Visualizing Short Text Answers

Dario Bogenreiter (11702132)

July 1, 2022

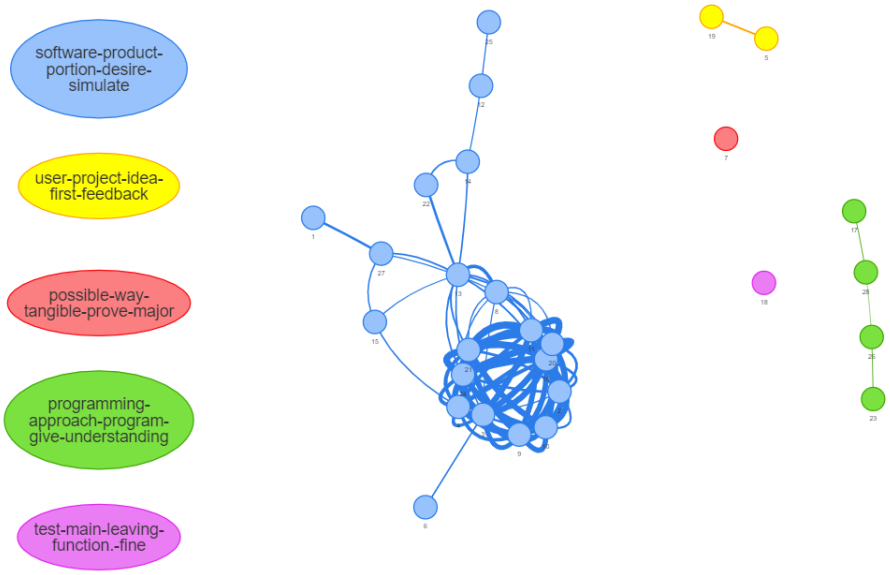

Figure 1: Snapshot from the final application

1 Introduction

Recently, not only due to the COVID-19-pandemic, but also due to the ongoing digitization in education, an increasing number of textual exams and assignments are being completed on the computer. Furthermore, there is greater global access to quality education and a growing number of (private) online courses. As a consequence, more and more textual answers to (exam) questions are digitally available. This ongoing digitization of texts opens up new possibilities for evaluation and analysis. For instance, machines could assist humans in the grading student submissions - which could save both time and money.

The developments mentioned in the previous paragraph also apply to exams or tasks where students get asked to write short text answers, which are the kind of texts that this this paper focuses on. These are, as the name suggests, short answers to open questions that rarely exceed the length of one paragraph. An essential difference to other answers to open questions - assay texts -is also that they usually have a reference answer [1]. Short text answers are highly popular as they allow teachers to train cognitive skills in their students with great effectiveness [1].

When it comes to the digital analysis of these texts, many questions arise as to how this can be carried out - how the differences between texts can be calculated and then visualized. Whether, for example, characters, words or word constructions (such as NGrams) are to be compared. Or how features can be extracted from the data - with word embeddings or bag of words models. And how these differences or similarities of the documents (of the different texts) can then be visualized.

This paper has therefore sets itself the objective of firstly providing an outline of current methods in this area and then secondly designing a prototype for the visualization of short texts.

The paper is structured as follows: section two discusses the main challenges of this project, section 3 deals with different approaches to the issue that this paper deals with. In section 4 a concrete design for this is introduced, whose implementation is then presented in section 5. Finally, section 5 deals with a brief analysis of the final design and section 6 highlights the limitations of this project.

2 Challenges

Based on a literature research and some hand-on experience/experiments with the data and corresponding visualizations, three major challenges were identifies that have to be overcome in order for a visualization to be meaningful and practically applicable in the end.

2.1 Scalability - the Number of Documents

The first difficulty arises from the number of documents (in this case student answers) that are compared with each other. A visualization in the short-question context should work for a range of different numbers of answers - so no matter if 5, 50 or 150 students answered a question, the visualization in the end should give a valuable comparison of all these texts with each other. In some cases the number of documents could be even higher than 150, thus this should also be considered in the visualization. Therefore, a method has to be found that can present the variety and differences of these texts as compactly as possible and yet informatively.

2.2 Choosing the right features

The next hurdle is to choose what the input of the visualization should be and how this input should be represented by graphical objects. Roughly speaking, the input could be: the whole texts, single words, word combinations (like n-grams), single letters, word embeddings and many more. Using only the whole raw text is mostly unsuitable, because a visualization of all texts quickly becomes too messy (especially if you have many texts) and the texts as raw character limits the possible comparison of texts quite a lot. Accordingly the question arises, on which level now the analysis is to take place and in which way the original input is to be transformed. For example, the original texts can be filtered for essential components (e.g. by stopword removal) or unified (e.g. lemmatization, stemming), and so on and so forth. Each step that we decide to do in this regards has an impact on the final product and should hence be considered carefully.

