MASTER'S THESIS 2020

Automatic Generation of Comments

Andy Trinh

ISSN 1650-2884
LU-CS-EX 2020-09

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-09

Automatic Generation of Comments

Andy Trinh

Automatic Generation of Comments

(An Application to Assess Student Assignments)

Andy Trinh
dat13atr@student.lth.se

June 29, 2020

Master's thesis work carried out at Jayway AB.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Johnny Dang, johnny.dang@jayway.com

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Laborious and repetitive tasks in education can be replaced by smart systems, and the necessity has become more clear, as indicated by the research made in this area. However, these systems suffer from many layers of complex tasks, such as data processing and evaluation. We present a complete solution in automatic feedback generation, which provides candidate responses for a given question and answer based on their similarities with previously-seen questions and answers. This is done using the information retrieval models BM25, TFIDF and the Boolean model. With ROUGE metrics, we pick the best response among the candidates with respect to a reference response. Our BM25 model achieved the highest F1-score of 52.11%, 33.93% and 50.19% using the metrics ROUGE-1, ROUGE-2 and ROUGE-L, respectively. Of all the models, the lowest mean reciprocal rank was 0.27. We conclude that the solution architecture and models serves as a good baseline for further improvements.

Keywords: feedback generation, natural language processing, information retrieval system, text generation, education

Acknowledgements

I would like to thank Jayway AB for trusting me in this exciting and rare opportunity of processing and analyzing production data as a part of my Master's thesis. With Johnny Dang's support, management advice, patience and unwavering encouragement, I felt confidence and spirit through thick and thin.

I would like to express my deepest appreciation to Pierre Nugues, who has given me invaluable advice and expert knowledge in the area of natural language processing. With his help and truly impressive commitment, I finished with results far beyond my own expectations. He has my eternal gratitude for his brilliant supervision.

Contents

Chapter 1
Introduction

1.1 Background

Computer-aided education has a history that dates back to the 1950s, taking place in the MIT labs in the form of a computer flight simulator to train pilots. This is the first documented case of using computers to teach (Sharp, 2001). Today, computers are essential for education. The rapid digitization of educational platforms entails extensive possibilities of optimizing tasks that were previously seen as laborious, time-consuming, repetitive and ultimately costly. Embracing new technology, considerable research is put into assisting teachers in making better use of their time to focus on tasks that require more qualitative care and attention.

One of the research areas in this optimization is automatic student assessment: the ability for computers to aid teachers in providing feedback and assessing student output. Feedback is essential in education and plays a central role in students learning. More specifically, feedback of high quality serves as a waypoint for students to assess their current skills and also as a form of formative assessment (Nicol and Macfarlane-Dick, 2006). These fundamentals promote an improved and accelerated learning (Liu et al., 2008, pp 745–783).

With the rise of e-learning platforms and massive online courses, the lack of face-to-face interaction can pose a serious challenge for teachers to give proper feedback to the students, causing adverse effects in the learning process. Studies have shown that a distance learning might have undesirable manifestations, such as feeling of isolation, self-evaluation problems and lack of support from tutors (Galusha, 1997). Kosba et al. (2007) seek to mitigate these disadvantages in form of smart software, which help teachers understand the needs of their students and give appropriate feedback.

In this study, we present an architecture and algorithms to generate feedback with a model akin to recommender systems using content-based filtering as described in Isinkaye et al. (2015). Our goal is to reduce time teachers spend on feedback by generating feedback based on completed exercises submitted by students, who often write similar answers and make the same mistakes, resulting in duplicated feedback. While duplication *per se* is not

necessarily a time-consuming task, completed exercises need to be carefully analyzed to give a fair and correct assessment. Such text analysis requires sophisticated reading comprehension, a non-trivial task for machines.

Starting from the extraction and preprocessing of data, we create a data structure for the acquired data set. Using an engine and techniques common in the areas of *information retrieval* (IR) systems, we populate the index of our engine with the prepared documents. Given a new pair of question and answer, we will use different IR techniques to score their relevance with indexed questions and answers. The question and answer with the highest score will have an associated response by a teacher which will be our candidate feedback. Finally, we will show a use-case in the form of a complete solution with a graphical user interface, and evaluate our system.

1.2 Related work

Generating appropriate response for a given text is a challenging task in natural language processing and, if well-performing, can help automatizing time-consuming, laborious and repetitive tasks.

Starting in 2016, a particular area in reading comprehension has been further catalyzed with the release of *Stanford Question Answering Dataset* (SQuAD) (Rajpurkar et al., 2016), which focuses primarily on finding a correct answer in a passage given a question. Focusing on finding a answer for a given question and a passage, the authors of the paper achieved an F1 score of 51%, a significant improvement of the set baseline of 20%, but lacking in comparison with a human performance of 89%. In 2019, a top scoring model achieved an F1 score of 92% (Lan et al., 2019).

Singh et al. (2013) discuss the pros and cons of traditional approaches to test-case based feedback and peer-reviewed feedback in introductory programming courses. For beginners, test-case base feedback is not ideal and poses a certain problem since error codes typically do not address the erroneous lines of code directly. Peer-reviewed feedback becomes unfeasible with a large class. They introduce a technique for automated assessment using program synthesis technology to match student code submission with a reference code provided by the teacher. While their contributions primarily focus on introductory programming courses using an error model for correction, they address a common denominator, specifically the high workload of teachers. Their system managed to correct 64% of all incorrect programming submissions.

Kosba et al. (2007) presents a novel feedback generation framework utilizing a model for web course management systems. Using a feedback taxonomy defined from problems found in distance courses from previous studies, the framework provides relevant information for teachers with help from tracking data. This, in turn, will allow teachers to gain a deeper insight into the situation of each individual student and thus provide better feedback.

A machine learning approach for feedback generation of essays is presented in Liu et al. (2017), which primarily focuses on linguistic aspects and writing style. They first present a feature model based on Coh-Metrix (Dowell et al., 2015), a system for computing cohesion and coherence metrics for texts. Using three common classifiers: naïve Bayes, support vector machine and decision tree, the authors trained a model which helped students improve their writing qualities.

Britt et al. (2004) present *Sourcer's Apprentice Intelligent Feedback mechanism* (SAIF), a tool to automatically provide feedback for a student's sourcing skills, the ability to write source correctly, for a given essay. Defining sourcing rules for their model to focus on, they used text-matching techniques (latent semantic analysis) to score the essays. Tested on high-school students, SAIF could not only perform as well as human raters but also help students source better relative to a student group without the aid of SAIF.

Ruambo and Nicholaus (2019) surveys different IR models used in modern systems, such as *vector space models* (VSM), Boolean models and probabilistic models. They present a general process flow which includes data indexing, IR model internals and querying handling. Their findings showed conclusively that understanding and adoptions of IR models are important for efficient retrieval.

Aravind et al. (2019) presents an IR based model for physicians to fetch relevant medical documents when facilitating diagnosis and making medical decisions. They also further optimize queries by introducing a learning to rank layer. Their model was tested on 30 queries, which as a whole yielded relevant results.

While the progress in feedback generation has been significant in the last few decades, the wide variety of technologies might indicate that there are no clear approaches to this matter. The studies done so far focus primarily on either a specific technology, providing domain-specific feedback or data tracking. Our study differentiates itself by being hands-on with a complete solution, starting from data extraction of a teaching platform database and concluding with an evaluation of our models. We also contribute to the general feedback generation repertoire by introducing an IR-based approach for teaching platforms. We first index question-answer-response triples, then we query using questions-answer pair. This is done using different IR models, which are evaluated with respect to a reference set.

1.3 Problem statement

We want to introduce a proof-of-concept model for automatic student assessment in education and its life cycle. This includes (1) analyzing, extracting, cleaning and preprocessing the data, and (2) building models with different scoring techniques and (3) evaluate the models using ROUGE metrics.

We will make an attempt at answering the following problems:

1. How can we generate an acceptable response for an answer given by a student based on past responses to a similar answer and question?

