Zurich University
of Applied Sciences

School of
Engineering
CAI Centre for
Artificial Intelligence

Bachelor thesis (Computer Science)

Summarize This!
Automated generation of meeting highlights

Author	Mika Ruch Simon Zaugg
Main supervisor	Mark Cieliebak
Sub supervisor	Don Tuggener
External supervisor	Manfred Vogel

Date	10.06.2022

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

DECLARATION OF ORIGINALITY

Bachelor's Thesis at the School of Engineering

DECLARATION OF ORIGINALITY
Bachelor's Thesis at the School of Engineering

By submitting this Bachelor's thesis, the undersigned student confirms that this thesis is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the Bachelor thesis have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date:	**Name Student:**
Zürich, 10.06.2022	Mika Ruch
Zürich, 10.06.2022	Simon Zaugg

Abstract

For many people, it is a weary task to keep meeting minutes. In addition, most meetings are not minuted at all. With meetings held increasingly in an online setting, this offers the opportunity to record and transcribe (i.e., turn recording into text) the meetings automatically. The Interscriber web app creates such transcripts based on an audio recording. However, it would be useful, particularly for people who did not attend, to receive a meeting summary containing the most important information. With this summary, they do not have to read the whole literal protocol.

Currently, much research is being done in the field of NLP (Natural Language Processing), the science of letting AI (Artificial Intelligence) understand, interpret, and manipulate natural language. The most advanced algorithms and AIs retrieve separate pieces of information. AIs that directly transform natural language into summaries still make many mistakes and may only be used with care.

Our work lays the foundation for integrating NLP solutions that retrieve separate pieces of information into Interscriber. For our thesis, we developed a feature within Interscriber that incorporates meeting information retrieved by NLP algorithms into predefined sentence templates. The most relevant sentences are portrayed as the highlights of the meeting. They are directed to people who did not attend the meeting. However, it is assumed that the Interscriber users have attended. Therefore, we provide a feature that enables them to view all generated sentences and select those they deem most relevant. After evaluating the highlights, they may email them to the people who did not attend the meeting.

For future developers who will manage the sentence templates, we designed our feature to be easily extendable with additional sentence templates and NLP APIs. In addition, we provide comprehensive instructions for managing the phrase templates.

Our final solution is a templating engine within Interscriber. The user feedback we gathered shows the feature is beneficial to the Interscriber users.

Zusammenfassung

Für viele ist es eine mühsame Angelegenheit, für Meetings Protokoll zu führen. Hinzu kommt, dass die meisten Meetings überhaupt nicht protokolliert werden. Da Meetings zunehmend online abgehalten werden, bietet sich die Möglichkeit, diese automatisch aufzuzeichnen und zu transkribieren (d. h. die Aufnahme in Text umzuwandeln). Die Web-App Interscriber erstellt solche Transkripte auf der Grundlage einer Audioaufnahme. Allerdings wäre es vor allem für Personen, die nicht an dem Meeting teilgenommen haben, nützlich, eine Zusammenfassung der wichtigsten Informationen zu erhalten, damit sie nicht das ganze Protokoll lesen müssen.

Derzeit wird viel im Bereich von NLP (Natural Language Processing) geforscht. NLP ist die Wissenschaft, KI (Künstliche Intelligenz) dazu zu bringen, dass sie natürliche Sprache verstehen, interpretieren und manipulieren können. Die fortschrittlichsten Algorithmen und KIs rufen spezifische Informationen ab. Dagegen machen KIs, die direkt Zusammenfassungen in natürlicher Sprache generieren, noch viele Fehler und können nur mit Vorsicht eingesetzt werden.

Unsere Arbeit legt den Grundstein für die Integration von NLP-Lösungen, die spezifische Informationen abrufen, in Interscriber. Für unsere Arbeit haben wir eine Komponente innerhalb von Interscriber entwickelt, die vordefinierte Satzvorlagen mit Informationen bestückt, welche von NLP-Algorithmen aus Meetings herausgelesen werden. Die relevantesten Sätze werden als Highlights des Meetings dargestellt. Sie richten sich an Personen, die nicht am Meeting teilgenommen haben. Es wird jedoch davon ausgegangen, dass die Nutzer von Interscriber am Meeting teilgenommen haben. Daher bieten wir eine Funktion an, die es ihnen ermöglicht, alle generierten Sätze anzusehen und diejenigen auszuwählen, die sie für besonders relevant halten. Nachdem sie die Highlights beurteilt haben, können sie diese an die Personen schicken, die nicht am Meeting teilgenommen haben.

Für künftige Entwickler, die die Satzvorlagen verwalten werden, haben wir unsere Funktion so konzipiert, dass sie leicht mit zusätzlichen Satzvorlagen und NLP-APIs erweitert werden kann. Darüber hinaus bieten wir eine umfassende Anleitung zur Verwaltung der Satzvorlagen.

Das Ergebnis unserer Arbeit ist eine Templating-Engine innerhalb von Interscriber zum Erstellen von Meeting Highlights. Das von uns gesammelte Benutzer-Feedback zeigt, dass die Funktion für Interscriber-Benutzer einen Mehrwert bringt.

Preface

Table of contents

1 Introduction

Meetings are held everywhere. They are the source of most of the important decisions, discussions, solutions, and presentations. With more and more meetings held online, the research in this field intensifies. Nowadays, many products exist on the market that transcribe (i.e., turn recording into text) meetings with good quality. While this is great, most people do not want to read the hole transcript utterance by utterance, but just get the most important conclusions out of it. Therefore, it would be beneficial, especially for people that did not attend the meeting, to receive a summary of the transcript. Automatic meeting summary creation, on the other hand, is a field of research that still needs a lot of work. There are many different approaches to the problem. The information can be portrayed in diagrams, structure text (e.g., lists and tables), or natural language. To generate these, various algorithms and AI models are available.

For this thesis we develop a tool that reliably creates meeting summaries based on predefined template sentences for the web app Interscriber, explained in the chapter 1.1.The goal is to build a stable foundation for meeting summarization within Interscriber, which is easy extendable by future developers. The solutions include a user interface (UI), which is designed with intuitiveness and user experience (UX) in mind, and the highlight sentence engine, which is responsible for generating the sentences.

We also analyze and experiment with NLP (Natural Language Processing) solutions, which can generate summaries from written text and conversations. This is to better understand where the advantages and downfalls of this type of summarization processes lays.

Then, we will explain our own summarization approach. How template-based sentence summary generation works, and which sentences we choose to implement. And which are currently possible to generate with the Transcript Analyser project, explained in the chapter 1.2.

Once there is an understanding of how our summarization approach works, we compile a list of user stories and use cases to define what the feature should be able to do. To refine the use cases the UI mockups were designed next. During the creation of the mockups, attention is given on the usability and intuitively of the feature itself. Once the mockups are done and implemented, a usability study is made. Suggested improvements are implemented.

Then the core part of this thesis is described, the highlight sentence engine. We analyze the architecture, describe how the generation works in general and how the ranking of the sentences is calculated.

Since a target group of this feature are developers, a technical documentation is written. This should help future developers using our engine to add more sentences.

Lastly, we do a general User Evaluation study, to analyze how the implemented solution works and if people would use it. To finish up, we analyze the solution critically and give an outlook on what features could be implemented in the future.

1.1 Interscriber

The Interscriber platform was created by SpinningBytes AG. Interscriber specializes in the transcription of meeting protocols. Interscriber utilizes speech-to-text APIs from Google, AWS, and IBM in addition to its own speech-to-text engine. This on-premises solution is particularly critical for confidential meetings that should not be shared with third parties. Interscriber can currently transcribe meetings in both German and English. There is an ongoing effort to include Swiss German.

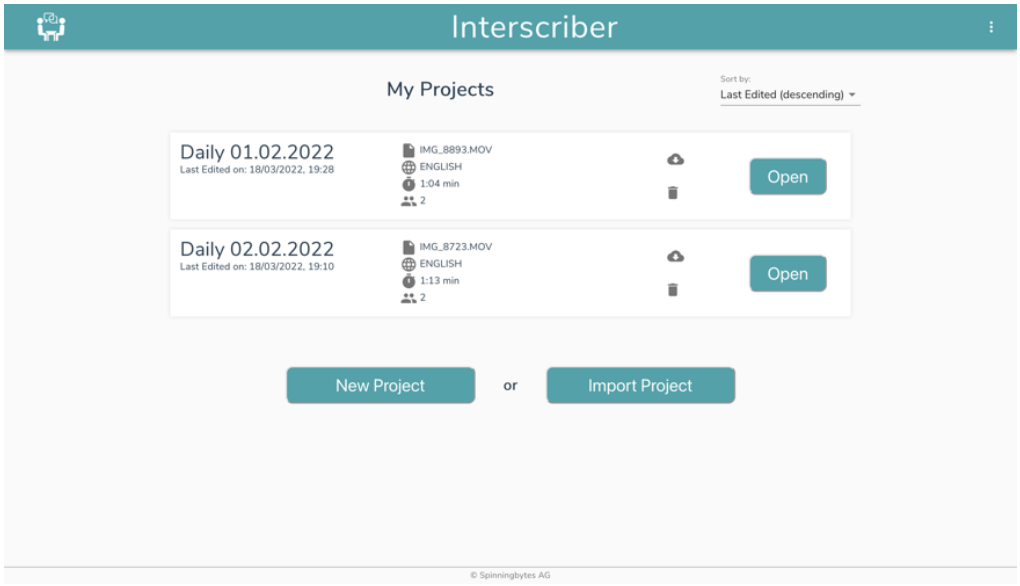


Figure 1: Interscriber dashboard page with multiple meetings

Interscriber, unlike many other speech-to-text engines, can recognize the speaker and allocate each statement to the correct meeting attendee. Once the transcription is complete, users can alter and reassign the utterances using the unique transcript editor. The transcript is exportable into text, Word, and Excel formats. Additionally, the entire project can be downloaded in a format that is compatible with Interscriber.