2.3 High dimensionality of features

The last of the three main challenges in this project results from the features chosen in point two. For example, if one decides to consider all words of the documents as such - this can quickly become a scalability problem here (similar to the first point). For example, some visualization techniques can no longer display such a high number of words in a meaningful way - since, for example, texts or geometric elements overlap and are no longer informative for the user.

3 Possible Visualizations techniques

The possible approaches to visualize texts are very numerous and differ based on many factors such as the domain - e.g. scientific articles, Social Media Texts. Patents, Poems - the task to be accomplished - clustering, monitoring, comparison, exploration - the type of data or its representation in the graphic - clouds, line plots, maps, icons, text and so forth [2].This project is about realizing the task of comparing texts with respect to their differences and similarities in the domain of Short Answer (SAQ) - for this purpose, different types of visualizations with different representations, alignments and data (through data manipulation, different types of inputs can be generated from one file) should be tried out.

One of the main ideas behind text visualization is to display texts in a way that avoids showing too much detail about them, as the sheer mass of information can be overwhelming for the user - not showing enough detail can however also be unacceptable, which make text visualization a challenging task [3]. This is one of the most important points that should be taken into account in the final solution.

Broadly speaking, there are three major ways to visualize texts: 1) direct text visuals (e.g. tag cloulds) 2. indirect text visuals (here, quantifiable properties/features of texts are visualized) or 3. hybrid - a combination of the first two possibilities [3]. In this project, only the last two types were considered, because the first type - like Word Clouds - are mostly too confusing for the user and additionally they lack scalability - especially when it comes to comparing the differences between texts. This was not only empirically determined but also in the literature [3].

3.1 Visualization data

As already mentioned at the beginning, it is usually not advantageous to take the documents directly in their raw state as input for the visualizations. Roughly speaking there are two main points to change the input - by processing - to change the texts themselves - and yet a change of the representation - by representing texts e.g. not as a combination of words/letters but as vectors.

3.1.1 Preprocessing

Simply taking the raw text data as input would cause problems in the analysis as well as in the visualization. A few pre-processing solutions are: stopword removal [4] - significantly reduces the number of words in the document corpus and can lead to an increased performance as noise can be avoided (just the relevant data gets analysed), -lemmatization/stemming [5] help to put words that have the same or a similar meaning in the same form - else these words would be treated as different entities, text cleaning, case folding, spell correction

[6] - to show the same words in the same form (e.g. parrot, Parrot, parrrot -> parrot, parrot, parrot).

3.1.2 Representation

Two of the most frequently used kinds of document representation are bag of words (BOW) and term frequncy - inverse document frequency (TF-IDF) models [7]. BOW is one of the simplest methods to create a new representation: simply count the frequency of all the words in the corpora per document (Term Frequency (TF)) [8] - and then use the resulting frequency counts for the different terms per document as a vector. Figures 2 illustrates a short example for this - here we have just filtered the two most important terms in this context - still we get a vector of length 2 for each document which we could use to compare the documents.

The TF-IDF model extends the BOW concept in that it introduces yet another measure of the importance of terms (rather than just considering their frequency in a single document) [9]. This is achieved by calculating the document frequency DF - how often a word occurs in all documents [9]. Terms that occur in a large number of texts are then considered less relevant, since they are assumed to be less suitable for representing the differences between documents [9].

However, this assumption is often inappropriate in the short-answer domain, as some experiments have shown. For example, on the question "Where do C begin to execute?" 95% of the respondents had the word "main" (method or function) in their answer - which was the most crucial word to distinguish it from the other answers (where students answered "test phase" or "root"). TF-IDF would thus have downplayed the importance of the key word to distinguish. Because of this characteristic and the inferior explainability, BOW was applied instead of TF-IDF as the vectorization method for this project.

```
question = 'How many constructors can be created for a class?'
```

student_answer		many	one
just one per classes .	Doc-0	0	1
as many as you want so long as they have different parameters .	Doc-1	1	0
there be no limit to the number of constructors in a classes because like functions , constructors can be overload .	Doc-2	0	0
it depend what type of classes be be define . typically you would have a constructor call for each object .	Doc-3	0	0
the constructor can be overload in that there can be more than one constructor for a classes , each have different parameters .	Doc-4	0	1
one	Doc-5	0	1
any number you want	Doc-6	0	0

Figure 2: Example for a Bag of Word representation

SideSide Note: In order to work comparability between the visualization techniques - all techniques visualize the answers to the question "Where do C ++ programs begin to execute" from a SAQ-dataset by Mohler et al. [10].