2. How can such responses be evaluated?

Chapter 2
Approach

2.1 The teaching platform

The teaching platform referred in this report is a digital platform for both students and teachers.

- For students, it acts as a hub for learning where students can find study material and exercises submitted by a teacher. The teaching platform offers a variety of tools and exercise formats to help engage students into learning. Students with reading disabilities, can also adjust the text complexity and text formatting of the study material.
- For teachers, it serves as a centralized platform for monitoring of student performances, creation of study material and more. This way of centralizing information allows teachers to save time and have a better overview of the individual needs of every student. Teachers can use the study material included in the platform or create their own. As of 2020, the teaching platform supports most subjects in primary and secondary school.

In this section, we will present a use case on who and how the teaching platform is used and which data is being extracted for the feedback generation.

2.1.1 Context model

This section presents in detail the different terms and agents involved in the teaching platform. The name of these entities will be referenced according to the following list.

Stakeholders

Teachers. Teachers are educators who use the platform to create exercises and give feedback to students. For our system, we will be utilizing exercise questions and written feed-

back given to students who have completed the exercises.

Students. Students complete exercises made by teachers. Exercises are typically formed as written questions. The students are primary and secondary school students. For our system, we will be utilizing the answer written by students to generate feedback.

Data elements

Study material. All study material belongs to a specific textbook published on the platform. While it could prove to be useful for feedback generation, it will not be used in this study.

Questions. In most cases a question is associated with corresponding study material, typically a chapter or subsection of a textbook. The teaching platform supports many different types of questions, such as multiple choice or image clicking.

Answer. The format of each student answer is highly dependent on the question type. We will primarily focus on free-text answers.

Response. When students have submitted their answers, the teacher can choose to leave a feedback in form of a response to the student. The responses virtually always come in free-text.

A diagram of the domain logic is shown in Figure 2.1. Teachers prepare the questions, which are then answered by students who in turn might receive free-text feedback in the form of a response from the teacher. Note the relationship between the questions and answers. For a given question, there are many answers, each written by different students. In turn, each answer has one response given by a teacher.

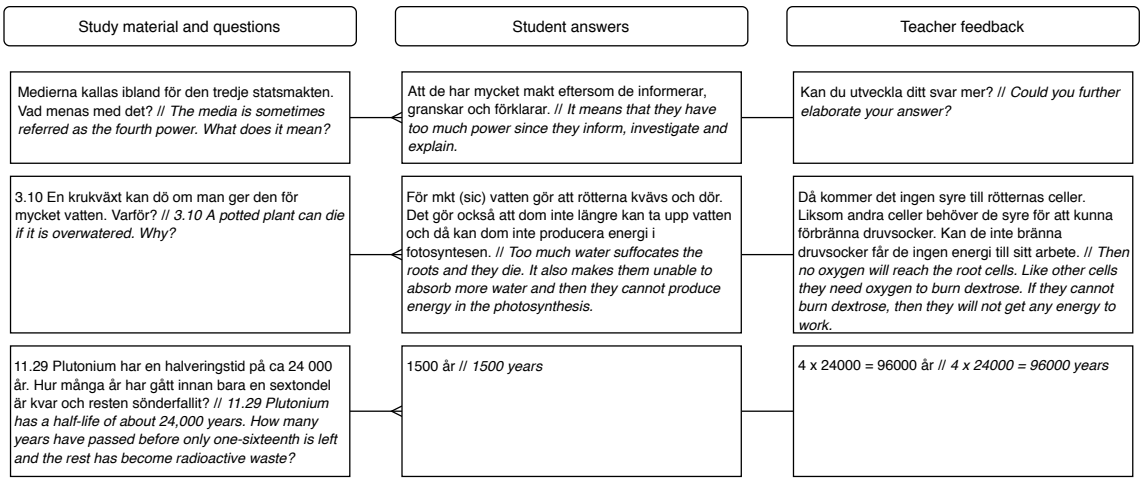


Figure 2.1: A diagram showing the relation between the questions, answers and responses. The diagram shows authentic samples from different subjects.

2.2 Application model and architecture

Using the data extracted from the database of the teaching platform, we will have a set of triples consisting of three data elements: a question, answer and a response. When given a new question-answer pair, the model will match it with the question- and answer value of each triplet in this set using a scoring model. The response value of the top ten scoring triples will be the candidate responses for the new question-answer pair. Among the top ten triples, the teacher can qualitatively choose which response that fits the best.

Figure 2.2 shows a simplified example of this model, where the indexed q, a and r are extracted from the database and the non-indexed is a new question-answer pair. In this case, r_i, r_n and r_p are the top three candidate responses. The main assumption of this approach is that two matching question-answer pairs have a similar, if not the same, response. In contrast to the example shown in Figure 2.2, it is also assumed that in a typical use-case, the set of questions is significantly smaller than the set of answers, meaning that the main differentiating factor is most likely the answer since there are many redundant questions. For preprocessing and fast computations when scoring, we will be using the Apache Lucene search engine library (v8.2).

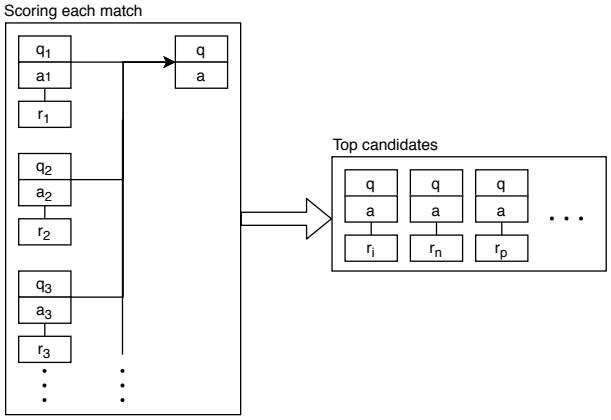


Figure 2.2: A diagram showing the relation between the questions, answers and responses. The diagram shows authentic samples from different subjects.

The architecture comprises:

- Cleansing data extracted from the teaching platform database;
- Indexing the data for efficient scoring in Lucene;
- The graphical user interface.

Section 2.3 presents the data available for extraction. Due to the sizable database of 50GB, we need to store the extracted data in a suitable data- and file structure with support for various data formats. The data will be cleansed, removing any text deemed useless for feedback generation. We will also analyze the data to find useful patterns for the feedback generation.

Using Lucene, the cleansed data will then be preprocessed, used to build an inverted index and generate feedback. To build the inverted index in Lucene, the cleansed data need to be

rebuilt in the form of documents. This process, together with the different scoring models, will be presented in Section 2.4.

At the end of this chapter, we will present a user interface as an example of how our model can be used from a teacher's perspective.

2.3 Data extraction and cleansing

The domain logic of the application is closely tied to the database structure. In object-oriented programming terms, each answer object has a question object and a feedback object. By join-querying the three objects, we conveniently acquire the answer, question and response. Such a triplet will be referred to as an *exercise instance*. Due to the relatively large size of the database, we use the seek method (Winand, 2012, pp 147-148) to query the table row by row.

2.3.1 Data set structure

The questions, answers and responses are primarily stored in HTML. Since we are only concerned about the text, we use the Python markup parsing library BeautifulSoup to extract the text from the HTML code. We chose to store the extracted data in XML due to two primary reasons:

- The length of the texts varies greatly and therefore the output size difficult to estimate. Storing the data in-memory can quickly become unfeasible. There are a handful of libraries in different programming languages that support buffered XML reading. Typically XML is read by constructing a DOM object, which is stored in-memory. As an alternative, we will instead look at *Simple API for XML* (SAX).

- XML is known for being human-readable, in case we want to inspect certain sentences during development.

