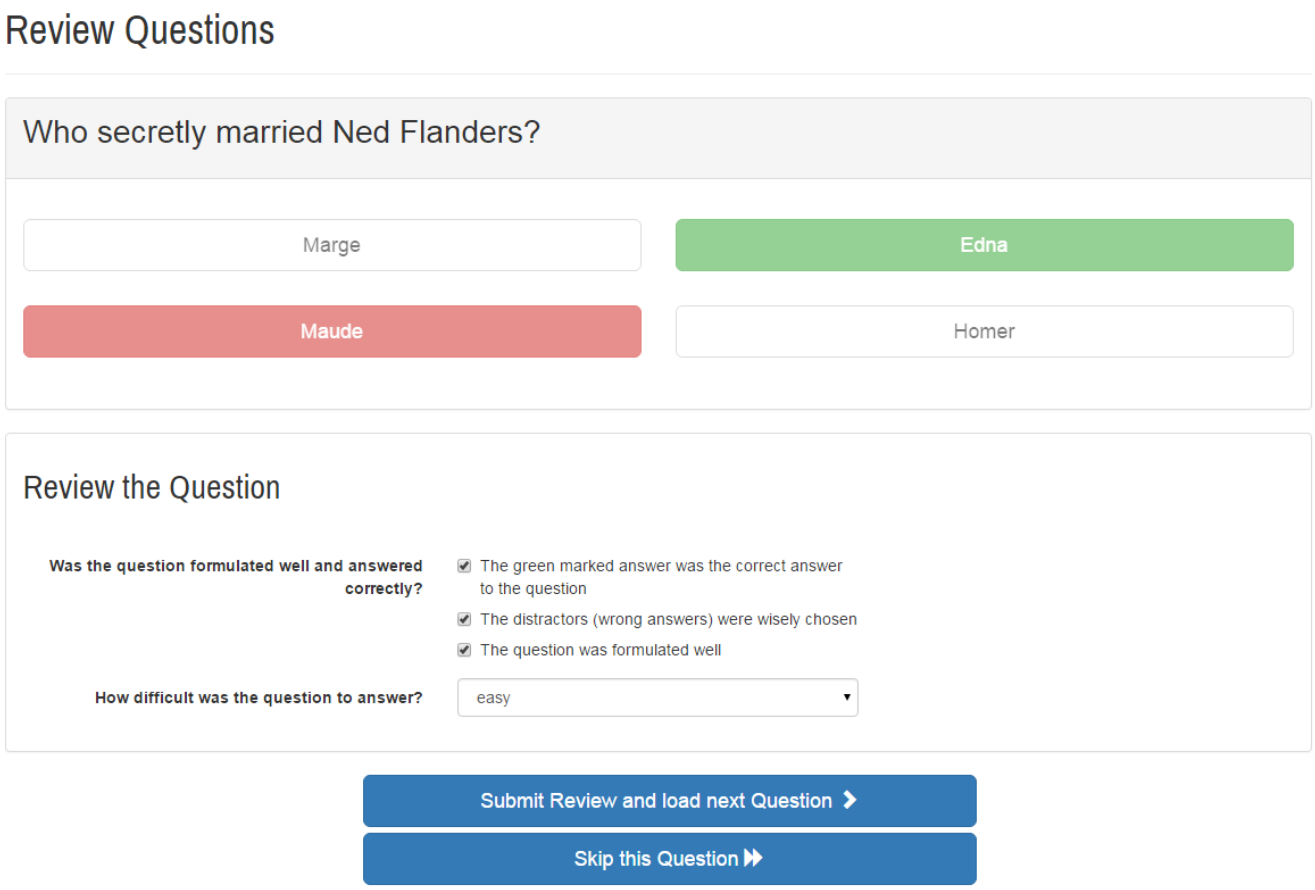


Figure 4.8.: Player B gave the wrong answer and Player A was right

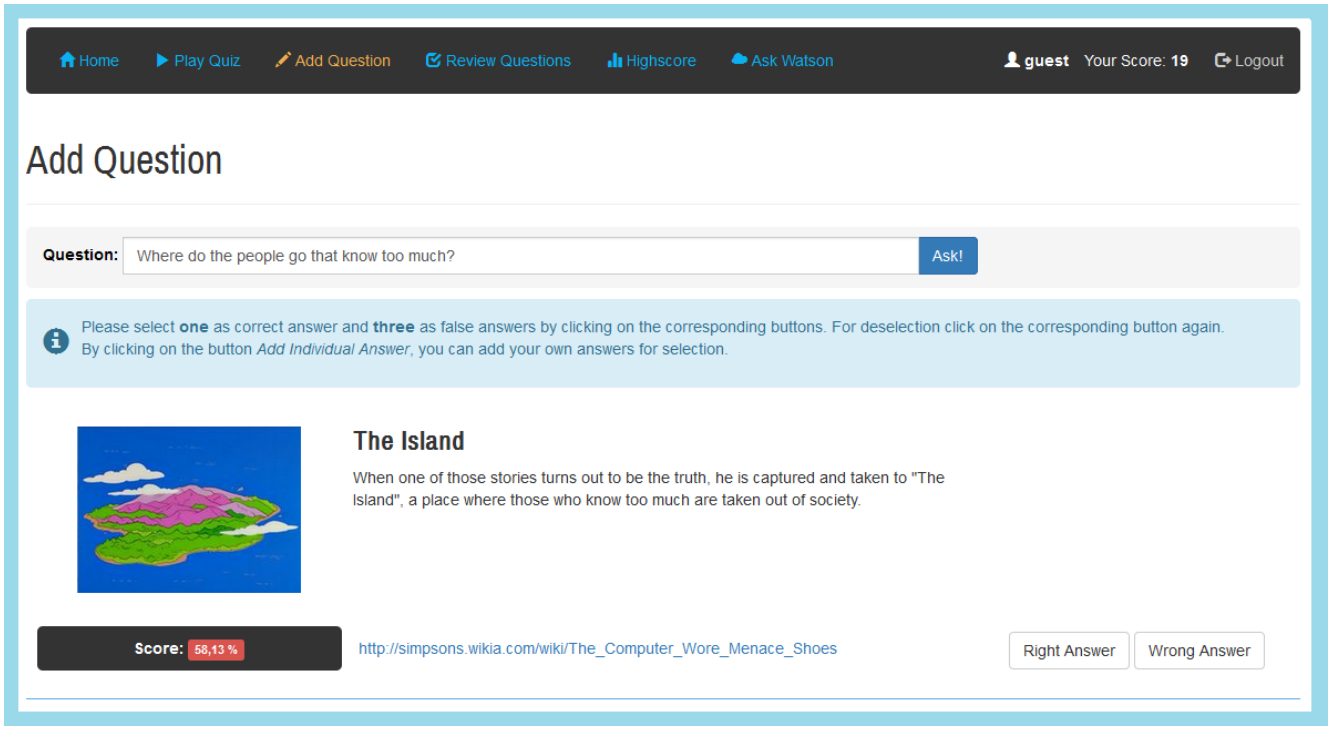


(a) The Question Answering Page

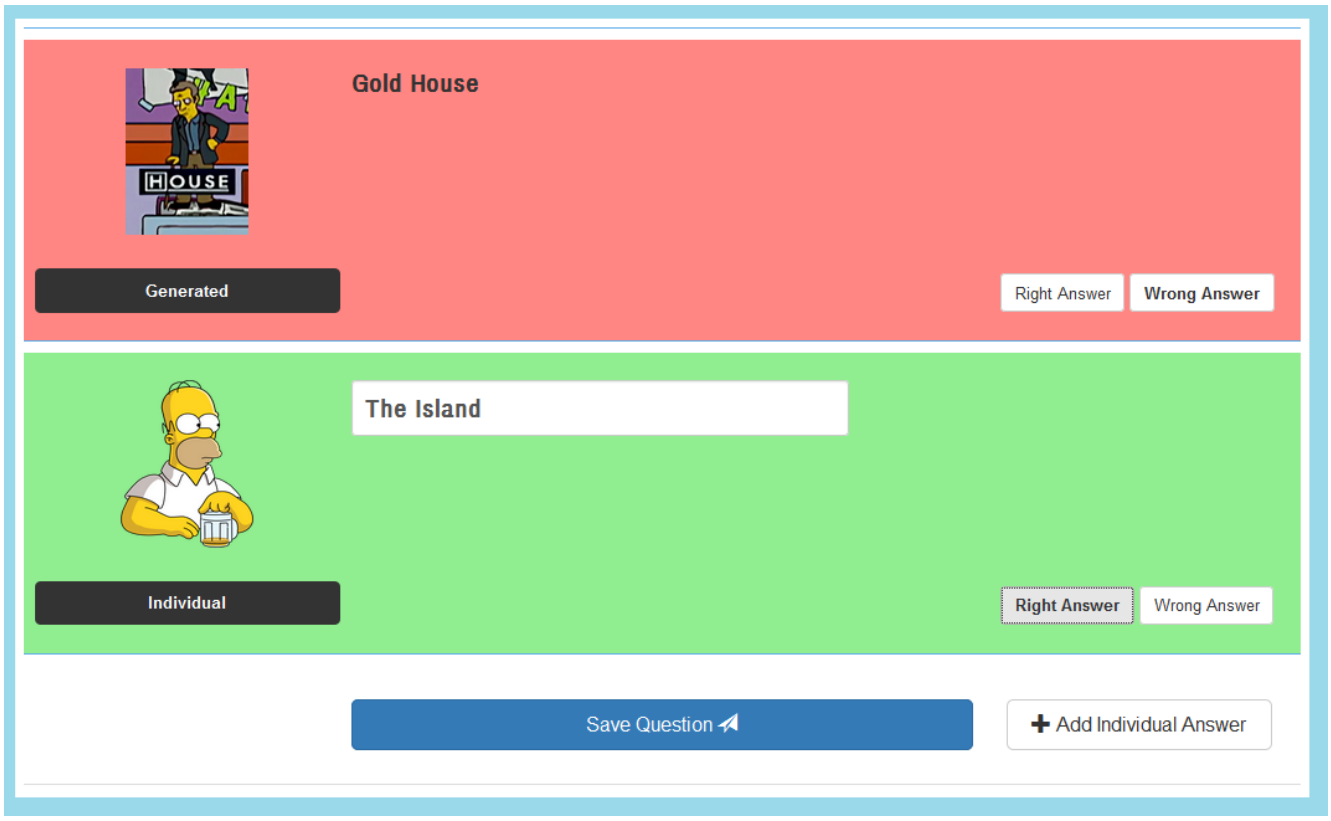


(b) The Review Page

Figure 4.9.: Question Answering Page and Review Page



(a) The Question Creation shows suggested answers.



(b) The Question Creation shows selected answers to the quiz question.

Figure 4.10.: Question Creation

5 NLP Server

For Natural Language Processing our system uses an additional server component. The HTTP server implementation is based on *Stanford CoreNLP XML Server* [Loh13] which offers the functionality of the *Stanford CoreNLP Toolkit* [MSB+14] as HTTP-XML-Server. For our purposes we added and extended the JSON-Outputter (from a newer version of Stanford CoreNLP) that allows to respond the annotated text in JSON format to the requesting client.

5.1 Language Processing

Our instance of this server is configured to use the following annotators of the Stanford CoreNLP Toolkit [MSB+14] in the listed order:

- **tokenize**: Tokenization of the given input text.
- **sspilt**: Splits a sequence of tokens into sentences.
- **pos**: Labels tokens with their Part-of-Speech (POS) tag.
- **lemma**: Generates the word lemmas for all tokens in the corpus.
- **ner**: Recognizes named (PERSON, LOCATION, ORGANIZATION, MISC), numerical (MONEY, NUMBER, ORDINAL, PERCENT), and temporal (DATE, TIME, DURATION, SET) entities. This Named Entity Regognition (NER) is based on models provided by Stanford CoreNLP.
- **regexner**: Implements a simple, rule-based NER over token sequences using Java regular expressions. We used this to incorporate Simpsons-Corpus specific named entities.