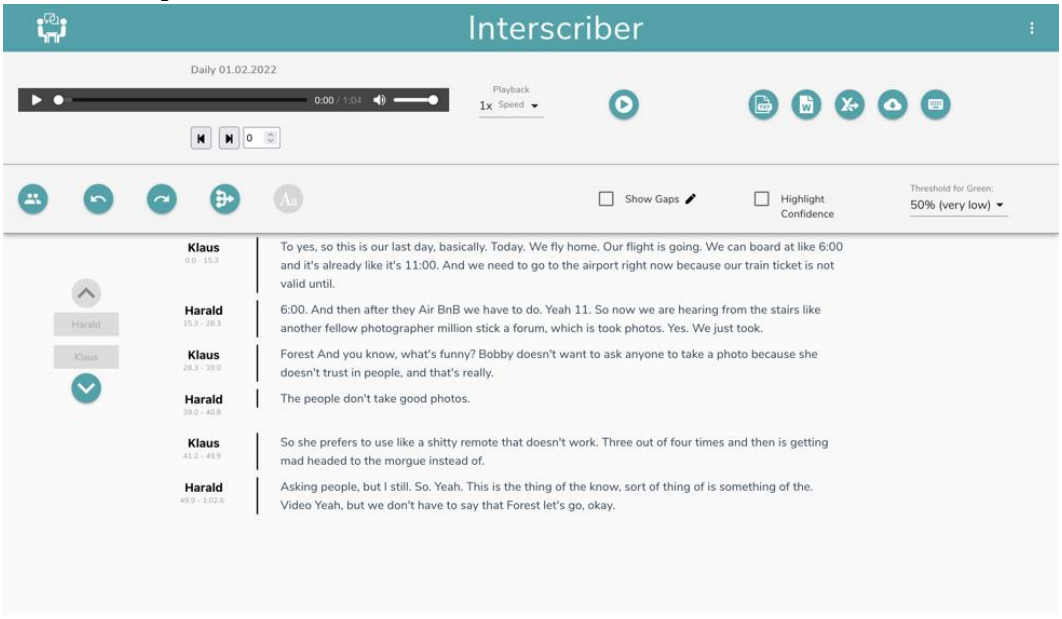


Figure 2: Interscriber transcript editor

1.2 Transcript Analyser

In addition to Interscriber, there exists a separate project known as Transcript Analyser. Its endpoints are capable of consuming Interscriber transcripts. In its current state, the Transcript Analyser project includes numerous endpoints for extracting diverse information from the transcript. Currently, the endpoints can extract general statistics, relevant keyphrases, important text blocks, related words, and the sentiment for each sentence.

In the next section, we will discuss what each relevant endpoint does and what it returns.

Statistics

The statistics endpoint extracts information from the transcript. It computes the overall number of speakers, the total number of utterances, and the speaking time allotted to each meeting participant. Additionally, it uses the keyphrases logic to extract the three most important topics of the meeting.

Keyphrases

There are numerous algorithms for extracting keyphrases, it may be advantageous to use different algorithms for various use cases. Therefore, this endpoint provides a total of three algorithms to choose from by the caller. The chapter 3.2 addresses these different algorithms.

Sentiments

It can be particularly interesting for supervisors and managers to learn what emotions were present during a meeting. The sentiments endpoint evaluates the sentiment (i.e., positive, or negative) of every utterance in the transcript. Additionally, the confidence score (i.e., how sure was the tool that it is positive or negative – indicated by a number between 0 and 1) of said result is passed as well.

Filtering

In the transcript each utterance is assigned to a speaker via a speaker id. Every endpoint has the capability to filter the transcript by speaker id. This can be used to identify the most important topics or sentiments for a certain speaker or a speaker group. In addition, the endpoints can be filtered by start and end time to enable granular analysis of a specific text passage.

2 Natural Language Summarization

When it comes to summaries and highlights, most people expect them to be written in natural language. A significant amount of current research is dedicated to training AI models which can generate human like written texts. As a result, we analyze existing models and products that generate summaries in natural language. Then we discuss their limitations and the reasons why we chose a different technique for creating meeting highlights.

2.1 BART

BART is a pretraining model that can be used to train sequence-to-sequence models proposed by Mike Lewis et al. in 2020 [1]. A sequence-to-sequence model can take text, for example a meeting transcript, and transform it to another text, for example a summarization. BART trains these models with a Bidirectional Transformer based on BERT [2] and the Auto-Regressive Transformer GPT [3]. BERT encodes a text bidirectional by analyzing the context for each word and sentence. The result is a representation of these as vectors in a high-dimensional vector space, also referred as word embeddings. GPT, on the other hand, works auto-regressively, predicting an output sequence based on an input sequence. However, because it decodes text from left to right, it only considers the leftward context. BART combines these two approaches to train models that generate texts by taking the full context into consideration.

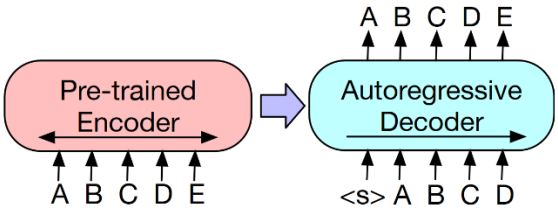


Figure 3: BART training model
Source: Adapted from [1]

Models trained with BART can be trained for different NLP tasks. Bart-large-cnn-samsum [4] is a pretrained model that specializes in conversation summarization. It is trained on text summaries extracted from the cnn_dailymail dataset [5] and messenger-like conversation summaries extracted from the samsum dataset [6]. BART is more effective than other NLP models when it comes to generating and understanding text.

2.2 GPT-3

GPT-3 is an NLP model developed and trained by OpenAI and was released in 2020 [7]. GPT-3 is the third generation of the Generative Pre-trained Transformer (GPT) model, which was introduced in the chapter 2.1. It uses deep learning to generate natural language texts.

GPT-3 operates by reading the entire input document and predicting the output word-by-word. Since GPT-3 was trained on massive amount of data, it can be fine-tuned to a specific task using only small amount of data. Due to this, GPT-3 can achieve a state-of-the-art level of summarization with only a few examples of fine-tuning.

The results of GPT-3 are often difficult to distinguish from texts written by humans [8]. Despite the sophistication of GPT-3, its summaries should be viewed with caution, as the AI may still make semantic errors. [9]

2.3 Related Product Analysis

Now that two of the most prominent NLP models have been analyzed, we can search for products that employ and expand upon these or similar models. The focus of the competitor analysis is on the summarization and highlight extraction of flow text and transcripts. The competitors listed below represent a subset of those currently available.

In a first step, we analyze the applications themselves, thereafter we evaluate each summarization tool using various transcripts and flow text.

Sembly AI [10] is a comparable application to Interscriber. Simply by inviting a technical user to a meeting, it can be transcribed and analyzed. Sembly AI extracts excerpts for various categories, like actions, issues, risks, or requirements. The results are displayed on the website within various tabs, structurally summarizing the meeting.

However, uploading a recording of a meeting to Sembly AI is not possible.

Interscriber does not have a technical user who can be invited into a meeting. This feature would enable Interscriber to assign sentences more precisely to their respective speakers.

QuillBot [11] is a tool for improving the wording of texts through rephrasing. However, it can also be used to automatically generate concise, readable summaries from text. In most cases, it heavily relies on the original wording; however, in some instances, QuillBot constructs its sentences by condensing and merging sentences from the original.

Wordtune [12] is like QuillBot in that it is a tool for paraphrasing and creating summaries. However, Wordtune produces a list of small paragraphs, which are intendent of each other. Additionally, it can handle short texts as well as lengthy documents, with the size of the summary varying according to the length of the source. On average, it produces one summary per paragraph of the original document.

Summari [13] is one of the numerous smaller companies that produce text summarization tools. It is limited to summarizing websites. The summary is a compilation of the most relevant sentences, with some sentences shortened or combined.

2.4 Discussion

We asses each tool using three different texts. The test results can be found in the Appendix chapter 15.3.

Test Data Name	Source	Reference
Switzerland Wikipedia	[14]	15.3.1
Slogan Meeting	own work	15.3.2
Weekly Meeting	[15]	15.3.3

Table 1: Test data of natural language summary experiment

Next, we analyze the outcome of the tests and highlight what went well and where problems were discovered.

2.4.1 Model and Product Experiment

The **BART** [1] approach is evaluated by using the **bart-large-cnn-samsum** [4] model. The results of our evaluation indicate that the model can produce adequate meeting summaries written in natural language. Eventually, however, the model makes semantic errors.