The data was not stored in its original form after extraction due to problems occurring when parsing the data set. This was partially remedied by storing the data in CDATA sections, as seen in Listing 2.1, allowing XML parsers to interpret the data as purely textual. The alteration of data during extraction means loss of information, but is in return more readable. Given the triple text:

Question: 9.6 Varför ser pupillen i vårt öga svart ut? *9.6 Why does the pupil in our eyes look black?*

Answer: För pupillen absorberar allt synligt ljus som träffar den. Därför studsar inget ljus tillbaka. *Because the pupil absorbs all visible light that hits it. That is why no light is reflected back.*

Comment: Det är inte pupillen som absorberar ljuset utan näthinnan - annars var det rätt. *It is not the pupil that absorbs the light, it is the retina. Otherwise, you are correct.*

we create the data structure seen in Listing 2.1.

```
<exerciseInstance>
  <question id="639463">
     <![CDATA[9.6 Varför ser pupillen i vårt öga svart ut?]]>
  </question>
  <answer id="245518341">
     <![CDATA[För pupillen absorberar allt synligt ljus som träffar
        den. Därför studsar inget ljus tillbaka.]]>
  </answer>
  <response answerId="245518341">
     <![CDATA[Det är inte pupillen som absorberar ljuset utan nä
        thinnan – annars var det rätt]]>
  </response>
</exerciseInstance>
```

Listing 2.1: The listing shows the data structure of the exercise instances in XML format.

Querying and storing data from large tables come with many challenges. While querying, we found bugs which might have affected some output. Other issues involve the structure of the texts. These obstacles will be more deeply discussed in Section 2.3.4.

2.3.2 Evaluating usefulness

Not all text extracted from the database was considered usable for our model. For instance, the majority of the exercise instances were missing responses from the teacher. To automatically assess if an exercise instance was usable, we set up requirements as a part of the cleansing process.

Firstly, a substantial amount of exercise instances contained empty strings. This was especially true for responses. Exercise instances that contained an empty answer or response were omitted as a part of the cleansing process, as they were not considered usable and also exceptional cases.

Secondly, we observed a large number of template responses while exploring the data. This was expected, but by looking at the top ten most frequent duplicate responses in Table 2.1, we can see that some template responses have not been written or manually handled by teachers. These exercise instances are deemed meaningless and are omitted during our cleansing process. As a minimal viable solution for this study, we chose to only remove meaningless template responses among the most frequent ones. Judging from the frequency graph of the top 30 most frequent responses in Figure 2.4, we concluded that most meaningless duplicate responses most likely had a relatively low frequency and would not have a major impact on the performance of our feedback generation.

Finally, some extracted data were not human-readable. These were also removed as a part of the cleansing process. Examples of such will be shown in Section 2.3.4.

2.3.3 Data analysis

The cleansing presented in Section 2.3 concentrated the data set, reducing its size significantly. Due to the data structure of the exercise instance, it could potentially skew the available data for the questions, answers and responses. Relevant frequency cutback for each question, answer and response are presented in Table 2.2.

Table 2.1: The table shows the top ten most frequent template responses found in the data set

Response	Translation	Count
Jämför ditt svar med mallsvaret (din lärare har inte skrivit någon kommentar).	Compare your answer with the suggested answer (your teacher has not written a comment)	559,212
Jämför ditt svar med mallsvaret (frågan har inte rättats av din lärare).	Compare your answer with the suggested answer (the question has not been corrected by your teacher).	303,459
Denna fråga har inte rättats och tas inte med i poängräkningen.	This question has not been corrected and will not be scored.	227,799
Compare your answer with the suggested answer (your teacher has not written a comment).		170746
Denna fråga har inte rättats av din lärare och ger därför inga poäng.	This question has not been reviewed and will not be scored.	144,350
This question has not been reviewed and will not be taken (sic) ignored in the score count.		67,650
This question has not been reviewed and will be ignored in the score count.		55,482
Bra!	Good!	16,244
ok		13,872
Bra	Good	10,708

The first row shows that there has been a significant decrease in the amount of exercise instances that could be used for feedback generation, with approximately 1.1% remaining after the cleansing. The second row displays the number of non-empty responses that existed pre-cleansing and after cleansing. For this data set, approximately 28% of the responses were non-template, as specified in Section 2.3.2. While the number of answers was reduced significantly, the third row tells that the pool of unique questions was not reduced as much. The final two rows show us the average character length of the different data elements before and after cleansing. The average length of answer and responses has been affected by the cleansing: removing empty responses increased the response length from 3.0 to 79.1. If there is a direct correlation between answer length and response length, is however, not known.

Looking at 2.3, which shows the distribution of the cleansed answers with a length or 200 or shorter, we can also see that there is a certain preference of writing an answer with a character length of between 0 and 50.

2.3.4 Quality of data

Skimming the data set revealed a lack of linguistic quality which could potentially hurt the performance. Due to the sporadic nature of the data set and language, we decided that it would be unfeasible to identify and resolve the abnormalities. The following list presents the encountered problems during data extraction and cleansing:

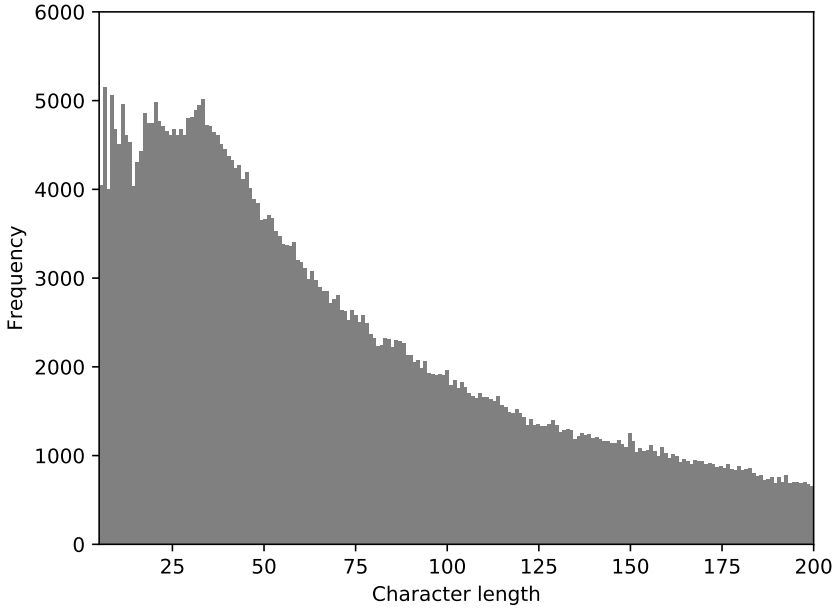


Figure 2.3: The distribution of the answer length between 0 and 200 post-cleansing. The average length of all the answers is estimated to be 153.7.

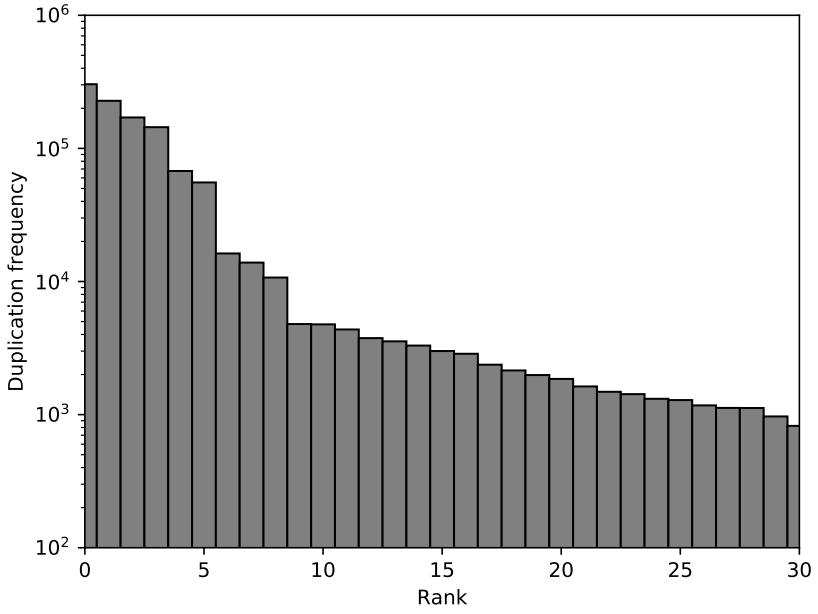


Figure 2.4: The graph shows the 30 most frequent duplicate responses, including the template responses shown in Table 2.1. Note the logarithmic scale in the y-axis.