More information about these annotators can be found at the Stanford CoreNLP website. [1]

[1] `http://nlp.stanford.edu/software/corenlp.shtml`

6 Quiz Backend

For managing Simpsons Quiz related data like users, questions and reviews a RESTful Web-Backend was implemented. The Web-Backend mainly acts as intermediary between the database and our Quiz applications. It was programmed in PHP and uses *Slim*, a micro framework for PHP [Sli15]. For communication with our applications the Web-Backend offers a REST API which is used by our QuizBackend Java-Client.

6.1 Database

The users, questions and reviews for the Simpsons Quiz are stored in a MySQL-Database. This section gives a short overview about the database structure.
The table *quizquestions* (see Table 6.1) contains the user created questions for the quiz. If an user created or modified a question, this action is saved as status in *user_questions* (see Table 6.2). When users review a quiz question, the whole review data is filed in the table *question_review* (see Table 6.3). In addition to the user credentials, a quiz score that indicates the user participation in reviewing and creating questions is saved in the table *users* (see Table 6.4).

6.2 REST API

The Web-Backend offers different functionality over its REST API which are listed in Table 6.5.

HTTP-Requests to the Web-Backend have to be send with corresponding HTTP-Method and Content-Type `application/x-www-form-urlencoded`, responses are sent as JSON-Objects with Content-Type `application/json` to the client.

For security reasons every request must contain the secret APP-Key. With the exception of register and login, all requests also must contain an user-specific API-Key that authorizes and identifies the requesting user.

Table 6.1.: Database table *quizquestion*

attribute	type	description
id	int	Unique ID (Primary Key)
question	text	The question
correct_answer	text	The correct answer to the question
false_answer1	text	The first distractor
false_answer2	text	The second distractor
false_answer3	text	The third distractor
category	varchar	Categories of the question (optional)
status	int	Status for manual approving
created_at	timestamp	Creation time
modified_at	timestamp	Modification time

Table 6.2.: Database table *user_questions*

attribute	type	description
user_id	int	User ID (Primary Key)
question_id	int	ID of related quizquestion (Primary Key)
status	varchar	Status of relation: {created, modified}
timestamp	timestamp	Modification time

Table 6.3.: Database table *question_review*

attribute	type	description
question_id	int	ID of reviewed quizquestion (Primary Key)
user_id	int	User ID of reviewer (Primary Key)
answer_correct	int	Estimation if correct marked answer is truly the correct answer
answer_distractors	int	Estimation if distractors were wisely chosen
question_formulation	int	Estimation if question was well-formulated
question_difficulty	int	Difficulty rating of quizquestion
timestamp	timestamp	Review time

Table 6.4.: Database table *users*

attribute	type	description
id	int	Unique ID (Primary Key)
username	varchar	Unique username
password_hash	text	Salted hash of user password
role	varchar	User role: {user, approver, editor, admin}
api_key	varchar	Generated key for API authorization
status	tinyint	User status
quiz_score	int	Score for participation (reviewing and adding questions)
created_at	timestamp	Creation time