The arguably best meeting summaries were created by the **GPT-3** [7] model. As demonstrated by other studies, it is also susceptible to semantic mistakes [9]. However, we have detected fewer errors than in other models and products.

The resulting summary of **Sembly AI** [10] explores the entire meeting. During our test it accurately extracted excerpts for the category types of actions and requirements. However, there is no link between the excerpts and the original transcript. Therefore, context from the preceding discussion is lost, obscuring potentially vital information.

We also discovered that neither written nor prerecorded meetings can be uploaded, limiting us to testing the product with only one of our three test texts.

Wordtune [12] is the most advanced text summarization product we have evaluated. Wordtune can combine information from multiple sentences into one. It is even capable of generating useful meeting summaries from transcripts.

Even though **Wordtune** specializes in summarizing longer documents, our tests demonstrated that it is also capable of summarizing written transcripts. It can identify relevant information and incorporate it into a concise summary.

QuillBot [11] typically generates a concise, natural-language summary containing the most essential information of the document. During testing, we discovered that **QuillBot** selects the most meaningful sentences and uses them with minor modifications. In some instances, it reformats or concatenates information into one sentence.

Summari [13] is limited to website summarization only. Therefore, it cannot be used for meeting summaries of any kind. However, we were still curious about the outcomes of this summarization tool. Unfortunately, although the summaries are adequate, it is not written naturally. Several sentences contained grammatical or semantic errors.

2.4.2 Mistakes: Changing the Meaning

Every summarization solution made in some instances the critical mistake to accidentally alter the meaning of the. This is demonstrated by the three instances listed below. All results can be found in the Appendix chapter 15.3.

Example 1
This example is from the **BART** summary of the Slogan Meeting.

BART extracted following sentence:

> *Holly and Tim went hiking with Silvy and Greg at the weekend.*

The issue is that in the original text Tim merely discusses this trip he went on with Holly. Silvy and Greg did not join this excursion. This is a typical issue that arises when creating summaries with BART or GPT-3.

Example 2
This summary was extracted by **Summari** from the Switzerland Wikipedia article.

> *Switzerland is a landlocked country bordered by Italy to the south, France, and Germany to the west, and Austria and Liechtenstein to the east.*

The statement that France and Germany share a western border with Switzerland is incorrect. This contradicts the original document, which states that Switzerland shares a western border with France and a northern border with Germany. The meaning of the text is altered by omitting important contextual information.

Example 3
This example was extracted by Wordtune also from the Switzerland Wikipedia article.

> *Switzerland is at the crossroads of Germanic and Romance Europe. Its national identity is rooted in Alpine symbolism.*

It states that the Swiss national identity is rooted in Alpine symbolism. While this is accurate, the summary implies that it is the only or most significant source of national identity. In contrast, the original text only mentions Alpine symbolism as one of many roots.

2.5 Conclusion

Good summaries in natural language are difficult to achieve. Due to the absence of context, selecting the seemingly most relevant sentences rarely results in a good summary. Whereas, generating sentences frequently results in the invention of new information which is not present in the original text.

Even though some technologies produce impressive results, they eventually make mistakes. The implication for the user is that they cannot rely on the summary without first validating it against the original text. Some products have taken this into consideration and offer user interfaces for direct summary validation and cleanup.

Therefore, we will develop a tool that only displays information with a high likelihood of being accurate. We accomplish this by utilizing specialized NLP algorithms to extract specific information. Then, we incorporate them into predefined sentence templates. The outcome is a collection of sentences that describe the meeting. This approach is called Template Base Summary.

3 Template Based Summary

In contrast to the natural language summary, our method provides a more static approach to summarizing. Instead of generating sentences from scratch, we make use of predefined template sentences into which only dynamic data must be inserted.

This allows us to utilize NLP algorithms that have been trained for one specific task. This provides a more reliable method of data collection and avoids the issue of a meaning being altered as what can be seen in natural language summary.

3.1 Sentences

We have compiled a list of phrases that attempt to summarize a meeting. It describes the interactions between meeting participants, the topics discussed, and any potential action items or events that were agreed upon.

Most important topics were **Topic1**, **Topic2** and **Topic3**.
The sentiment of the **meeting** was **positive**.
Klaus talked most about **Topic1**.
Klaus was very dominant, speaking more than **60%** of the time.
Klaus spoke the most, **Hans** spoke the least.
Present in the meeting: **Klaus, Hans**
Topic3 was the topic where most participants contributed something.
Event1 will take place on the **21.08.2022**.
The deadline for **ActionItem1** is set to **22.08.2022**.
Klaus finishes **ActionItem1** by **22.08.2022**.
The main decisions were **Decision1, Decision2**.
Klaus spoke critical of **Topic 1**.
The topics on the agenda were: **AgendaTopic1, AgendaTopic2**.
There was a fight between **Klaus** and **Hans** for **4 minutes**.

Table 2: Highlight sentence type/templates

3.2 What is possible with Transcript Analyser

The Transcript Analyser project is in its early stages. It can retrieve statistical information, keyphrases, important text blocks, and related words from a given transcript.

For our purposes, the statistical information, keyphrases, and sentiment endpoints are the most relevant. They could enable us to create the following sentences:

Template Sentence	**Proposed Endpoint to use**
Most important topics were **Topic1**, **Topic2** and **Topic3**.	Keyphrases
The sentiment of the **meeting** was **positive**.	Sentiment
Klaus talked most about **Topic1**.	Keyphrases with speaker filter
Klaus was very dominant, speaking more than **60%** of the time.	Statistics
Klaus spoke the most, **Hans** spoke the least.	Statistics
Present in the meeting: **Klaus, Hans**	Statistics

Table 3: Possible highlight sentences with Transcript Analyser

The keyword extraction endpoint, however, implements multiple algorithms, such as RAKE, YAKE!, and KeyBERT. A quick analysis and comparison of the algorithms will determine which one will be used for our implementation.

The thesis by Aigner, Gerber, and Rohr [16] analyzes all the subsequent algorithms in depth. Consequently, we will provide a brief overview of how the algorithms function and what their advantages and disadvantages are. Please refer to the chapter "*5 Keyword Extraction*" of [16, pp. 7-16] for a more detailed explanation with examples.

3.2.1 RAKE

Rapid Automatic Keyword Extraction or RAKE is a keyword extraction algorithm developed by Stuart Rose et al. in 2010 [17]. The motivation was to 'develop a keyword extraction method that is extremely efficient, operates on individual documents ... is easily applied to new domains, and operates well on multiple types of documents, particularly those that do not follow specific grammar conventions.' [17, p. 5].

A quick overview on how RAKE works

1. Split the text by stop-words and phrase delimiters and remove those
2. Build contender keywords.
   Contender keywords are all possible keywords, which need to be evaluated by its relevance. A candidate keyword is created by combining words that are not separated by stop-words.
3. Determine the frequency and degree of the words
   Frequency: the number of times the keyword appears in the text
   Degree: the sum of the frequency of the word and the co-occurrences with other words
4. Determine the final score for each word. RAKE uses *deg(w)/freq(w)* to calculate the final score.
5. Rank the keywords according to their highest final score

3.2.2 YAKE!

Yet Another Keyword Extractor or YAKE! is a keyword extraction algorithm developed by Ricardo Campos et al. in 2020 [18]. This algorithm has the significant advantage of not requiring training on any text corpus. Additionally, the algorithm is applicable to all languages without modification.

A quick overview on how YAKE! works

1. convert the text into a machine-readable format and identify candidate keywords
2. Apply the feature extraction to each individual word
   Casing: Words with capital letters are more relevant (except start of sentence)
   Word position: Words at the beginning of the document are more relevant than those towards the end
   Word frequency: More frequent words are more relevant -> can cause a bias towards frequent words in large documents
   Word relatedness to context: Words which often occur with a variety of other words are less relevant – e.g., *the* (occurs with many different words) is less relevant than *white* (may only occur with *white house*)
   Term different sentence: Words which occur frequently are more relevant
3. Determine the total score based on the feature scores
4. Create n-grams and calculate the candidate keyword score
   The n-gram number defines the maximal size of a keyword
5. Remove similar keywords and rank based on the keyword score

3.2.3 KeyBERT

KeyBERT is a keyword extraction algorithm developed by Maarten Grootendorst et al. in 2020 [19]. It is leveraging the Bidirectional Encoder Representations from Transformers or BERT which was developed by Jacob Devlin et al. in 2018 [2]. The main difference between the above-mentioned algorithms and KeyBERT is that the former rely on statistical text properties, whereas KeyBERT emphasizes semantic similarities. Making this possible is BERTs representation of the words as vectors in a high-dimensional vector space, also referred as word embeddings. Generally, words with similar meanings are also close together in the vector space.

A quick overview on how KeyBERT works

1. Separate by and remove stop-words
2. Extract the contender keywords
   CountVectorizer [20]: extracts keywords according to a particular n-gram range (i.e., range 1-3 extracts a keyphrases consisting of one to three words)
   KeyphraseVectorizers [21]: extracts keywords based on a part-of-speech (pos) pattern. A pos pattern describes the structure of a particular phrase. The default pattern is <J.*>*<N.*>+ This indicates that the phrase should start with zero to multiple adjectives followed by at least one noun.
3. Transform the initial text (into one) and candidate keywords (into many) BERT sentence embedding.
   Sentence embeddings are like word embeddings. The meaning of sentences with close vectors is comparable.
4. Calculate the cosine similarity between the original text vector and each contender keyword vector. Select the closest in similarity (the smaller the value the more similar)

3.3 Discussion

Based on the algorithms and their inner workings, YAKE and KeyBERT appear to be in the lead. If the feature is to be used for multiple languages, YAKE looks to be a great option. Multilingual use of KeyBERT is also possible, but the transformer model must be retrained for the specific language in question. However, the plan is to initially only support English, which should not be a problem for any of the three.