1. Variety of data. Since the data extracted was primarily in HTML, problems in parsing might occur during parsing, leading to textual artifacts. One example of such problem is `vadjQuery1910766663286886358\_1552293736779?` appearing mid-sentence.

Table 2.2: The table shows the different frequencies of different data elements, namely the question and responses, before and after cleansing. It also shows the average length for answers and responses.

	Pre-cleansing	Post-cleansing
Exercise instances	53,194,420	599,070
Responses	2,127,058	599,070
Unique questions	41,202	11,500
Average answer length	55.9	153.7
Average response length	3.0	79.1

2. Conversational writing style. From observing sample data, we presumed that a large portion of the text was written in brevity and in a conversational level of grammar, sometimes addressing the student directly instead of the student answer. For instance, as shown in Table 2.3, some sentences are incomplete and do not address the problem of the answer.

3. Parsing error. Some words were concatenated as a result of BeautifulSoup's HTML parser.

Examples of each problem type can be found in Table 2.3

Table 2.3: The table shows common textual problems encountered while inspecting the text manually.

Problem type	Example
1	I try to buy jQuery112009065460396115321_1516188811941??? meat
2	Latis ;) Du skulle ju leta upp på skolan... // *Slacker ;) You were supposed to search for it at school...*
3	Choose the five words that you think describe the girl best.happyimpatientrebelliousobedientexciteddisappointedstrangeangrypatientnormal

2.4 Building the index

In Section 1.2, we presented related work which all used a wide variety of different feedback-generation technologies, ranging from artificial intelligence to data tracking technologies. Our model scores similarities between question and answer pairs to derive feedback. To score the similarities between these, we use models common in IR, namely Boolean Model, TF-IDF and BM25.

2.4.1 Preprocessing

To score documents efficiently, Lucene builds an index with preprocessed text. The preprocessing done by Lucene is mainly involves tokenzation, removing stop words, lower casing

and stemming. The class `Analyzer` is a Lucene API which allows us to specify in which language text should be preprocessed. We expect that the majority of the exercise instances are written in Swedish, therefore our model preprocesses according to the Swedish language. Intentionally, this approach might cause a side-effect when generating feedback for documents written in other languages, especially English.

2.4.2 Scoring models

Apache Lucene comes with well-established ranking models. For this study, we will be using Lucene's implementation of a simple Boolean model, TF-IDF and Okapi BM25.

The Boolean model

The Boolean model is based on Boolean logic and set theory and searches documents for the exact words specified in a query. Formally, a query is a Boolean normal form expression consisting of words, naturally supporting AND, OR and NOT operators. It is regarded as design-wise simple, fast during search and intuitive (Lashkari et al., 2009).

As explained by Lashkari et al. (2009), the disadvantages of the Boolean models are:

- Lack of scoring. The model does not formally specify any means of scoring. A document either matches a query or not.
- Lack of control. The results could be too many or too few. The lack of scoring also makes it difficult to regulate and rank the potentially explosive results.

Lucene uses the Boolean models (McCandless et al., 2010, p. 16) to narrow down the documents, which will then be scored using different scoring models. While the Boolean model formally does not provide any scoring schema, Lucene provides a rudimentary scoring function which will only score query terms based on whether they match a document or not. For this study, this score will be the default value of 1. (Apache Software Foundation, 2019a).

Vector space model and TFIDF

The VSM is a model for representing documents and queries as vectors in a high-dimensional space, in which each word correspond a dimension. If a term occurs in a sentence, its corresponding dimension is a non-zero value. Such representation makes it possible to compare two documents, which can be computed using cosine similarity. Given a query in a text retrieval or search system, the aim is to find the most relevant document by measuring the cosine similarity of the query and all the documents in a collection. The values of each term in VSM can be computed in different ways, TFIDF being one of the most popular ones (Manning and Schütze, 1999, pp 539-541).

TFIDF is a family of weighting scheme used to evaluate the importance of a word in relation to a document. It comes in two parts, term frequency (TF) and inverse-document frequency (IDF). Given a document, TF is the frequency of a term in the document. Intuitively, we say that the more frequent a word occurs in a document, the more important it is. However, TF does not capture the specificity of a term and can be seen as trivial if it occurs in many different documents, albeit a high frequency. Jones (1972) argued that while TF is

important, an IDF could further improve the weighing scheme by introducing IDF. Given a collection of documents, IDF gives a higher value for terms that appears in fewer documents, effectively mitigating the disadvantages of TF.

Lucene uses the variant following TFIDF formula to calculate the scoring for a term t_i with respect to a document d_j, with some slight modifications:

$$TFIDF(d_j, t_i) = \sqrt{tf(d_j, t_i)} * (1 + log(\frac{N + 1}{df(t) + 1})) \tag{2.1}$$

where N is the total number of documents in a corpus, $df(t)$ is the number of documents containing t_i and $tf(d_j, t_i)$ is the frequency of t_i in d_j. Lucene's implementation of Eq. 2.1 is slightly different in that it adds extra weights to make it possible to boost the value of certain query terms and documents, making some documents more relevant than others. Lucene also tweaks the normalization factor when computing cosine similarity by boosting the score of shorter documents (Apache Software Foundation, 2019b).

Although TFIDF is considered a classic weighting scheme with good heuristics, it has been criticized for being an ad-hoc solution, lacking a mathematical foundation (Manning and Schütze, 1999, p 544).

BM25

BM25 is an alternative, probabilistic approach which has shown good and consistent performance (Harmon, 1996) (Trotman and Keeler, 2011).

While BM25 is inspired by probabilistic models, it still shares it similarities with TFIDF. Likewise, BM25 consists of two parts: TF and IDF. These are however calculated differently in comparison with their TFIDF counterpart. The weight w of a term t is calculated according to the following formulae:

$$TF_{BM25}(t, d) = \frac{tf(t, d)}{k_1((1 - b) + b\frac{l_d}{avl_d}) + tf(t, d)} \tag{2.2}$$

$$IDF_{BM25}(t) = log(\frac{N - df(t) + 0.5}{df(t) + 0.5}) \tag{2.3}$$

$$w(t, d) = TF_{BM25}(t, d) * IDF_{BM25}(t) \tag{2.4}$$

where k_1 and b are free parameters with default values of 2 and 0.75 respectively, l_d is the document length, avl_d is the average document length of a collection. Looking at Eq. 2.2, assuming that the denominator is strictly larger than 1, it is substantially more punishing for highly frequent terms than its TFIDF counterpart. The TF_{BM25} eventually converges at 1 while TF of the classic TFIDF is missing an asymptotic maximum. Another notable difference is that the scoring function $R(q, d)$ does not utilize any vector space model and the score is calculated directly on the results, whereas TFIDF weighs the individual terms for a vector space representation.

The full document score R is computed by summing all the weights of a given query q:

$$R(q, d) = \sum_{t \in q} w(t, d) \tag{2.5}$$

The standard scorer of Lucene is BM25F, a variant of BM25 that supports structured document properties, such as fields. Since it adds another level of depth into the scoring function $\boldsymbol{R}$, it is slightly more complex but does not divert much from the original formula (Pérez-Iglesias et al., 2009).

BM25 is arguably a more abstract model than TFIDF, drawing inspiration from probability and statistical assumptions. With respect to the scope of this study, we will direct the curious reader to Robertson and Zaragoza (2009) for a comprehensive explanation of its internals.

To summarize, Lucene uses the Boolean model to acquire documents that are relevant to a query. These documents are then scored using one of the following: (1) the Boolean scoring function which will give a score of 1 for each matching term, (2) VSM and TFIDF or (3) BM25.