3.2 Parallel Coordinates

Parallel Coordinates are a tool with which one can visualize multivariate data [11]. The method maps each data record to a ploryline - whose trace depends on the different features (e.g. Income, life expectancy, Illiteracy) in the dataset [11]. By doing so one can often quickly detect patterns see for example if many data records behave in the same way (for example - if many of the persons that have a high income also have a low life expectancy).

Figure 3 show an example for our research domain. Here the features are the different terms that appear in the different documents. What we can take away from this first graphic is that parallel coordinates are quite irritating if we use them in the traditional way -by mapping one data record to one polyline. This is because of the characteristics of this domain - first of all the features matrices (e.g. doc-term matrix) are usually sparse - many terms only appear in a few documents. Additionally, we have the problem that a features (term frequency) is either 1 (in some rare cases 2 or 3) or 0 leading to a sort of zig-zag line making it hard for the viewer to recognize patterns.

Other disadvantages of this method result from the fact that it is hardly possible to follow a line and see from which components a certain document is composed. Furthermore, the space to display all the important terms (which lead to the distinction of answers) is often not enough and the font often becomes so small that it is hardly readable.

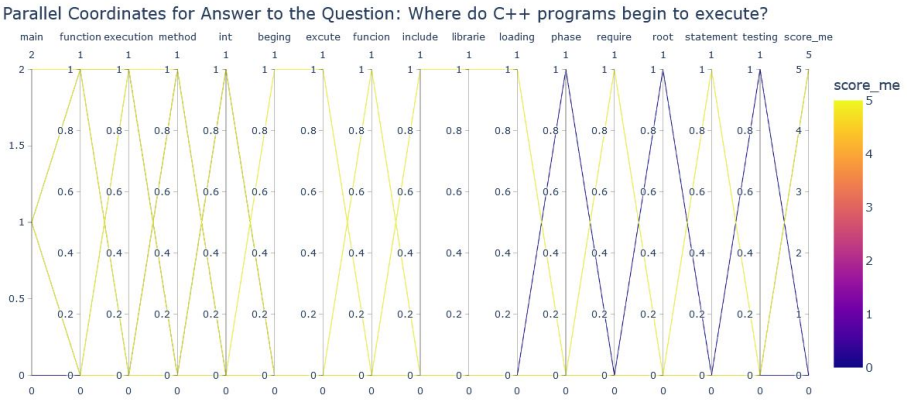

Figure 3: Visualization-Solution: Initial Parallel Coordinates

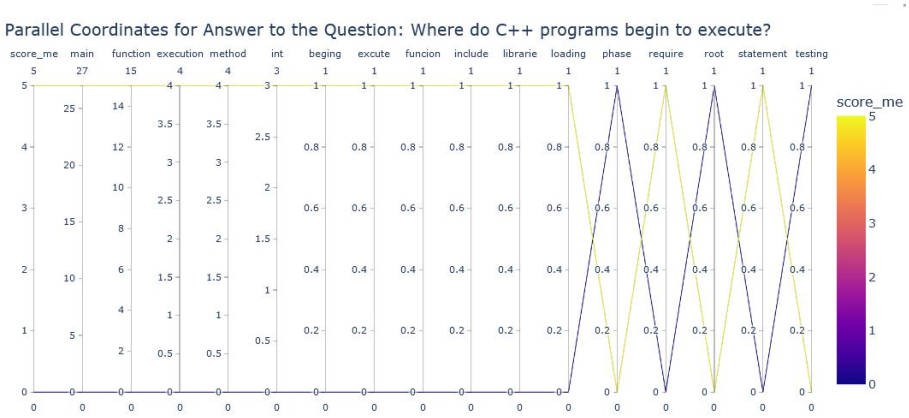

Figure 4: Visualization-Solution: Improved Parallel Coordinates

An improved version for parallel coordinates is shown in figure 4. - Here the answers (the documents) get grouped by the score they received. This solves the issue of not being able to follow the lines and gives a decent overview about the words that the answers with low or a high score consisted of (in our case there are only answers who achieved full or none points). However, the visualization here assumes that the dataset stores already the information on the scores. Which in reality would often not be the case. For example if a teacher would like to analyse the different answers of the questions he would have usually not graded them already.

Overall, it can be concluded that parallel coordinates tend to be rather unsuitable for visualizing our data in this domain. Therefore, they are not applied in the final app.