As demonstrated in *chapter 5.5* of [16, pp. 14-16], the KeyBERT algorithm is up to one hundred times slower than the second slowest. This will not be a problem for shorter transcripts, but meetings can easily last several hours.

However, KeyBERT leads in one major category. It can comprehend the context of a word while others cannot.

To assess the utility of each algorithm, we will test each with three distinct transcripts. Then, we will evaluate the top three extracted keywords and compare them with one another to determine which algorithm is superior.

For testing purposes, the recordings were uploaded to Interscriber without any post-upload modifications. This was done to test the generation of highlights in the worst scenario. That is, the user did not modify the protocol.

Transcript	**Length**	**Source**	**Reference**
Mary Susan interview	1:40 min	Transcript Analyser	15.4.1
Consulting Interview	13:30 min	[22]	15.4.2
Interview with feedback	8:30 min	[23]	15.4.3

Table 4: Test data description

3.3.1 Result of the Mary Susan Interview

The complete result can be found in the Appendix chapter 15.5.1.

RAKE	**YAKE!**	**KeyBERT with part-of-speech pattern**	**KeyBERT with n-gram**
uh huh mary hi hello	Susan Thompson Resource	kitchen jobs	experience working kitchen
Susan thompson resource manager	Thompson Resource manager	susan thompson resource manager	mary experience working
would really like	Susan Thompson	writing skills	kitchen want learn

Table 5: Keyword extraction experiment result of Mary Susan interview

3.3.2 Result of the Consulting Interview

The complete result be found in the Appendix chapter 15.5.2.

RAKE	YAKE!	KeyBERT with part-of-speech pattern	KeyBERT with n-gram
asked really important clarifying questions	Verizon	cellular carrier	theo wi fi
four hundred dollar cost makes	customers	wearable device	partner cellular
old 1 billion dollar business	customer	cellular carriers	want partner cellular

Table 6: Keyword extraction experiment result of consulting interview

3.3.3 Result of the Interview with Feedback

The complete result can be found in the Appendix chapter 15.5.3.

RAKE	YAKE!	KeyBERT with part-of-speech pattern	KeyBERT with n-gram
list every almost every semester	Dean List	interview question	help preparing interview
part formula 1 pick something	good	interview	preparing interview
professional seasoned professional level	questions	resume	professional level interviewing

Table 7: Keyword extraction experiment result of interview with feedback

3.4 Conclusion

While YAKE! looks very good on paper, the keyword extraction in real life does not work well with this type of text. The transcripts are either too noisy or unstructured to really allow YAKE! to perform well.

RAKE also did not perform well. The top extracted keywords often did not make sense and they were always very long. The reason for this is long keywords get preferred by RAKE, due to the way the relevance is calculated.

KeyBERT with the n-gram candidate extraction method did not convince. It takes the longest and the extracted keywords are grammatically incorrect, *e.g.*, *mary experience working*.

On the other hand, KeyBERT with the part-of-speech candidate extraction worked surprisingly well. The extracted keywords mostly make sense, and they are grammatically correct. To circumvent the issue with the long processing time, the Transcript Analyser project may be run on a GPU accelerated system. Sometimes the extracted keywords are very similar. To avoid this problem, KeyBERT implemented a way to diversify the keywords. This works well on large transcripts since there are many candidate keywords available, however on smaller transcripts this may result in results which do not make sense. A proposal for the Transcript Analyser project is to dynamically toggle this option when a certain length of transcript is reached. This will improve the keyword extraction even more.

4 Requirements

In order to proceed in a structured manner, we recorded user stories and use cases and iteratively refined them.

Implementing a new feature effectively, relies heavily on the requirements engineering process. It lays the foundation for the entire subsequent procedure. If the requirements engineering is executed properly, it will pay off in the future without requiring significant changes due to poor planning.

Additionally, the use cases and user stories were frequently discussed with the stakeholders of Interscriber, and their feedback was incorporated in order to enhance the feature.

4.1 Target Groups

Before creating user stories and use cases, the target groups must be identified. Different types of users interact with the highlight sentences.

The first target group consists of the **Interscriber users**. They will interact directly with the highlights feature user interface.

Because the highlight sentences can be sent by email, **people who missed the meeting** are the next target group. They will receive the highlights through the email.

The final target group consists of **Interscriber developers**. As there are numerous highlight sentences that have not yet been implemented, other developers may implement them.

4.2 User Stories

A user story is a general, informal explanation of a software feature written from the end user's perspective. Its purpose is to describe how a software feature will benefit the customer [24].

In the following user stories, we describe the goals of each target groups.

As an **Interscriber user**, I want to …

- see the most important highlights of a meeting/transcript, that I have not attended.
- see highlights that are relevant.
- see highlights, so that no relevant information is obscured.
- see the highlights sorted by importance.
- share the highlights with others.

As an **Interscriber developer** I want to …

- easily add new highlight sentences.
- be able to define the importance of sentences.

4.3 Use Cases

Use cases are used to identify and clarify system requirements. They are more detailed than user stories, allowing for the identification of any conflicts or missing components. Consequently, the use cases are based on the user stories from the preceding chapter. In the following sections, we provide a concise overview of the use cases to illustrate the tasks that must be completed. Detailed use case descriptions can be found in the Appendix chapter 15.2.

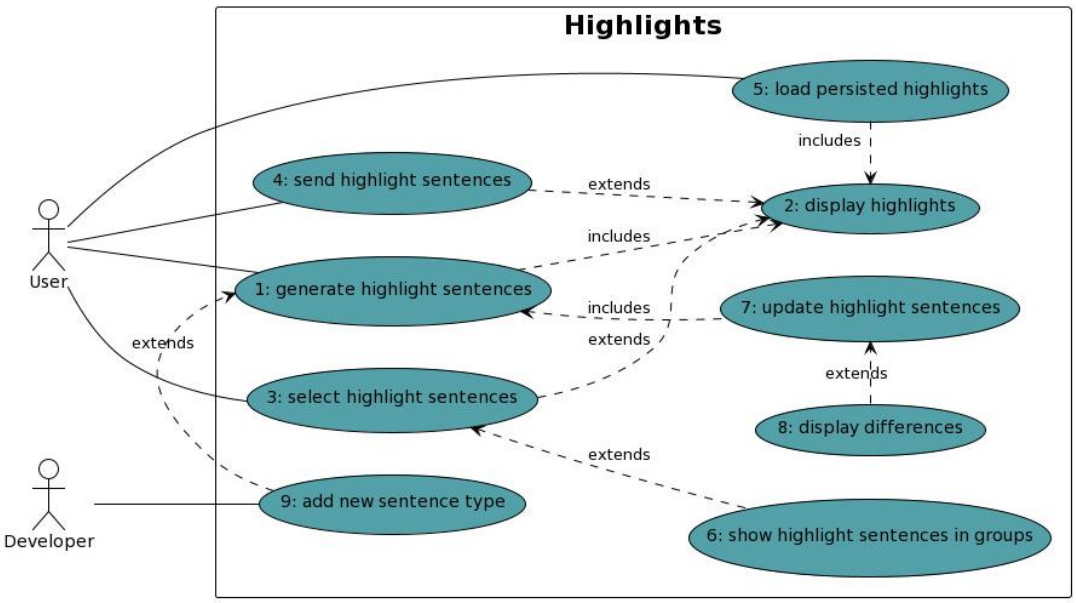


Figure 4: Highlights Use Case Diagram

1: Generate and Display Highlight Sentences

1. On the *transcript* page the user clicks on the *highlights* button and is redirected to the *overview* page.
2. There are no saved highlights. The system loads the sentence templates and analyses the transcript for missing template variables. All generated sentences and variables are saved.
3. Includes: 2: Display Highlights.

2: Display Highlights

- The system displays the most relevant sentences to the user, where the variable placeholders are replaced by the generated values.

3: Select Highlight Sentences

1. On the *overview* page the user clicks on the *edit* button.
2. The system displays the *edit* page to the user, which lists all generated sentences in order of importance.
3. Sentences can be un/selected if needed.
4. By clicking on the *finish* button, the user returns to the *overview* page.
5. The system saves the changes and shows the user the new selection of sentences.

Extends: 2: Display Highlights

4: Send Highlight Sentences

1. On the *overview* page the user clicks on the *send* button.
2. The system opens a new email in the default email program. The highlights are inserted as body.

Extends: 2: Display Highlights

5: Load persisted Highlights

1. On the *transcript* page the user clicks on the *highlights* button.
2. Previous generated highlights are found. The system loads both the sentence templates and the saved highlights.
3. Includes: 2: Display Highlights

6: Show Highlight Sentences in Groups

- On the *edit* page, the system shows the sentences by groups. The groups are formed according to sentence categories: “Speaker”, “Team” or “Topic”.