2.5 Application use-case

This section presents a proof-of-concept application for generating feedback based on a given question and answer. The teacher inputs a question and an answer given by a student to the model that will return ten top scoring responses. If none of the responses are satisfactory, the teacher is given an opportunity to edit a response and add it as a new exercise instance to the index. The model in the following screenshots are scored using BM25 as described in Section 2.4.2.

2.5.1 User input

To generate responses, the user writes a teacher-provided question and a student answer in the text fields, as shown in Figure 2.5, where the translated question is “How big is the force of attraction between you and the Earth?” and the answer “Big.”. The application will then show the top ten scoring responses in a list. Figure 2.6 shows an example such a list, in this case the top four response candidates: (1) “How big?”, (2) “Ten times as big.”, (3) “You need to answer how big the Earth’s force of attraction is in Newton!” and (4) “Ten times your weight.”. Figure 2.8 shows the prompt which allows the user to edit the response before sending the feedback to a student. An edited response, together with the given question and answer, can be added to the index, so that the next time a similar question and answer is inputted, the newly added response will most likely appear.

Figure 2.7 shows the advanced mode, which is enabled by checking the checkbox. This mode will instead show a detailed list that also displays the question and answer, i. e. the exercise instance, that generated a particular response stored in the index.

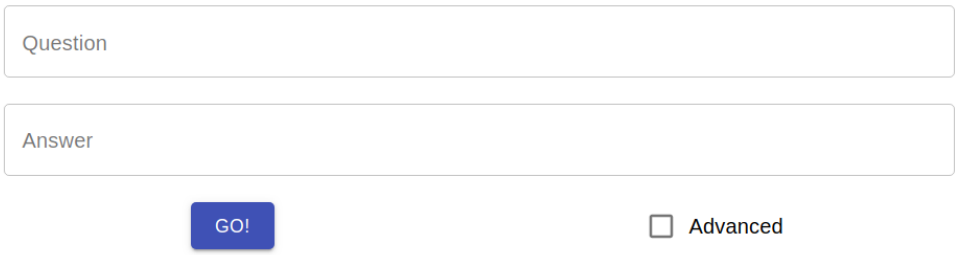


Figure 2.5: The figure shows the initial screen of the application. The user can input a question and an answer in the fields to start generating a response.

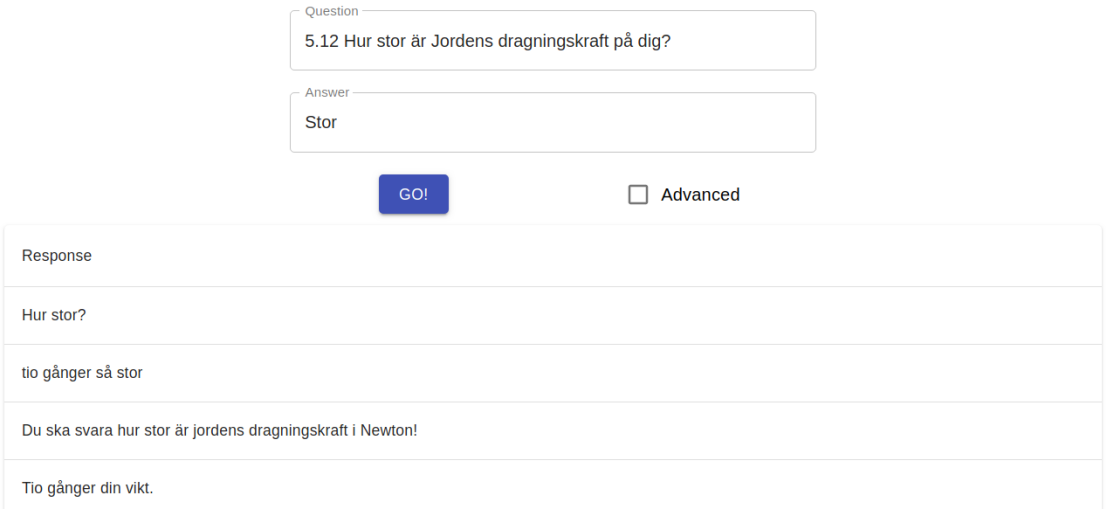


Figure 2.6: A list consisting of the top ten candidates are generated after inputting a question and answer. Translation: (question field) *5.12 How big is the force of attraction between you and the Earth?*, (answer field) *Big.*

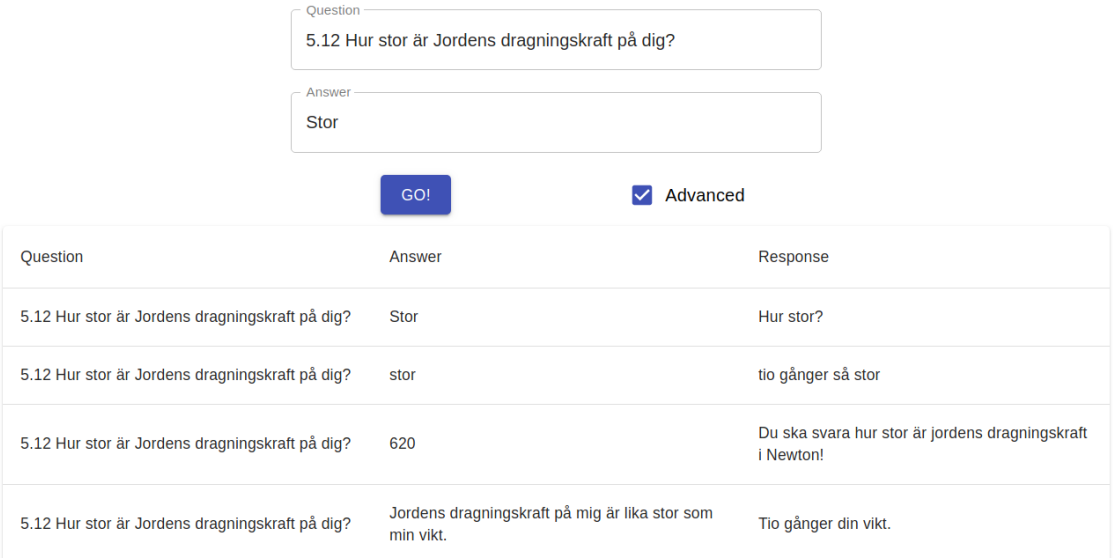


Figure 2.7: Checking the "Advanced" checkbox displays the indexed question-answer pairs that generated the responses in the list.

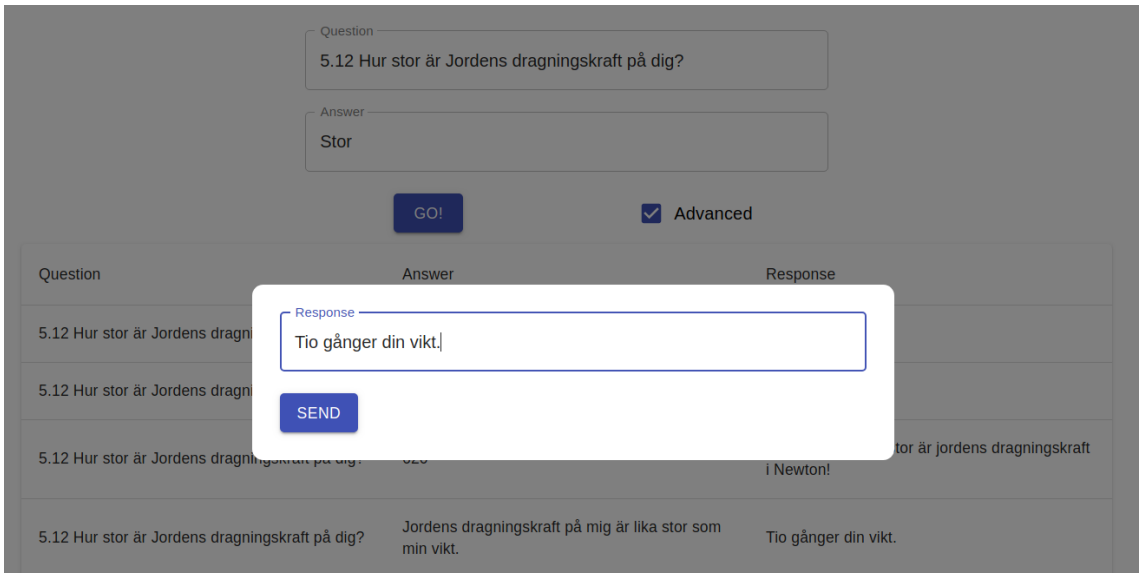


Figure 2.8: Clicking on a response in the list prompts the user to edit the response and then send it to a student. This is an opportunity to adjust responses.