Table 6.5.: Overview about REST-API

URL	Method	Parameters	Description	Notes
api/register	POST	username, password	User registration	no authentication required
api/login	POST	username, password	User login	no authentication required
api/user	POST	username, password	Editing current user	
api/user	DELETE	-	Deleting current user	
api/user/:id	GET	-	Retrieving user for given id	requires user role *admin*
api/user/score	GET	-	Fetching score for current user	
api/user/score	POST	points	Incrementing score for current user	
api/user/score	DELETE	-	Resetting score for current user	
api/highscores	GET	-	Fetching highscores	
api/questions	POST	question, correctAnswer, falseAnswer1, falseAnswer2, falseAnswer3, category	Creating new question	
api/questions	GET	[filter_unapproved, category, difficulty]	Fetching all questions (optionally filtered by approving/category/difficulty)	
api/questions/user	GET	[filter_unapproved, category, difficulty]	Fetching all questions unseen by user (neither answered nor modified) (optionally filtered by approving/category/difficulty)	
api/questions/:id	GET	-	Fetching single question	
api/questions/edit/:id	POST	question, correctAnswer, falseAnswer1, falseAnswer2, falseAnswer3, category	Updating single question	requires user role *editor*
api/questions/:id	DELETE	-	Deleting single question	requires user role *admin*
api/questions/user-status/:id	GET	-	Fetching user status for question	
api/questions/user-status/:id	POST	-	Updating user status for question	
api/questions/approve/:id	POST	status	Approving the question	requires user role *approver*
api/questions/review	GET	-	Fetching questions for review	
api/questions/review/:id	POST	answerCorrect, answerDistractors, questionFormulation, questionDifficulty	Saving user review for question	
api/questions/review/:id	GET	-	Fetching user review for question	
api/questions/review/:id	DELETE	-	Deleting user review for question	
api/questions/avg-review/:id	GET	-	Fetching average review for question	
api/questions/export	GET	-	Fetching questions with rating information	requires user role *admin*

:id has to be replace with the question id, [...] indicates optional parameters.

6.3 Java-Client

The QuizBackend Java implementation acts as the client component that communicates with the Web-Backend over the REST API. It is used by our Android-App and the Web-Interface.

The BackendCommunicator-Class handles sending requests to and retrieving/parsing responses from the Web-Backend. The QuizBackend-Class holds the API-Key of the current user and offers different functions for reading, saving and deleting quiz related information. Each of these functions creates the corresponding API requests that are send with the help of the BackendCommunicator methods and processes the retrieved responses. For parsing and processing the package *types* contains different classes like `QuizQuestion` and `QuizUser` for creating corresponding objects.

7 Corpus

Our Corpus for the Simpsons Question and Answer Service includes 1680 documents from the SimpsonsWiki [Wik15a] and 1000 question-answer pairs as training data.

7.1 Contents

We have uploaded and added 1680 SimpsonsWiki documents to the Corpus

- 580 Episodes from Season 1 until 27 Season
- 600 Characters
- 500 Locations/Organizations

We have manually annotated the html pages for 35 main characters with sub-headings for paragraphs.

7.2 Training data

We added 1000 question-answer pairs as training data to the corpus. Examples of question-answer pairs that were used for training can be found in Appendix A.

8 Imagefinder

The Imagefinder offers matching pictures for episodes, people, seasons, locations and organizations of the Simpsons to improve the User Interface. We get access to images from Wikisimpsons [Wik15b] via the Mediawiki API[1] and our BackendCommunicator.

8.1 allImagesQuery

For all images on the website we receive the image with URL and title that starts with the handed string by alphabetical order.
`http://simpsonswiki.com/w/api.php?action=query&format=json&list=allimages&ailimit=1&aiprop=url|size&aifrom=Homer_Simpson`

8.2 imagesOnArticleQuery

We pass an article name to get all the images that are on this page.
`http://simpsonswiki.com/w/api.php?action=query&format=json&prop=images&imlimit=20&titles=Bart_the_Genius`

8.3 thumbImageQuery

We pass an image name and an image width to get an URL for a scaled thumb image.
`http://simpsonswiki.com/w/api.php?action=query&format=json&prop=imageinfo&iiprop=url&iiurlwidth=220&titles=File:Homer_Simpson.png`

[1] `http://simpsonswiki.com/w/api`

A Training Data

These question-answer pairs are excerpts from the training data used to train the Watson Private Instance. The answers are each part of a larger paragraph that is also identified during the training process.