Extends: 3: Select Highlight Sentences

7: Update Highlight Sentences

1. On the *overview* page the user clicks on the *update highlights* button.
2. If the highlights are outdated, the system generates the highlights again, saves and displays them. Includes: 1: Generate and Display Highlight Sentences

8: Display Differences

- After the highlights are updated, the system shows the differences between the new and old version.

Extends: 7: Update Highlight Sentences

9: Add new Sentence Types

- The highlight feature in the backend provides an easy way for a developer to configure new sentences.

Extends: 1: Generate and Display Highlight Sentences

5 Frontend

The purpose of this chapter is to explain how and why certain UI design decisions have been made. As well as critically analyzing our solution and having an outlook on what can be improved.

5.1 Frontend Design

The UI design is a crucial step in the process of developing a new feature. It is likely that users will not use a feature which is not intuitive or aesthetically pleasing. Therefore, special care was taken to create high-quality UI mockups. In addition, it aids in validating use cases and identifying issues with them.

5.1.1 Recap: What the User Must be able to do

Before the design process could begin, it was necessary to determine the user's functional capabilities. The use cases assisted us with this process.

The Interscriber user must be able to …

- select and deselect highlight sentences, so that uninteresting sentences can be hidden.
- send highlight sentences via email.
- see changes in sentences after updating them (e.g., if the speaker names were changed, generated sentences change accordingly).

5.1.2 Initial Design

During the creation of the mockups, the progress and ideas were regularly discussed with the stakeholders of Interscriber. And ideas of them were incorporated accordingly. The final mockups exist of two pages.

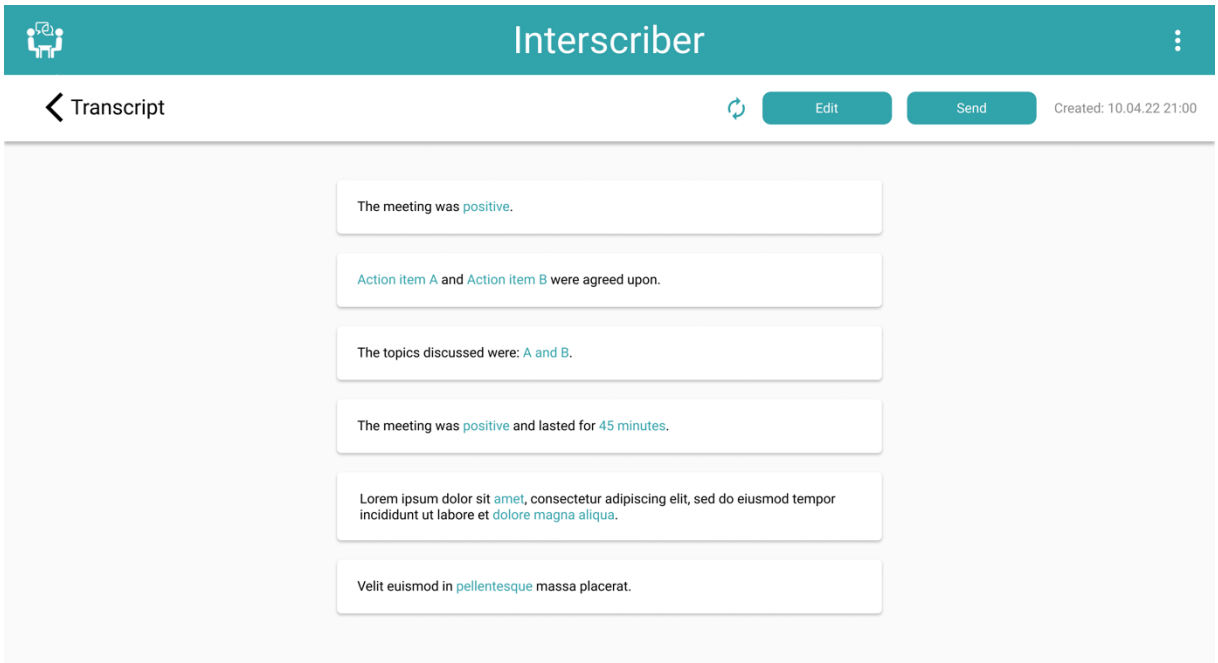


Figure 5: Interscriber highlights overview page mockup

The purpose of the **highlights overview page** is to present the user with the generated sentences as quickly as possible. If available, previously generated sentences are loaded from the database upon page load. If they are unavailable or outdated, the recreation process is started.

To draw attention to the important information of each sentence, the variables of the sentences (dynamic content, such as the sentiment result) are highlighted in "Interscriber blue".

In the upper right corner of the screen are action buttons. The first of the action buttons initiates the process of sentence regeneration if they are out of date.

Figure 6: Interscriber highlights overview page action button mockup

If the recreation process results in new or modified sentences, the changes will be displayed in dialog. This assists the user in understanding which sentences have changed, as some may be unselected. This is a particularly useful feature given that the Transcript Analyser project is frequently updated, which could lead to improved results.

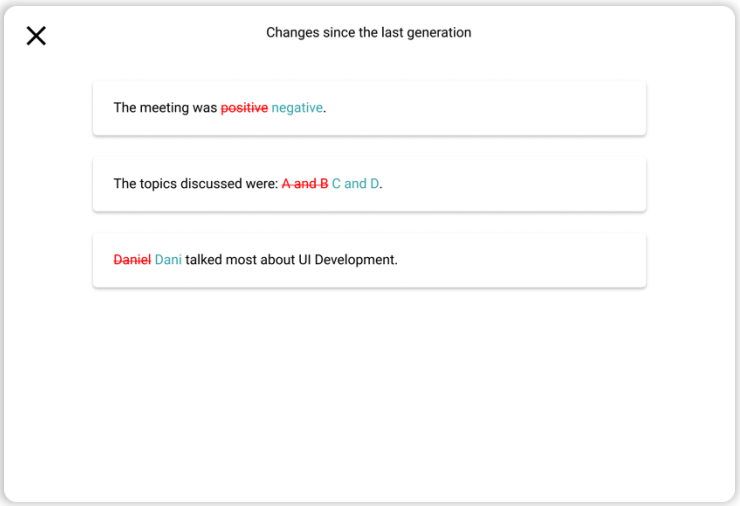


Figure 7: Interscriber differences dialog mockup

The **Send** action button launches the default email client of the operating system. It automatically inserts the highlight sentences into the body of the email. This choice is based on the fact that the recipients of the emails vary from meeting to meeting. In addition, it permits the user to add his own thoughts to the email, which could not be done if it were sent directly by Interscriber.

This method of sending the email prevents however the usage of custom html and css for the appearance.

The **Edit** action button redirects to the **edit highlights page**.

On the highlights edit page, the user can manage the selected sentences. Like on the overview page for highlights, the action buttons are located in the upper right corner of the screen.

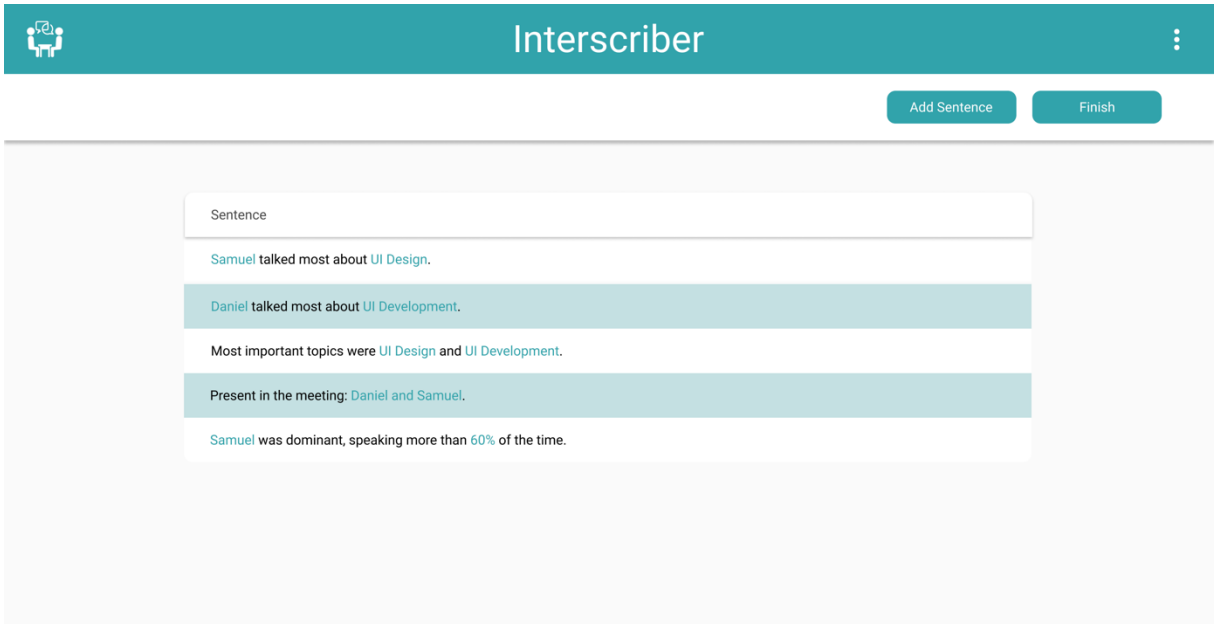


Figure 8: Interscriber edit highlights page mockup

The **Add Sentence** action button displays a dialog containing all generated sentences. Since there are numerous sentences, they are organized into categories. The category is sent by the backend for each sentence. This was a deliberate choice, as the frontend should be as decoupled from the backend as possible. Thus, the backend can be deployed independently from the frontend, allowing for greater future flexibility. However, the categories are fixed for now, due to limitations with translation.