Chapter 3
Evaluation

We took a quantitative approach to evaluate the performance of our model. To evaluate, we split the data into a reference set and a candidate set, which will be used to generate ten candidate responses. The metrics used will be two-fold. We measure the quality of the candidate responses by comparing each one with a ground-truth response from the reference set. This comparison is done using *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE), a set of metrics originally used to measure machine-generated summarizations. The highest scoring candidate response will be considered the best response for a given question-answer pair. We also evaluate the systems ability to rank the top candidate by their *mean reciprocal rank* (MRR), with respect to the ROUGE evaluation. This approach is inspired by Ferrucci et al. (2010), where they narrow down a large set of documents to a few hundred using IR technologies and rank them using a more specialized evaluation method.

3.1 Preparing the data

The indexing done so far has been populated using the whole data set available. To evaluate the different scoring models, a technique similar to k-fold cross-validation is used. The cleansed data set will be split into two parts: a reference set and a candidate set. The candidate set will be indexed and act as our model. On the other hand, question-answer pairs of the reference set will be used to query the model and the responses of this set will act as the ground-truth. Given the question and answer from the reference set, the model will generate ten responses from the candidate set, as described in Chapter 2. Figure 3.1 shows a diagram of the approach, where (q_r, a_r) is a reference question-answer pair, r_r is a reference response and $r_{c,i}$ are candidate responses ranked i by our model.

To reduce the bias, we split the data set into five folds, where one fold is the reference set and the remainder the candidate set. It is assumed that the data is somehow ordered when extracted from the database, so the data set is shuffled before the split. We then change the reference fold, until every fold has been the reference set. This is repeated for all the scoring

models. For each iteration, we clear and re-index with the new candidate set.

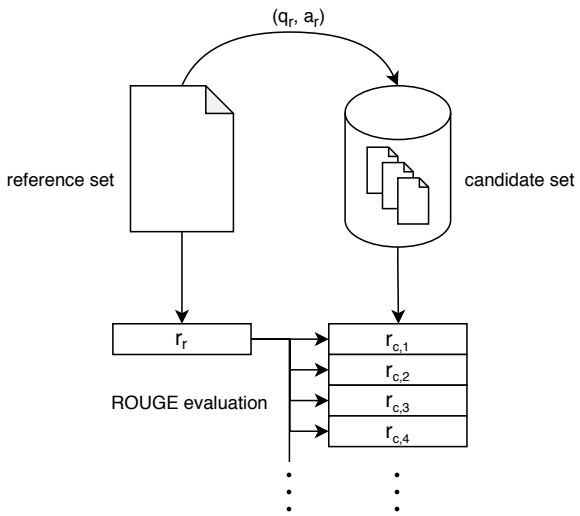


Figure 3.1: The diagram shows the evaluation procedure. The reference set contains ground-truth responses for a given question-answer pair. The candidate set is used to build the index. Each question-answer set from the reference set is fed into the model, which will provide the top ten scoring responses. These responses are then compared with the reference response.

The feedback generation was developed to cater to teachers and does not provide any means to evaluate the candidate responses. To evaluate, the application stored each set of ten candidate responses in folders, one for each set and labeled with an identification number. This identification number is used later to find the corresponding reference response. Using the Python library `rouge`, the candidate responses were evaluated by comparing to the response from the reference set. Since the candidate responses come in unprocessed form, they need to be preprocessed once again before evaluated using ROUGE. This was done using the Python library `nltk`.

3.2 Evaluation methods

To choose the best response among the ten candidate responses, we use ROUGE, which measures the term overlap between a reference response and candidate responses. The more overlap a reference response and a candidate response have, the better the quality of the candidate response is. MRR measures the probability of correctness when looking at the top scoring responses.

3.2.1 ROUGE

In this study, the generated responses will be evaluated using a set of recall-oriented metrics as described in Lin (2004). Namely, we will be using ROUGE-1, ROUGE-2 and ROUGE-L.

ROUGE-N

ROUGE-N is calculated by summing all the overlapping N-grams of the reference and the candidate response and divide it by the total number of n-grams in either the reference set or candidate set. Dividing by the reference set will give yields the recall value, and dividing by the candidate set will yield the precision value. Let $C_{\text{overlap}}(\text{gram}_N)$ be the number of overlapping N-grams in the reference set and the candidate set, and $C_{\text{total}}(\text{gram}_N)$ be the total number of N-grams in either the reference set or the candidate set. Then, the ROUGE-N score is calculated as such:

$$\text{ROUGE-N} = \frac{\sum_{R \in R_r} \sum_{\text{gram}_N \in R} C_{\text{overlap}}(\text{gram}_N)}{\sum_{R \in R_{r|c}} \sum_{\text{gram}_N \in R} C_{\text{total}}(\text{gram}_N)} \tag{3.1}$$

where R_r and R_c is the response set and candidate set, respectively. As a singular score to benchmark each model in ROUGE-N, we use the well-known F1 scoring. It is the harmonic mean of precision and recall and is calculated using the following formula:

$$\text{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.2}$$

ROUGE-L

ROUGE-L scores a candidate and reference response by its *longest common subsequence* (LCS). As described by Cormen et al. (2009), a sequence $Z = [z_1, z_2, ...z_k]$ is a subsequence of another sequence $X = [x_1, x_2, ...x_k]$ if there exists a strictly increasing sequence $[i_1, i_2, ...i_k]$ of indices of X such that for all $j = 1, 2, ...k$, we have $x_{i_j} = z_j$. In this case, the LCS is the longest subsequence of the reference and candidate response with respect to their terms. We say that the longer the LCS is, the more similar the responses are.

ROUGE-L precision score is calculated by dividing the LCS between a reference and a candidate response with either the length of the reference or candidate response. Dividing the LCS by the reference response length yields the recall, and dividing it by the candidate length yields the precision.

While the paper by (Lin, 2004) specifies a weighted F-measure, also known as F_β, we will be using the traditional formula as shown in Equation 3.2, giving equal weight to precision and recall.

ROUGE-L is, in essence, a unigram comparison that rewards sequence. A major disadvantage is that it only considers the longest sequence. If a candidate response is somehow structurally rephrased, e.g. active and passive form in the English language, but has the same meaning as the reference response, LCS will severely punish the candidate response score.

3.2.2 Mean reciprocal rank

MRR is a scoring metric used to evaluate the performance of IR systems, originally used as the main scoring method to evaluate submissions in automatic question-answer workshops held by *Text Retrieval Conference* (TREC) (Voorhees and Tice, 2000). At TREC-8, MRR was used to score a submissions ability to find the correct answer in a corpus given a question.

The submissions returned a list of five candidate answers, and the rank in this context is the rank of the correct answer, which was decided by human judges. The score of a given question was the reciprocal, the multiplicative inverse, of its rank. If the answer does not exist in the returned list, it is given a score of 0. The final score is the mean of all the reciprocal ranks for a given set of questions.

Our evaluation will be using a similar approach to Voorhees and Tice (2000), with one slight modification. We let ROUGE evaluate the correctness of the candidate responses, instead of having a human judge. More specifically, the candidate response with the highest ROUGE F1-score is considered the most correct "answer". If the F1-score is 0.0, meaning no ROUGE-L or ROUGE-N matches, the reciprocal rank will also be 0.0. It has the following formula:

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{3.3}$$

where Q is the set of all the question-answer pairs from the reference set and $rank_i$ is the rank of the highest-scoring candidate response, with respect to the reference response.