3.3 Matrix Visualization

Since we can also convert our input texts into vectors, as mentioned in point 3.1.2, we can now use this type of representation for further analysis. One of the central advantages of vectors is the simplified calculation of the commonality. One method for this is the cosine similarity [12], which determines if a pair of vectors points in a similar direction (in a multidimensional space where in our case the terms are the dimensions). This has advantages over the Levenstein distance [13], which can be calculated only with the plain text and where the equality of two documents is determined by the operations (deleting, exchanging letters) that have to be done to get from one text to the other. The reason for this is that the analysis with word vectors and cosine similarity considers the texts ("main method", "in the method main") - as 100% equal (after the stopword removal) - while the Levenstein distance would show a greater distance. The Levenstein distance is therefore also mainly useful only for single words - if it is e.g. about spell correction and there is the actual word to find.

The result of this similarity analysis of each of the documents with each other can also be represented in a matrix visualization [14] - as figure 5 shows. In our case, an nxn matrix is built from quadarats - each document comes once as a row so that all can be compared with each other. The color of the individual squares is then according to the similarity of the documents. This visualization gives you a good overview if there are many documents with similarities and how strong this similarity is. For further questions, however, this visualization method is rather unsuitable and therefore it is not used in the final app.

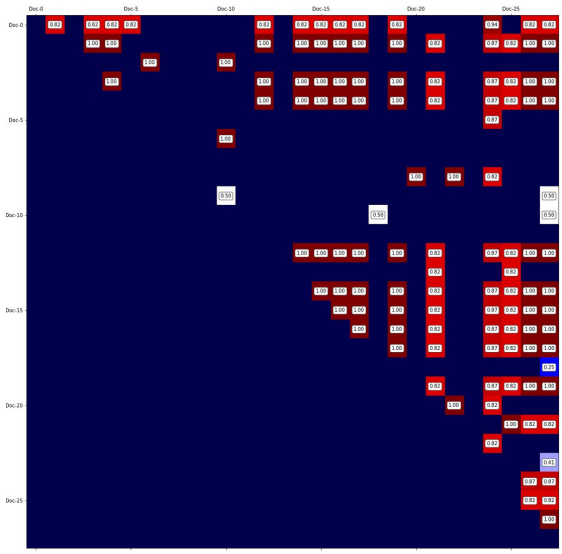

Figure 5: Visualization-Solution: Matrix Visualization

3.4 Dendrograms

Dendograms visualize the similarity of single data records (in our case documents) in a tree-like structure [15]. This method is used especially when it comes to clustering - grouping the individual records [15].

Figure 6 shows an example where the similarity comparison is displayed over the whole set of documents - so you can see how many documents are similar to others and how strong these similarities are - e.g. you can see that Doc-26, 19,17, 16, 15, 14, 12, 4, 1, 3, 10 - form the largest similarity group or that Doc-18 is quite different from all other documents. However, the analysis can also be done by features (in our case words) as shown in figure 7- here you can see, for example, that the words main and function - often occur together in texts. However, both visualizations have the problem of scalablity - that they quickly become confusing - with an increasing number of features and documents.

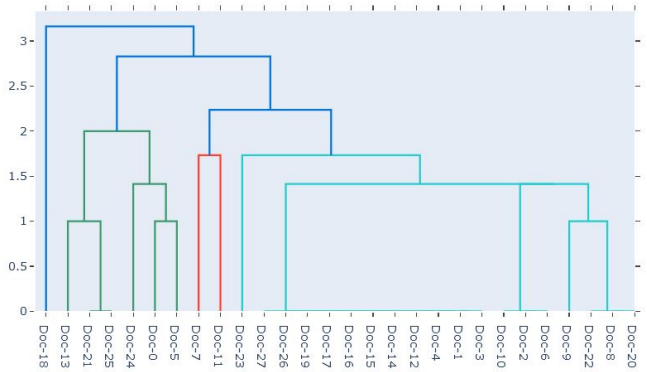

Figure 6: Visualization-Solution: Dendograms Type 1

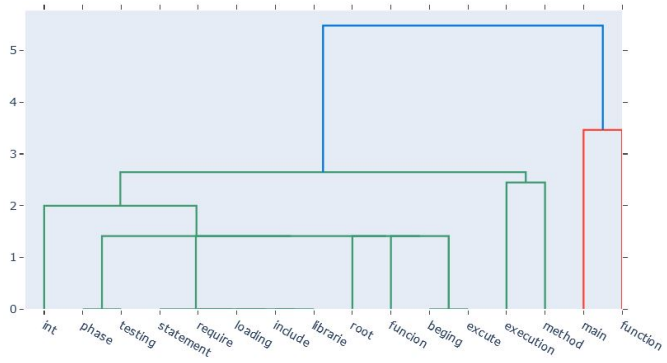

Figure 7: Visualization-Solution: Dendograms Type 2

3.5 Principal Component Analysis

The basic idea of Principal Component Analysis (PCA) [16] is to perform a dimensionality reduction to find a new coordinate system that reflects the information content of the high number of original features. To construct this, a small number (usually two or three) of so-called principal components are found - based on the covariance and eigenvectors of the original features [16]. Two principal components (PC0 on the x-axis and PC1 on the y-axis) created by conducting this analysis can then used to enable that the individual records can then be displayed as individual points in a 2D visualization - as shown in figure 8.