All changes made on this page and dialog do not immediately trigger a backend request. Therefore, the operations can be combined into a single large update request. This reduces the number of requests the backend must process.

Once the user is satisfied with the changes, the backend is updated by clicking the **Finish** action button. The button also redirects back to the **highlights overview page.**

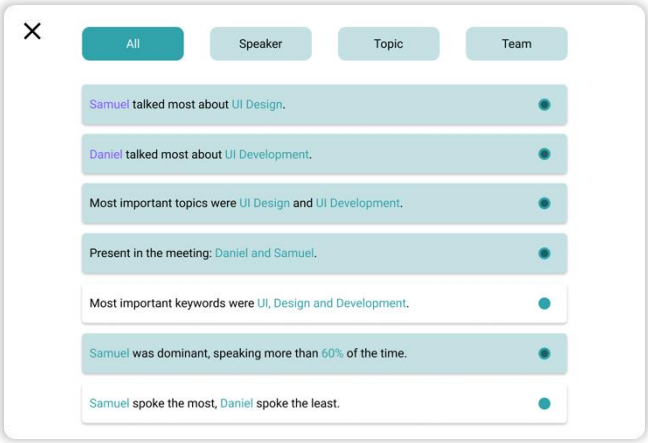


Figure 9: Interscriber add sentence dialog mockup

5.1.3 Evaluation of the Implemented UI

The highlights sentences feature was implemented, and the screens were designed according to the mockups. After deploying into a pre-production environment, user feedback was collected.

Based on this feedback (see Appendix chapter 15.1), the following improvements were compiled and implemented:

- Add instructions, to give guidance for first-time users.
- Edit highlights page is unnecessary.
- If changes are made in a dialog, it should be dismissed with an OK button rather than an X.
- In general, selecting sentences should require less clicks.

5.1.4 Changes for the Final Implementation

Based on the feedback, the design and user experience could be significantly improved. A help dialog was implemented to aid first-time users in understanding what highlighted sentences are and how to they use this feature.

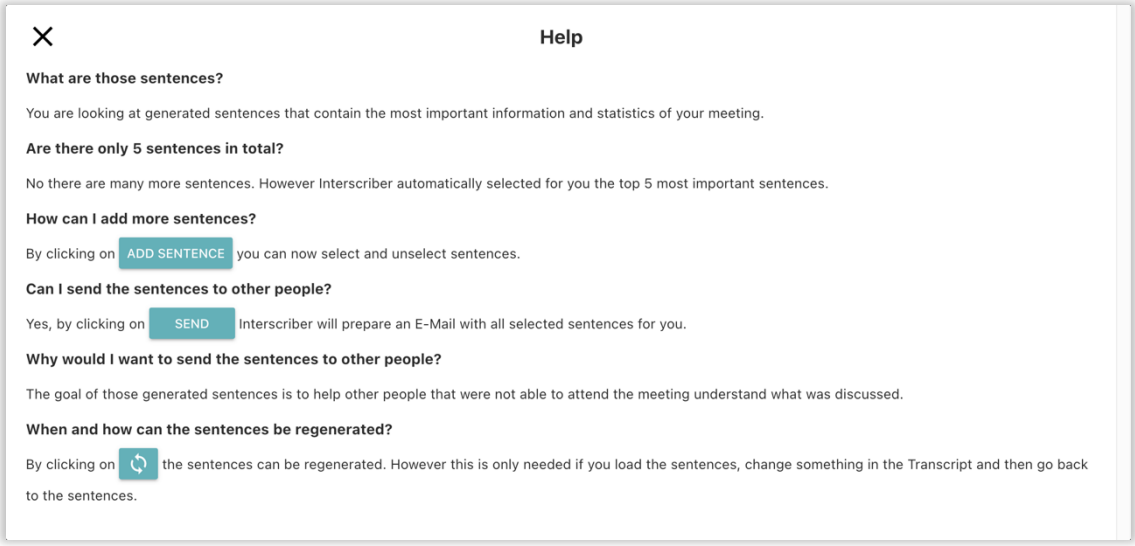


Figure 10: Interscriber help dialog

The help dialog is structured like a FAQ (Frequently Asked Questions) because it helps structuring the data, allowing users to easily locate the desired information. The dialog is automatically opened when the first set of highlights is created. If the user needs the dialog later again, it can be opened by clicking on the **question mark (?)** action button.

Figure 11: Interscriber overview action buttons

Since the goal of the user interface is to be as simple and intuitive as possible, the edit highlights page has been removed and the add sentences dialog has been moved to the overview page. This modification results in a significantly better user flow, with fewer clicks and a more intuitive action button layout.

The **X** has been replaced with a **Save** button in the add sentence dialog. This clarifies that when this button is clicked, an action with these modifications is performed.

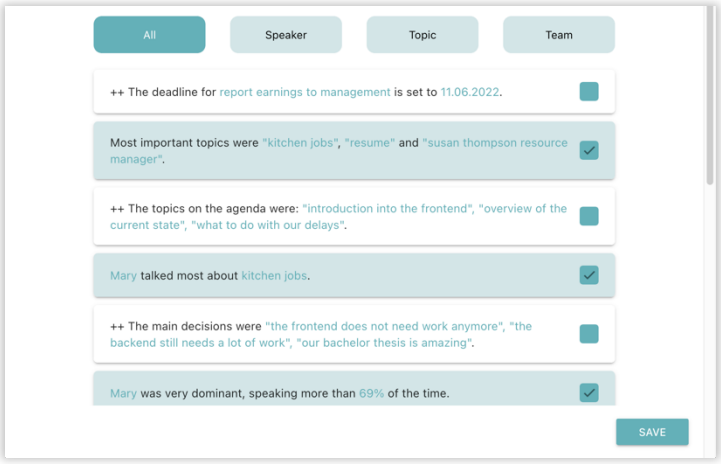


Figure 12: Interscriber add sentence dialog

6 Backend Architecture

The purpose of this chapter is to describe the architecture and explain how and why certain decisions have been made. As well as critically analyzing our solution and having an outlook on what can be improved.

6.1 Introduction

Important during the design phase is the creation of the architecture. A solid architecture lays the groundwork for an effective implementation. We begin by analyzing what has already been discussed and what can be extracted from that data.

To determine what is required from the backend, we examine the characteristics of the sentences, as defined in chapter 6.5, and how this helps our architectural design. The sentiment sentence type already provides hints:

*The sentiment of the **meeting** was **positive**.*

*The sentiment of the **first half** was **fairly positive**.*

*The sentiment of the **second half** was **very positive**.*

As can be seen, two parts of these sentences have changed. The dynamic data in these sentences is referred to as **Variable**. The first **Variable** is the section of the meeting from which the sentiment was extracted from.

In the first sentence, the sentiment was calculated based on the entire meeting, whereas in the second sentence, only the first half was considered and in the last sentence only the second half.

However, for all these sentences only one sentence type was configured:

*The sentiment of the **$SENTIMENT_PART** was **$SENTIMENT**.*

This is made possible by the **Configurations** or short **Configs**. Configurations are added statically to the configuration file, where sentences are defined. The sentimental sentence has three configurations:

First half

Second half

Full meeting

This configuration will result in the three sentences described above. If a new section of the meeting is to be analyzed, the configuration file must be extended, and the backend must be redeployed.

The most important topic per speaker sentence type provides additional insight into what is required:

> ***Klaus*** *talked most about* ***Topic1****.*
>
> ***Hans*** *talked most about* ***Topic2****.*

As we already know, Klaus & Hans and Topic1 & Topic2 are **Variables.**

However, this sentence type cannot be configured statically in the configuration file since each meeting has a unique set of attendees.

Configs must therefore be created at runtime, more accurately during the sentence generation. This is where the **DynamicConfigs** are used.

And analyzing following sentence, will give us the last insight into what is needed:

> ***Klaus*** *was very dominant, speaking more than* ***60%*** *of the time.*

This sentence is only relevant if a specific speaker stands out based on the amount that they spoke. A **Filter** can be configured to the sentence type to automatically hide such sentences. Utilizing a configured **Comparator**, a **Filter** compares the **Variable** to a configured value.

In the case of this sentence type, a configured value of 50 and the **Comparator** LARGER_THAN is configured.

Then after the sentences were generated, the 60% of the **Variable** is compared to 50% from the **Configuration**. If the **Variable** value is greater than the threshold, the sentence is displayed; otherwise, it is hidden.

6.2 Functions provided by Backend

We implemented the so-called **HighlightService**. This service is decoupled from all other service within Interscriber.

The **HighlightService** must be able to ...

- generate sentences, based on templates persisted in a configuration file.
- retrieve previously generated highlight sentences from the database.
- modify the sentence selection state.

6.3 Structure of Highlight Service

Using the fundamental concepts from the preceding chapters, we can explain the implementation details of the **HighlightService**.