Question	Answer
What is Helen Lovejoys maiden name?	Schwartzbaum
What is Carl's Job?	Carl works in sector 7G, along with Homer and Lenny but was later promoted to supervisor after Ted, the previous supervisor, left.
Who knocked Bart off his skateboard with a car?	Mr. Burns
Where is the escalator to nowhere located?	near the Springfield Monorail, 50 Foot Magnifying Glass and the Popsicle Stick Skyscraper
In which district of Springfield is the restaurant Luigi's located?	Little Italy
Where did Hans Moleman live?	His home address was 920 Oak Grove, Springfield.
What is the name of the student that Lisa is chosen to mentor?	Lisa volunteers to help Alex Whitney, a fashion conscious new student, by showing her around the school.
How does Homer lose weight in episode 7 of season 7?	liposuction
What does Becky like to wear?	Becky's normal appearance in the show is her wearing a blue-green dress, long white socks, and blue-green shoes
Which University did Carl attend?	Springfield A&M University
Who has a brief trip to heaven after being hit by Mr. Burns in his car?	Bart
Which product features Homer's face as its logo?	Homer goes on a quest to find out why his likeness is featured as the logo of a Japanese detergent company.
Who is based on Sherlock Holmes' assistant Dr. Watson?	Eliza was based on the fictional detective, Sherlock Holmes, and her assistant Dr. Bartley was based on Sherlock's sidekick, Dr. Watson.
Who is the owner of the Italian restaurant Luigi's?	Luigi Risotto
In which episode has Mark Zuckerberg a guest role?	"Loan-a Lisa" is the second episode of Season 22. It originally aired on on October 3, 2010. Chris Hansen, Mark Zuckerberg and Muhammad Yunus make guest appearances as themselves.
Who does Lisa date in season 8?	Nelson is found guilty and Lisa falls for his rebellious ways and soon develops a crush
Who is a major gossip in Springfield?	Helen is the wife of Reverend Timothy Lovejoy and the mother of Jessica Lovejoy. She is a fair-weathered, judgmental, moralistic person and the typical town gossip
How is the three-eyed fish called that Bart catches in the river?	Blinky

Table A.1.: Question-answer pairs that are used as training data.

Bibliography

[Fah15a] Uli Fahrer. tudarmstadt-lt/jwatson - github. https://github.com/tudarmstadt-lt/jwatson, 2015. [Online; accessed 05-Oct-2015].

[Fah15b] Uli Fahrer. tudarmstadt-lt/watsondemo - github. https://github.com/tudarmstadt-lt/watsondemo, 2015. [Online; accessed 05-Oct-2015].

[FSGK06] Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, and Sokratis Katsikas. Effective identification of source code authors using byte-level information. In *Proceedings of the 28th international conference on Software engineering*, pages 893–896. ACM, 2006.

[Goo15] Google Inc. Download android studio and sdk tools. https://developer.android.com/sdk/index.html, 2015. [Online; accessed 05-Oct-2015].

[IBM15] IBM. Ibm watson. http://www.ibm.com/smarterplanet/us/en/ibmwatson/, 2015. [Online; accessed 05-Oct-2015].

[Loh13] Niels Lohmann. Stanford corenlp xml server - github. https://github.com/nlohmann/StanfordCoreNLPXMLServer, 2013. [Online; accessed 06-Oct-2015].

[LPM+12] 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*, 56(3.4):2:1–2:14, May 2012.

[MSB+14] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[Sli15] Slim. Slim - a micro framework for php. http://www.slimframework.com/, 2015. [Online; accessed 06-Oct-2015].

[Wik15a] Wikia. Simpsons wiki. http://simpsons.wikia.com/wiki/Simpsons_Wiki, 2015. [Online; accessed 05-Oct-2015].

[Wik15b] Wikisimpsons. Wikisimpsons. https://simpsonswiki.com/wiki/Wikisimpsons, 2015. [Online; accessed 05-Oct-2015].

[ZT15] Zengularity and TypeSafe. Play framework - build modern and scalable web apps with scala and java. https://www.playframework.com/, 2015. [Online; accessed 05-Oct-2015].