The resulting visualization has the advantage that it functions for 10 answers as well as for 200 or more (only the point size may have to be adjusted). With this kind of visuals it is possible to see which documents are similar and to what extent. The color of the dots again stands for a high or low score that the answers have reached. Here it is easy to see that the visualization arranges documents with a low score close to each other and that these can be easily separated from the other points by a straight line - which speaks for the usefulness of the visualization.

One problem with the PCA plot, however, is that documents that are exactly the same are placed in exactly the same place and thus overlap, preventing you from seeing that there are actually multiple documents at that point and not just one. A solution to this would be to introduce a position dodge factor, but this artificially distorts the data slightly.

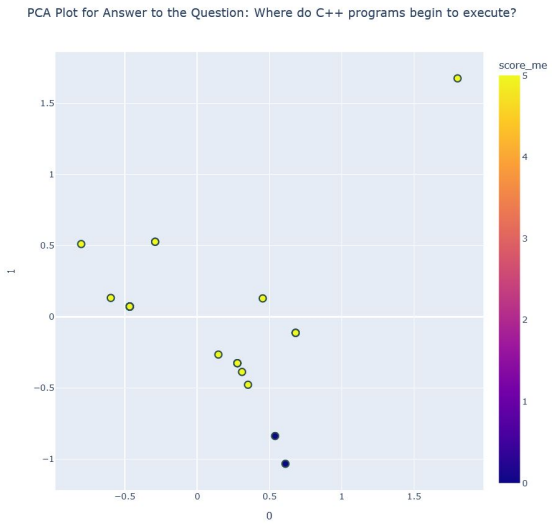

Figure 8: Visualization-Solution: Principal Component Analysis

3.6 Network-based Visualization

Graph-like visualization have been applied in a wide range of domains - some application areas include medicine/biology [17], traffic forecasting [18] or linguistic [19]. One of the main idea often is to represent the n different observation in a dataset as n nodes in a network and their relationships between each other with edges. In traffic analysis [18] we would for example have roads or sensors as nodes - whereas in our case the observations are the different documents.

As already previously discussed we can calculated the similarity of documents of which we have a word-vector representation with cosine-similarity. We can then use the computed cosine-similarity to decide which nodes should have edges between them (in this case edges represent a similarity between he documents). This criteria can also be used to determine the thickness of the edges, as we will we in later visuals.

Figure 11 shows one of the simplest ways to put the above into practice. Here, too, you can see that most of the answers are very similar - while two are clearly different from all the others. The disadvantage of the visualization is that the individual names of the documents overlap and the viewer usually has no idea what is behind document XY - so why do the documents differ.

One advantage of graph visualization is - that the understanding of the relationships of the documents is presented here very intuitively. In addition - if the nodes are represented only as points - a very large number of documents can be compared with each other clearly - we have thus again the advantage of the scalability, as already in the subsection before - with PCA plots.

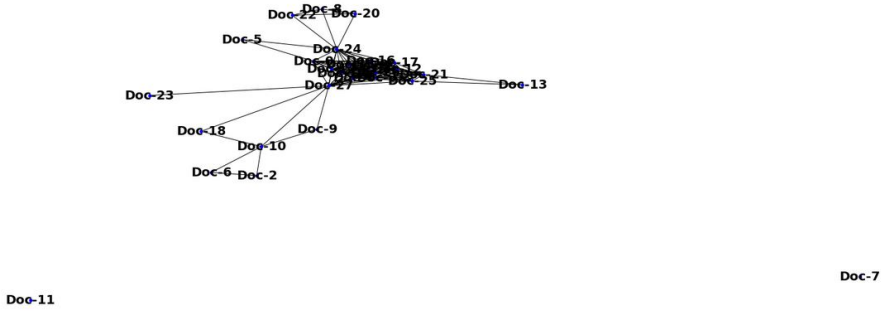

Figure 9: Visualization-Solution: Network-based Visualization

4 Developed Design

This next section describes the final app developed based on the previous experiments. The two most promising visualization methods are used: graph-based and PCA plot.