Figure 13 is a simplified component diagram of the actual implementation. It conceals simple data objects and the concrete implementations of **DynamicConfig** and the **Variables**.

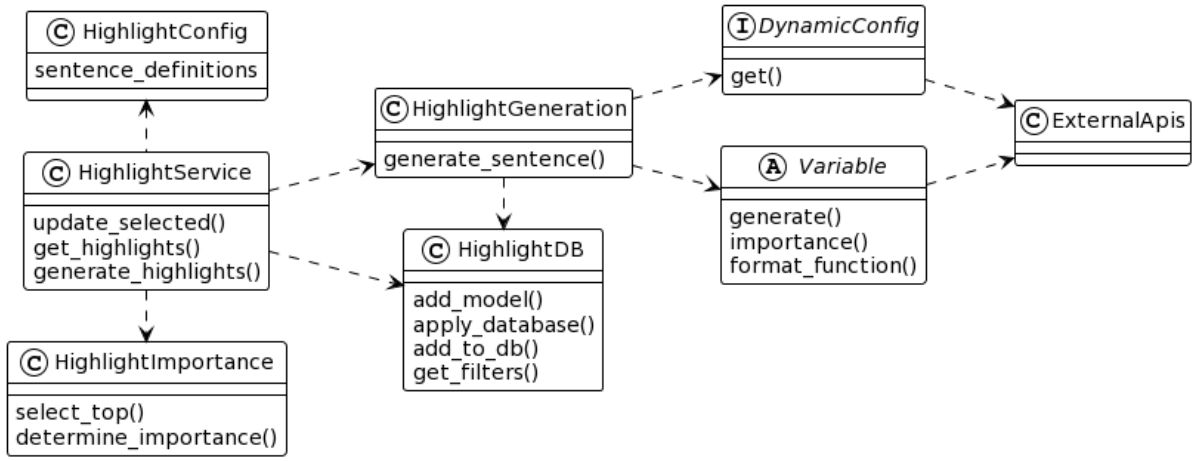


Figure 13: Highlights Component Diagram (simplified)

6.3.1 HighlightService

The **HighlightService** component is responsible for coordinating the creation of each sentence and associated variables. It can retrieve previously generated highlights from the database and transform them into data objects. In addition, it can update the selection state of the sentences in the database.

6.3.2 HighlightConfig

The **HighlightConfig** component is responsible for loading the configuration. During application startup, the configuration file is automatically parsed and converted into data objects. This is possible because the configuration file is written in yaml.

6.3.3 HighlightImportance

The **HighlightImportance** component is responsible for calculating the importance of a generated sentence. This is explained in greater detail in the chapter 6.6.

If the highlight sentences have never been generated before, the **HighlightImportance** component is responsible for selecting the five most relevant sentences based on the previously calculated score.

6.3.4 HighlightGeneration

The **HighlightGeneration** component contains the logic to generate one sentence. It generates a list of **Configurations** for each sentence; if none exist, the default **Configuration** is used. Then, it generates the **Variable** values and importance for each **Configuration**.

This means that every sentence includes a list of **Configurations**. And each **Configuration** includes a list of **Variables**.

6.3.5 HighlightDB

The **HighlightDB** component contains the sole database access point. Its primary purpose is to add sentences to the database.

In addition, if previous results are detected following the generation of highlight sentences, the HighlightDB component compares the old and new values. It can detect any differences and notifies the frontend of any changes.

6.3.6 Variable

As explained in the chapter 6.1, the **Variable** represents a single dynamic element embedded within a sentence. Therefore, it needs access to the **Configuration** parameters and the transcript, as well as the ability to invoke external APIs. This allows the **Variable** to collect all the information required to determine the returned word.

Additionally, it calculates the importance of the generated value.

6.3.7 DynamicConfig

The **DynamicConfig** is a concept of generating a list of **Configurations** prior to the start of sentence generation.

It can extract data from the transcript or invoke an API. It can then create the Configs and inject the gathered data into them, allowing each Variable to access this data.

6.3.8 ExternalApis

The **ExternalApis** are helper components, simplifying the API requests for **Variables** and **DynamicConfig**.

6.4 Highlight Generation

To help understand how the **HighlightService** functions, the sequence diagram in Figure 13 was made. It describes the generation of the highlight sentences, the most complicated aspect of the HighlightService.

Figure 14: Highlights Generation Sequence Diagram is a sequence diagram giving a high-level explanation of how the HighlightService operates.

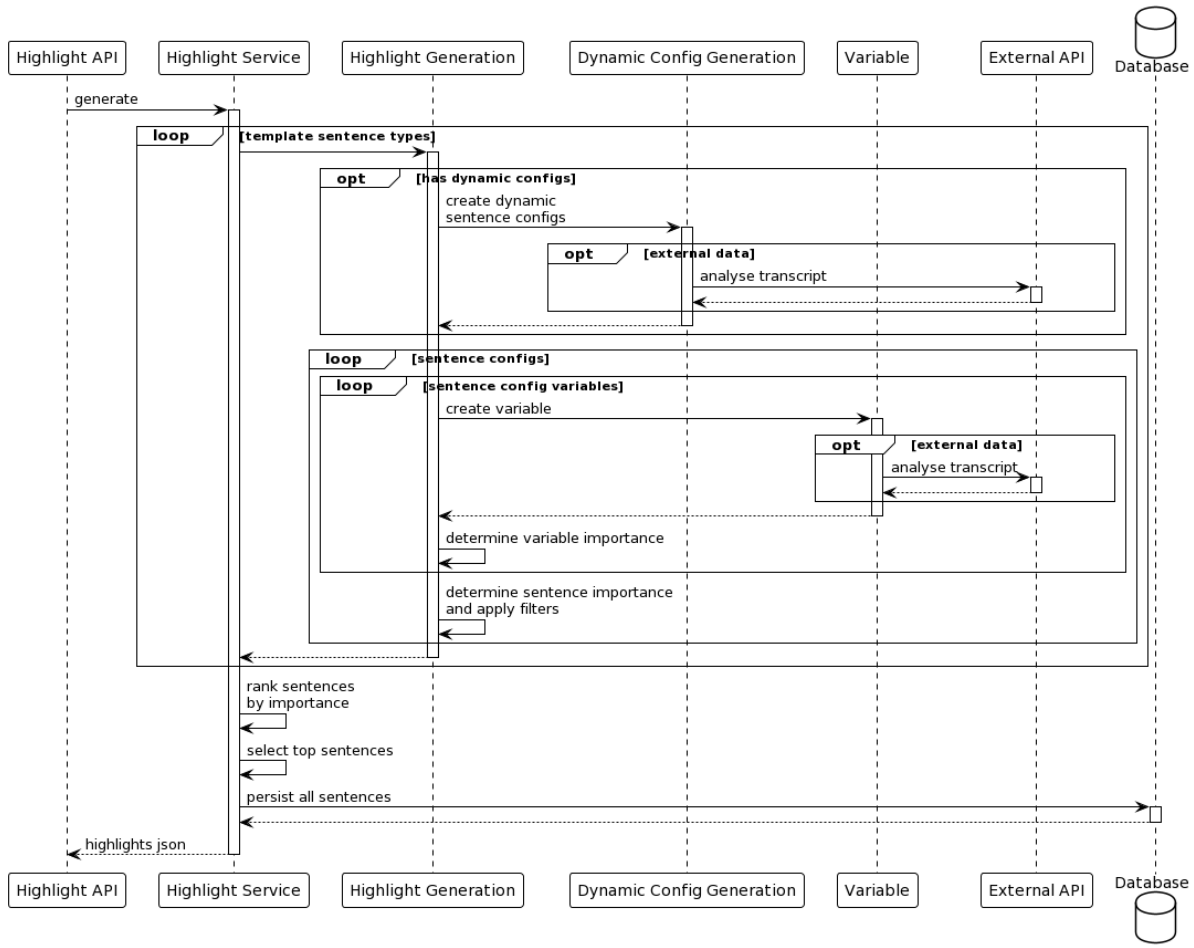


Figure 14: Highlights Generation Sequence Diagram

1. Load the sentences from the **HighlightConfig** component.
2. For every sentence type the generation within the **HighlightGeneration** component is triggered.
    a. If a **DynamicConfig** was configured, its **Configuration** list is loaded.
3. If no **Configurations** are present, the default implementation is used.
4. For all **Configurations** the **Variable** values are generated, and their importance calculated. The **Variable** may call an external API to fetch data.
5. Once each **Variable** is generated, the importance of each sentence is calculated; the method for calculating this score is described in the chapter 6.6.
6. If a sentence type filter was configured, it is applied. This might hide sentences.
7. The sentences are ranked according to their relative importance.
8. If this is the first time the highlights have been generated, the top five highlights are selected.
9. The highlights are persisted in the database.

6.5 Sentence Configuration

In our first attempt, we created a class for each sentence and variable type, each holding both static and dynamic information. This resulted in an object-oriented design, having the objects create themselves. The implementations mostly held static information. We quickly realize it would be easier for a developer to change the configuration of sentences if it is detached from the implementation.

For this reason, the configuration was moved into a YAML file called *sentences.yml*. The file is loaded during the startup of the Interscriber application. The resulting list of sentence definitions is imported as a static variable into the HighlightService.