3.3 Results

All the scoring models performed competitively when evaluated using ROUGE metrics and MRR, differing at most a few percentages in their precision and recall. Tables 3.1, 3.2 and 3.3 are evaluated using ROUGE-1, ROUGE-2 and ROUGE-L, respectively. The tables show the intermediate results for each fold and the total average of the intermediate results, which we will use as the final score for each model. The highest final score of the scoring models is made bold. The presented results show the precision (prec.) recall (rec.), the F-score (f) and MRR (mrr).

It is clear that ROUGE-2 is a more challenging metric for the scoring models, which all scored substantially lower than the sister metrics as seen in Table 3.2. While performing only marginally better, the BM25 seems to have an edge over the other scoring models, achieving at least two top-scores for all the ROUGE metrics. BM25 scored the highest f-score among all the evaluation methods and scoring models, formally making it the best scoring model.

Table 3.1: The table shows the results of each fold and the total score evaluated using ROUGE-1.

	BM25				**TFIDF**				**Boolean**			
fold	prec.	rec.	f	mrr	prec.	rec.	f	mrr	prec.	rec.	f	mrr
#0	54.26	53.66	51.39	0.36	54.64	52.81	51.14	0.36	51.0	53.85	49.32	0.36
#1	54.23	53.64	51.3	0.36	54.65	52.84	51.11	0.36	51.0	53.75	49.25	0.36
#2	58.68	56.61	55.07	0.38	58.53	54.43	53.69	0.36	55.28	57.78	53.64	0.39
#3	54.35	53.71	51.4	0.36	54.77	52.96	51.23	0.36	51.03	53.81	49.31	0.36
#4	54.29	53.7	51.4	0.36	54.8	52.98	51.29	0.36	51.12	53.98	49.45	0.36
total	55.16	54.26	**52.11**	**0.37**	**55.48**	53.2	51.69	0.36	51.88	**54.63**	50.19	**0.37**

Table 3.2: The table shows the results of each fold and the total score evaluated using ROUGE-2.

	BM25				**TFIDF**				**Boolean**			
fold	prec.	rec.	f	mrr	prec.	rec.	f	mrr	prec.	rec.	f	mrr
#0	34.26	34.39	33.47	0.24	33.87	33.67	32.9	0.24	33.45	34.36	32.96	0.24
#1	34.35	34.38	33.5	0.24	33.96	33.67	32.94	0.23	33.51	34.35	32.97	0.24
#2	36.94	36.23	35.58	0.26	35.71	34.07	33.75	0.23	36.7	37.43	36.12	0.27
#3	34.34	34.36	33.47	0.24	33.99	33.7	32.96	0.23	33.46	34.3	32.93	0.24
#4	34.44	34.51	33.61	0.24	34.06	33.83	33.07	0.24	33.63	34.52	33.13	0.24
total	**34.87**	34.77	**33.93**	0.24	34.32	33.79	33.12	0.23	34.15	**35.0**	33.62	**0.25**

Table 3.3: The table shows the results of each fold and the total score evaluated using ROUGE-L.

	BM25				**TFIDF**				**Boolean**			
fold	prec.	rec.	f	mrr	prec.	rec.	f	mrr	prec.	rec.	f	mrr
#0	52.95	53.3	49.49	0.36	53.17	52.47	49.23	0.36	49.99	53.35	47.38	0.36
#1	52.83	53.28	49.37	0.36	53.15	52.5	49.18	0.35	49.96	53.25	47.3	0.36
#2	57.22	56.26	53.13	0.38	56.93	54.14	51.67	0.36	54.25	57.28	51.77	0.39
#3	53.02	53.35	49.47	0.36	53.33	52.61	49.3	0.35	50.02	53.31	47.37	0.36
#4	52.93	53.33	49.48	0.36	53.32	52.65	49.37	0.36	50.11	53.5	47.53	0.36
total	53.79	53.91	**50.19**	**0.37**	**53.98**	52.87	49.75	0.36	50.87	**54.14**	48.27	**0.37**

3.4 Discussion

The data analysis presented in Chapter 2 gave us some insight into the writing behavior of the teaching platform users, which is discussed in Section 3.4.1. In Section 3.4.2, we discuss the outcome of the evaluation of each model in Section 3.3.

3.4.1 The data set

As expected, the data extraction and preprocessing for our application brought many challenges. We partially address this by putting a minimum requirement by defining "usefulness" for the data extracted as a part of the cleansing process. The data cleansing also allowed us to gain some insight into the behavior of students and teachers on the teaching platform.

The statistics tell us that teachers relatively seldom leave feedback to students. In our data analysis, we found that only 1.1% remained after removing all templates and empty responses. This scarcity of useful responses is reasonable since, for instance, many questions could be too simple or are automatically corrected, leaving no opportunity or necessity for feedback.

Among the question-answer pairs that had a useful response, we saw that many responses are in fact, copies of each other. This could imply that the mistakes the students make are also similar. One of the goals of our proposed model is to address this, by helping teachers to recognize and identify the common mistakes made by students, and also give an opportunity to improve our model by allowing the teacher to update the model index.

As for preprocessing, we have only done it at a bare-minimum level. Integrating spell checking, language detection and further noise removal could help the performance.

3.4.2 Evaluation

We expected BM25 to outperform TFIDF in every metric by a larger margin, contrary to what is presented in the tables of Section 3.3. While BM25 has a slight edge over TFIDF, TFIDF is still a strong contender in evaluating the similarity or relevance of a query with respect to indexed documents. Granted the simplicity of the Boolean scoring, it performed surprisingly well. While the margins could be considered noise and circumstantial, it shows that the Boolean scoring is still a model to be considered for our endeavors. The Boolean scoring showed poor precision, but high recall. This is perhaps expected due to the generous nature of the Boolean scoring, i.e. all matching terms qualify.

3.5 Conclusion

In Chapter 1, we presented the following questions as a part of our problem statement:

1. How can we generate an acceptable response for an answer given by a student based on past responses to a similar answer and question?

2. How can such responses be evaluated?

As a solution for the first question: in summary, we have designed and built an architecture that generates feedback based on previously given feedback. We began by cleansing the extracted data, only keeping the data deemed useful. This process also allowed us to have a deeper look into the writing behavior of the users of the teaching platform, such as a higher tendency of writing answers with an approximate character length of 50 and high response duplication among teachers. We then used Lucene to efficiently generate ten candidate responses by comparing the similarity of a newly given question-answer pair with already-seen question-answer pairs.

As a possible solution for the second question, we used different scoring models and metrics for a quantitative and nuanced evaluation. The ROUGE metrics evaluate the similarity of two sentences by counting the overlapping terms. By scoring the similarity between the candidate responses with a non-indexed reference response, we find the best responses among the candidate. The similarity can be measured by either counting n-grams or the LCS. The top-scoring model, BM25, achieved an F1-score of 52.11%, 33.93% and 50.19% when comparing with reference responses using ROUGE-1, ROGUE-2 and ROUGE-L, respectively. Using ROUGE metrics, we also see that the best candidate can be found among the top four candidate responses, indicated by the worst MRR of 0.27.

3.5.1 Future Work

For future work, we list directions of potential development for further improvements of this study.

- Study material. The study material for each question is available but was not a part of the scope of this study. The study material could aid the scoring models by adding more specificity to the index. For instance, it would be easier to separate subjects by their study material due to the specialized vocabulary used.

- Long answer obscurity. Our model simply queries the question-answer pair, meaning that a long answer could outweigh the question when comparing similarity. For instance, consider the case where the question is short and the answer long. The question, being important for the context, will not have any impact on the scoring. Our model does, however, have the ability to generate a response for a never-before-seen question, which could prove to be useful for a different data set.