4.1 practicability

The layout of the app is kept as simple and clear as possible to guarantee a practical use. The user has a total of six controls:

1. **changing the plot type** this will switch the change the network-based visuals to the PCA-ones or visa versa. The control is located above the visualization.
2. **changing the question** this allows the user to easily select the question which he would like to analyse
3. **choose the top n words** this defines how many of the most frequent words for each component in the graph are displayed
4. **choose the similarity threshold** with this tool the control the number of edges drawn - the lower this threshold - the more edges will be drawn.
5. **Define the marker size for the PCA plot** - to avoid problems with scalablity
6. **Set a position jitter of the PCA plot** - to avoid problems of overlapping points

All these options are deliberately always shown - so that the user has full control at all times - which should not lead to any clarity problems with the small number of inputs (six). In addition, the user interface has a certain constancy.

4.2 maintainability

The maintainability of the code is achieved in so far as the code is firstly commented out and secondly also designed in such a way that it should be able to cope with different data from the short answer questions area. It is important that new data have the the following columns that the old dataset also had:

1. question
2. student answer
3. desired answer
4. score - if available

4.3 extensibility

The extensibility is due to the fact that shiny has a kind of modular structure. For example, if you decide to add a new type of visualization, you can simply add it as a new tabset panel and easily integrate it into the existing user interface.

4.4 portability

The portability of the app is guaranteed in that the final version of the app is a shiny application that can be opened in almost any standard browser. The calculation of the visualization is done on the servers and therefore the app can be used on low-powered computers. Theoretically, even a mobile use would be possible - for a tablet, for example, the use would be conceivable, a smartphone display would be mostly too small for the user interface.

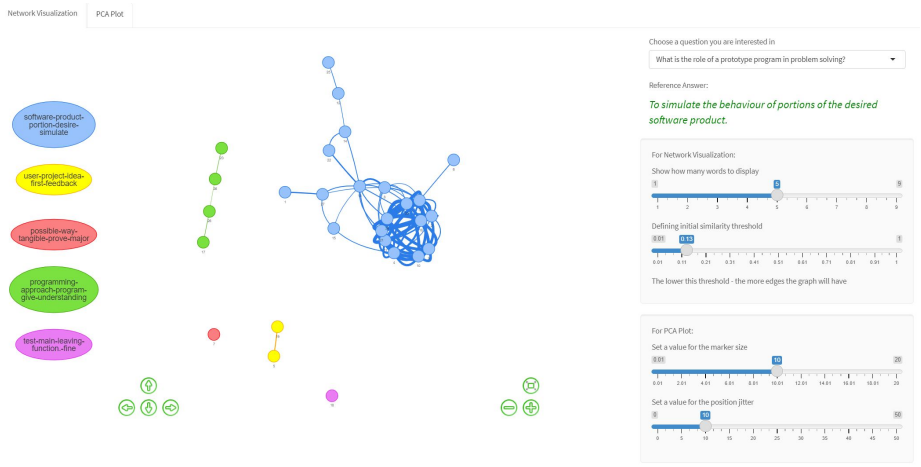

Figure 10: Final User Interface

5 Description of the Source Code

The implementation of the app is divided into the following three parts:

1. **Preporcessing_and_Experimentation.ipynb**

   This Jupyter-notebook fulfills two basic tasks: first, the visualization experiments discussed in section 3 took place there, and second, it puts the data into a form usable for visualization - this outsourcing of the preprocessing activity makes the app itself run faster. Central points here are the cleaning and spell checking, as well as the normalization of the texts (norms are moved here into the singular, with verbs and other words the root form is found (e.g. was becomes is), the stopword removal - this takes place in several steps, on the one hand generic stop-words like (also, is, a) - are removed and on the other hand those words are removed - which occur in the question. After this is done, a principal component analysis is performed and the data is saved in a new document.

2. **app.R** this allows the user to easily select the question which he would like to analyse

3. **hoster.R** - deploys the main app to the web. A shiny user account is requested - and a token must be generated before in order to be able do the hosting.

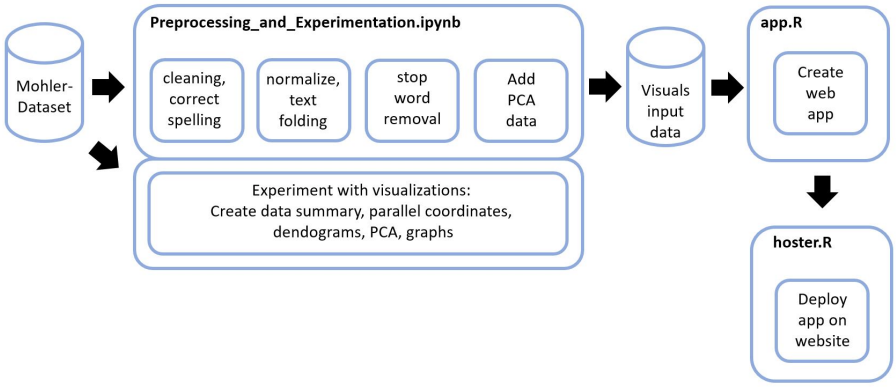

Figure 11: Programming Pipeline

6 Analysis & Evaluation of the proposed solution

Due to the short time frame of the project, it was not possible to perform a detailed evaluation, e.g. by means of a user study. For the sake of simplicity, ten questions were selected and then analyzed by one researcher. A basic problem in the analysis is that the interpretation of whether a visualization is useful is often highly subjective and can be evaluated according to a wide variety of criteria. For a detailed user study, this would therefore be defined more precisely. However, the goal of this evaluation is only to reflect the intuitive assessment of a single user in order to achieve a first idea about the actual the usefulness of the proposed solution.

Figure 13 shows the results of the analysis divided by visualization type. First, the interaction (the use of the controls presented in section 4.1.) often seems to be a good way to make the visualization clearer for the user. For the graph visualization it is mainly the similarity threshold that was used most often and for the PCA the position doge (to detect overlays).

Furthermore, it should also be noted that it is usually possible to visualize the similarity and differences of documents - but it is more difficult to show why they differ (e.g. by indicating the top N words in a graph). However, the hover tool is useful here, which displays the original document/student answer when the mouse is hovered over a point, as which which you can usally get a good idea why two documents are assumed to be similar.

'Where do C++ programs begin to execute?',
'What is the scope of global variables?',
'What is the stack operation corresponding to the enqueue operation in queues?',
'What is typically included in a class definition?',
'What operations would you need to perform to find a given element on a stack?',
'What stages in the software life cycle are influenced by the testing stage?',
'When defining a recursive function, what are possible causes for infinite recursion?',
'When does C++ create a default constructor?',
'Where are variables declared in a C++ program?',
'What is the role of a prototype program in problem solving?'

Figure 12: Evaluation Questions

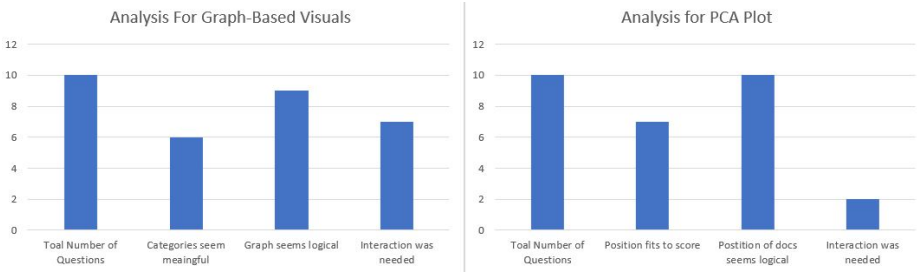

Figure 13: Evaluation Results

7 Conclusion

The goal of this project was to identify the current challenges and problems in comparing documents in the Short Text Answer area, to investigate them empirically, and then to develop a prototype based on the findings. This prototype has been completed and can be tested in almost any web browser.

Some of the main findings from this work are - that sclablity is a key difficulty, that visualizations need to adapt well to different types of questions and that there is probably not one visualization method that works best for every question, but the best technique often depends on the specific case.

8 Limitations And Future Work

For future work, it would be interesting to conduct a detailed user study. Ideally, the system would be tested out with different data on exams/tests from a variety of areas. Afterwards, a qualitative survey could be conducted and the software solution could be tested for real practical suitability. For example, it could also be investigated whether the visualizations and the controls are intuitively understood.

An evaluation of the user design with interviews and other techniques would also be of interest. For example, could the interface be made clearer and how is the UI generally received by the user - do some design elements confuse them or do they find their way around with all of them right away?

A limitation of this project was that obviously not all visualization possibilities were tested out and those that were explored were not analyzed to the last detail, which would have simply exceeded the time frame of the project. For future work, however, it would be interesting to consider even more techniques and to analyze the existing ones in greater depth.

Another interesting field of research for the future is the customizing of visualizations. It would be interesting to find out how far the user can change the visuals himself (e.g. by changing the node size) or whether some settings should rather be made automatically or even be kept constant.

References

[1] Tuanji Gong and Xuaxia Yao. An attention-based deep model for automatic short answer score. *International Journal of Computer Science and Software Engineering*, 8(6):127–132, 2019.

[2] Kostiantyn Kucher and Andreas Kerren. Text visualization techniques: Taxonomy, visual survey, and community insights. In *2015 IEEE Pacific visualization symposium (pacificVis)*, pages 117–121. IEEE, 2015.

[3] Harri Siirtola, Poika Isokoski, Tanja Säily, and Terttu Nevalainen. Interactive text visualization with text variation explorer. In *2016 20th International Conference Information Visualisation (IV)*, pages 330–335. IEEE, 2016.