- Data extraction. The data extraction was done using a simple school laptop. This posed problems for exploring different approaches to extracting the data and resulted in a loss of information during extraction. A cleaner and lossless extraction would allow better data cleansing.

- Qualitative evaluation. We have taken a quantitative and automating approach in evaluating our model. While fast and efficient, the use of such scoring only goes so far. For instance, in Rajpurkar et al. (2016), the results are compared to human performance. The same kind of issue is also reflected upon the top candidate responses generated by a particular scoring model: ROUGE does not necessarily propose the best candidate among the top candidates. We suggest a qualitative evaluation of the generated responses for a complete benchmark of our models.

Appendix A
Libraries and frameworks

Data processing: BeautifulSoup, containerized MySQL database with Docker

Backend: Java Spring with Maven, Lucene Core 8.2, JAXP

Frontend: React, Material UI

Evaluation: nltk (stop words, stemming), rouge 0.3.2, pandas, numpy, lxml

Bibliography

Apache Software Foundation (2019a). Class booleansimilarity. `https://lucene.apache.org/core/8_2_0/core/org/apache/lucene/search/similarities/BooleanSimilarity.html`. Accessed: 2019-01-28.

Apache Software Foundation (2019b). Class tfidfsimilarity. `https://lucene.apache.org/core/8_2_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html`. Accessed: 2019-01-28.

Aravind, M., Viswanath, S., Mohan, N., Adarsh, R., and Bhaskar, J. (2019). A modified medical information retrieval system. In *2019 IEEE 9th International Conference on Advanced Computing (IACC)*, pages 218–222.

Britt, M. A., Hastings, P., Larson, A., and Perfetti, C. (2004). Using intelligent feedback to improve sourcing and integration in students' essays. *International Journal of Artificial Intelligence in Education*, 14:359–374.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

Dowell, N., Graesser, A., and Cai, Z. (2015). Language and discourse analysis with coh-metrix: Applications from educational material to learning environments at scale. *Journal of Learning Analytics*, 3.

Ferrucci, D. A., Brown, E. W., Chu-Carroll, J., Fan, J. Z., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J. M., Schlaefer, N., and Welty, C. (2010). Building watson: An overview of the deepqa project. *AI Magazine*, 31:59–79.

Galusha, J. M. (1997). Barriers to learning in distance education. *Interpersonal Computing and Technology Journal*, 5(3):6–14.

Harmon, D. K. (1996). *Overview of the Third Text Retrieval Conference (TREC-3)*. DIANE Publishing Company.

Isinkaye, F., Folajimi, Y., and Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261 – 273.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.

Kosba, E., Dimitrova, V., and Boyle, R. (2007). Adaptive feedback generation to support teachers in web-based distance education. *User Modeling and User-Adapted Interaction*, 17(4):379–413.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations.

Lashkari, A. H., Mahdavi, F., and Ghomi, V. (2009). A boolean model in information retrieval for search engines. In *2009 International Conference on Information Management and Engineering*, pages 385–389.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Liu, G.-Z., Spector, J., Merrill, M., Van Merrienboer, J. J. G., Driscoll, M., and Erlbaum, L. (2008). *Handbook of Research on Educational Communications and Technology, 3rd Edition*. Lawrence Erlbaum Associates, USA.

Liu, M., Li, Y., Xu, W., and Liu, L. (2017). Automated essay feedback generation and its impact on revision. *IEEE Transactions on Learning Technologies*, 10(4):502–513.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co., USA.

Nicol, D. J. and Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: a model and seven principles of good feedback practice. *Studies in Higher Education*, 31(2):199–218.

Pérez-Iglesias, J., Pérez-Agüera, J. R., Fresno, V., and Feinstein, Y. Z. (2009). Integrating the probabilistic models bm25/bm25f into lucene.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Ruambo, F. A. and Nicholaus, M. R. (2019). Towards enhancing information retrieval systems: A brief survey of strategies and challenges. In *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 1–8.

Sharp, V. (2001). *Computer Education for Teachers: Integrating Technology Into Classroom Teaching with PowerWeb*. McGraw-Hill Higher Education.

Singh, R., Gulwani, S., and Solar-Lezama, A. (2013). Automated feedback generation for introductory programming assignments. *SIGPLAN Not.*, 48(6):15–26.

Trotman, A. and Keeler, D. (2011). Ad hoc ir: Not much room for improvement. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 1095–1096, New York, NY, USA. Association for Computing Machinery.

Voorhees, E. and Tice, D. (2000). The trec-8 question answering track evaluation. *Proceedings of the 8th Text Retrieval Conference*.

Winand, M. (2012). *SQL Performance Explained: Everything Developers Need to Know about SQL Performance*. M. Winand.

EXAMENSARBETE Automatic Generation of Comments
An Application to Assess Student Assignments
STUDENT Da Yin-Andy Trinh
HANDLEDARE Pierre Nugues (LTH), Johnny Dang (Jayway AB)
EXAMINATOR Jacek Malec (LTH)

Botten som underlättar pappersarbetet - och låter lärare vara lärare.

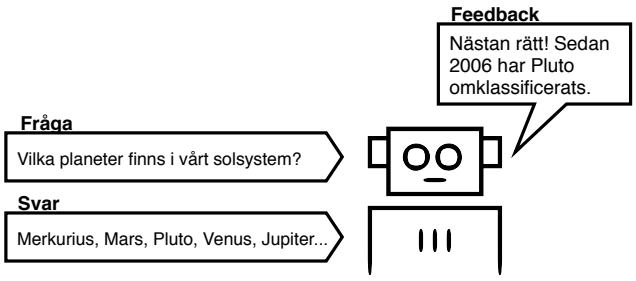
POPULÄRVETENSKAPLIG SAMMANFATTNING **Da Yin-Andy Trinh**

Med inspiration från AI botten Watson, som vann mot mänskliga deltagare i frågesportsprogrammet *Jeopardy!*, har jag utvecklat ett smart system som automatiskt genererar feedback åt elevers inlämningsuppgifter. Grundidén är att ge mer tid åt lärare, så kan de välja att fokusera på annat.

Idag ägnar lärare mycket tid åt repetitiva och tidskonsumerande pappersarbete som hindrar dem från att göra en av samhällets viktigaste uppgifter - *att lära ut*. Detta kan tyckas vara ålderdomligt i en tid där vi samtidigt har tillgång till självkörande bilar, smarta dammsugare och virtuella assistenter. Frågan är då, hur kan smarta system hjälpa till lärandet i skolan?

I mitt arbete har jag utvecklat ett smart system som underlättar för lärare genom att automatiskt generera feedback till elevers inlämningsuppgifter. Läraren matar först in en fråga och ett svar, sedan kommer systemet att returnera en rangordnad lista med bäst lämpad feedback. Rangordningen är baserat på vad läraren förr har gett för feedback till liknande svar och frågor. Då det inte är någon hemlighet att många elever brukar begå liknande misstag, så blir rättandet snabbare då lärare bara behöver godkänna, eller redigera feedback från listan. Om ingen respons tycks vara lämplig kan läraren uppdatera systemet genom att skriva en egen respons. Med tiden blir systemet smartare, och samtidigt så sparar lärare mer tid, en win-win situation.

För att rangordna feedbacken efter relevans utnyttjar systemet statistiska modeller som även används av sökmotorer. Vanligtvis anger man söktermer till ett sökmotor, som därefter returnerar sökresultat i form av en rangordnad lista. I de modellerna jag har utvecklat anger man istället en fråga och ett svar.



Kvalitén på modellerna utvärderades genom att jämföra ordsekvens och vokabulär i den genererad feedbacken med referensfeedback. Detta gjordes på drygt 100 000 frågor och svar, och resultaten visar modellens genererade feedback är relativt lik referensen. Utgångspunkt är att ju mer lika de är, desto bättre är modellen.

Hur systemet faktiskt presterar på produktionsnivå återstår att se. Utvecklingsmässigt finns det många riktingar att gå, som exempelvis att integrera maskininlärningslösningar.