[4] Dhara J Ladani and Nikita P Desai. Stopword identification and removal techniques on tc and ir applications: A survey. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 466–472. IEEE, 2020.

[5] Vimala Balakrishnan and Ethel Lloyd-Yemoh. Stemming and lemmatization: A comparison of retrieval performances. 2014.

[6] Andrea Esuli and Fabrizio Sebastiani. Training data cleaning for text classification. In *Conference on the Theory of Information Retrieval*, pages 29–41. Springer, 2009.

[7] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR, 2015.

[8] Maximilian Schmitt and Björn Schuller. Openxbow: introducing the passau open-source crossmodal bag-of-words toolkit. 2017.

[9] Sang-Woon Kim and Joon-Min Gil. Research paper classification systems based on tf-idf and lda schemes. *Human-centric Computing and Information Sciences*, 9(1):1–21, 2019.

[10] Michael Mohler, Razvan Bunescu, and Rada Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 752–762, 2011.

[11] Julian Heinrich and Daniel Weiskopf. State of the art of parallel coordinates. *Eurographics (State of the Art Reports)*, pages 95–116, 2013.

[12] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1, 2012.

[13] Li Yujian and Liu Bo. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095, 2007.

[14] Han-Ming Wu, ShengLi Tzeng, and Chun-houh Chen. Matrix visualization. In *Handbook of data visualization*, pages 681–708. Springer, 2008.

[15] M Forina, C Armanino, and V Raggio. Clustering with dendrograms on interpretation variables. *Analytica Chimica Acta*, 454(1):13–19, 2002.

[16] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[17] Georgios A Pavlopoulos, Panagiota I Kontou, Athanasia Pavlopoulou, Costas Bouyioukos, Evripides Markou, and Pantelis G Bagos. Bipartite graphs in systems biology and medicine: a survey of methods and applications. *GigaScience*, 7(4):giy014, 2018.

[18] Bing Yu, Mengzhang Li, Jiyong Zhang, and Zhanxing Zhu. 3d graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting. *arXiv preprint arXiv:1903.00919*, 2019.

[19] Open Culture. A colorful map visualizes the lexical distances between europe's languages: 54 languages spoken by 670 million people. https://www.openculture.com/2017/08/a-colorful-map-visualizes-the-lexical-distances-between-europes-languages.html.

9 Appendix

An example for parroting in the dataset - as justifciation for removing the words mentioned in the question itself

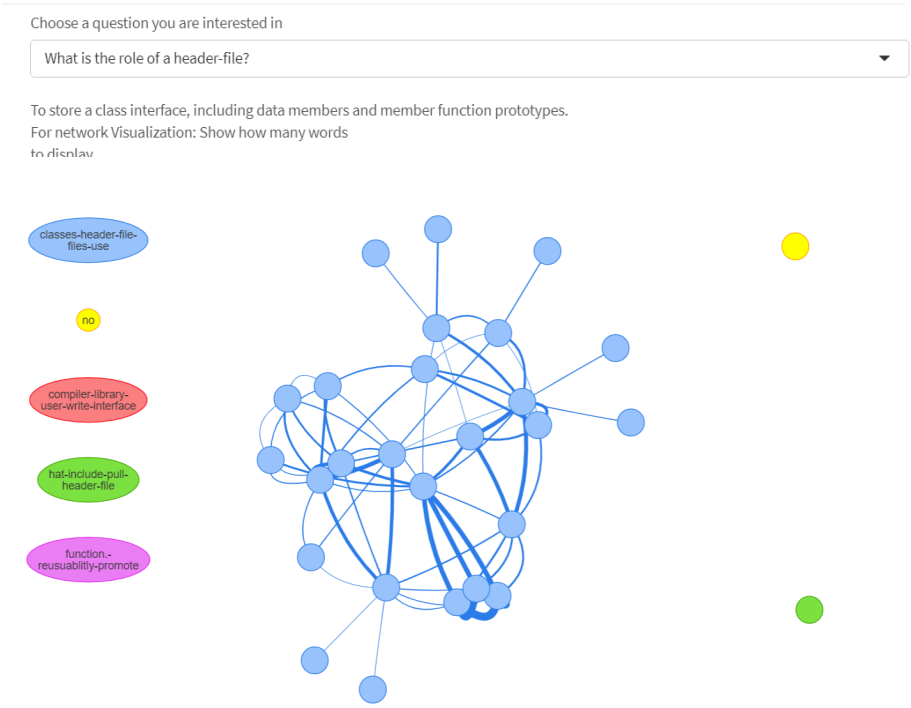

Figure 14: parroting Example