The structure of such a sentence configuration is shown in the following object diagram. A more detailed description is found in the chapter 7.2.1 of the technical documentation.

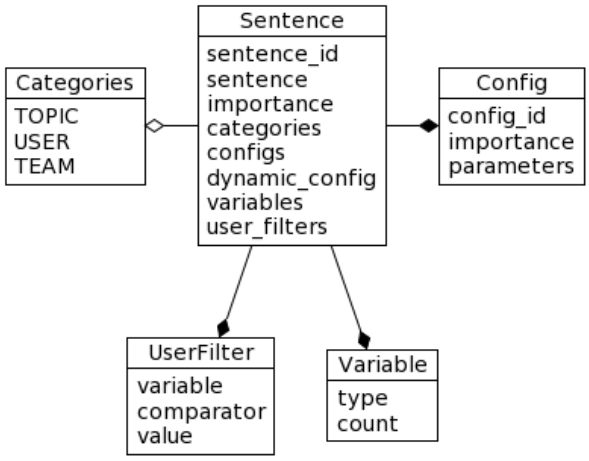


Figure 15: Sentence Definition File Structure

The configuration contains the data for the previously presented concepts (as seen in chapter 6.1). The following section explains how they are implemented within the configuration.

configs is a collection of **Configuration** objects. They are represented by a unique id, a list of arbitrary parameters, and the importance of the configuration in question.

dynamic_config connects the configuration to the **DynamicConfig** object. As each DynamicConfig has a unique key, this key can be used to store it within this configuration.

variables is a collection of **Variable** objects. They are represented by the unique id of the **Variable** and an optional count parameter, which is by default set to one.

6.6 Sentence Ranking

For each meeting, numerous highlight sentences are created. The dynamic values differ between meetings. The objective is to arrange the sentences in descending order of importance, with the most important appearing first. To achieve this, an importance for each resulting sentence is calculated. The importance is a number between zero and one, one being extremely important and zero being unimportant. The importance should vary depending on the sentence type and take the influence of the **Variable** sentence parts into account.

The configuration file defines a base importance for each sentence type. This base importance may be overridden by the **Configuration** or **DynamicConfiguration** configurations.

Each **Variable** provides an importance factor which is multiplied with the base importance. The importance factor gets calculated from the value of the **Variable**, the parameters from the **Configuration**, and the confidence of that value. The **external APIs** return the confidence as a value between zero and one indicating the probability that the value of the variable is correct.

$$\text{importance}_{\text{sentence}} = importance_{base} * \prod_{i=1}^{|config_{vars}|} importance_{var_i}$$

$$importance_{base} = \begin{cases} importance_{config}, & importance_{config}\ is\ defined \\ importance_{sentence\ type}, & importance_{config}\ is\ not\ defined \end{cases}$$

$imp = importance$
$var = variable$

Figure 16: Calculation of the Sentence Importance

A good example for a sentence type that uses all the importance features is the sentiment sentence ranking:

> *The sentiment of the **first part** was **quite negative**.*
> *The sentiment of the **second part** was **very positive**.*
> *The sentiment of the **meeting** was **fairly positive**.*

The configurations for this sentence type are static. Thus, the sentence referring to the entire meeting can be defined as more important than the sentence referring to only half of the meeting.

In addition, the second variable that defines the sentiment calculates its importance factor based on the value and its confidence. The importance factor increases with decreasing sentiment. This was done because it generally is expected to be a positive meeting.

6.7 Database Model

This ERM (Entity-Relationship Model) shows in what structure the generated highlight sentences are persisted in the database. For clarity, only the relevant entities and entity fields are shown. With our work we added the highlight entities, both the user and the transcript entities already existed before.

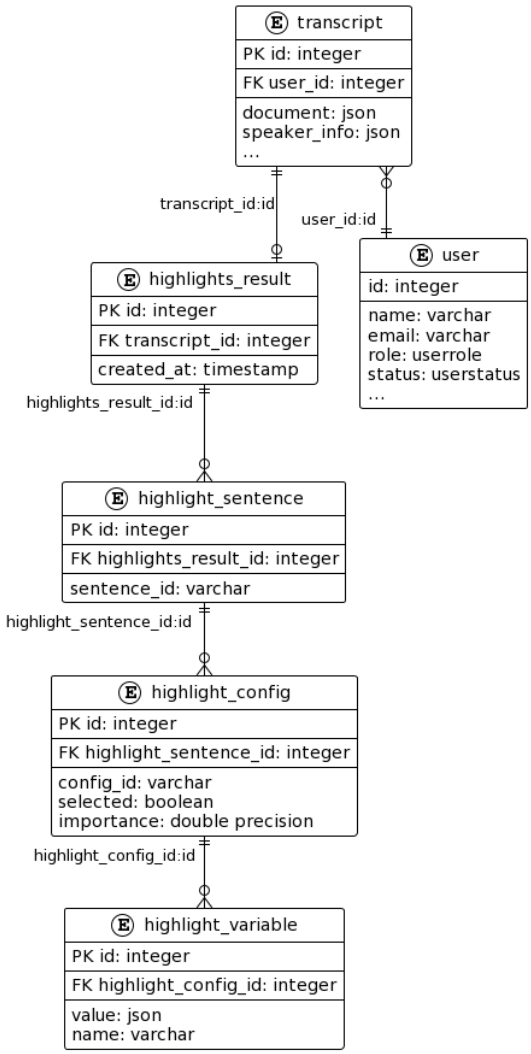


Figure 17: Highlights ERM Diagram (simplified)

Each user can have multiple transcripts. Each transcript has exactly one highlights result. Only the dynamic parts of the sentences are persisted in the database. This implies that for being able to display the complete sentences to the user, the static information from the sentence configuration file must be present as well.

The sentence entity refers to the sentence type defined in the configuration file. Each sentence has a **Config**, that represents an actual sentence with its characteristics and each **Config** has its **Variables** which represents the dynamic data of the sentence.

6.8 Rest Endpoints

We created three REST endpoints to connect the frontend with the **HighlightService**. What it does and how it works is described in the following section.

6.8.1 Get Highlights

This endpoint invokes the get method of the **HighlightService**. It retrieves the previously generated highlights from the database. If they do not exist, the frontend receives a 404 error.

The stale parameter is returned to indicate if the highlights are out-of-date. This parameter compares the date of the most recent generation of the highlights with following values:

- the date of the last transcript change
- and the date of the last configuration file change

If any of these values are more recent than the last generation, stale is set to true, instructing the frontend to regenerate the highlights.

6.8.2 Generate Highlights

This endpoint invokes the generate method of the **HighlightService**. It generates all sentence configurations by extracting data from the selected transcript. The generation occurs whether the highlights are out of date. Once the generation is complete, the newly created values are compared to previously generated values that have been stored in the database. If any variable changed, the database sentence is updated, and the frontend is notified.

6.8.3 Update Selected

This endpoint invokes the update method of the HighlightService. It receives a list of sentences and their respective user selection state as a parameter. The database is then updated accordingly.

6.9 Enhancements

During the development phase of this project, we identified and noted several enhancements. In general, they enhance the architecture, which is why they are listed here.

6.9.1 Parallel Sentence Generation

All sentences are currently generated sequentially. One possibility to improve this performance issue is to generate the sentences simultaneously in parallel threads.

6.9.2 Caching External APIs

Caching the results of external APIs is another technique for improving performance. Even though the only external API we currently support (Transcript Analyser) has its own caching system, it would be preferable to implement one in Interscriber as well. It would significantly reduce the number of API calls and permit the integration of new external APIs that do not cache themselves.

6.9.3 Change the Sentence Configuration File without Deployment

With our current architecture, the sentence templates and variables are statically defined by developers within the sentences.yml file. By enabling the upload of a new sentences.yml file without deploying the whole application, the process would be improved greatly, allowing new sentences to be added, without the need of redeploying the whole application.

6.9.4 Detection of External API Updates

We always want to show the most recent highlight sentences. However, they should not be regenerated every time the user reloads the webpage. Currently, we detect changes within the transcript and the sentences.yml file.

However, it would be beneficial to have a similar system for detecting changes in the external APIs. Because a new version of the API could mean an improved NLP algorithm, which could potentially extract more accurate information.

7 Technical Documentation

One of our goals is to make it simple for future developers to extend the Highlights Feature. This technical documentation is intended for such developers and assumes that they already possess fundamental technological skills and knowhow.

The documentation is divided into four sections. First, it introduces the technologies of Interscriber. Then, a detailed description of the highlight service explains its inner workings and how a sentence is configured. Further, an example how to add a new sentence is provided. The chapter concludes with suggestions for technical enhancements.

7.1 Technologies

The Interscriber webapp consists of a frontend (`src/frontend/`) and a backend (`src/backend/`). The frontend is implemented in Vue2, with vue-router for routing and vuex as global data store. The design is based on the Google Material Design (`vue-materialze`).

The backend is implemented in python, as a dependency management tool poetry is used. The python app is built with flask, a powerful backend framework to easily build REST APIs.
The data persistence layer of Interscriber is a postgres database. With the SQLAlchemy library the backend has a powerful ORM (Object–relational mapping) to connect and manage the database.

To deploy the webapp, Docker with docker-compose is used. During the deployment process, the frontend is compiled into a html, js and css bundle. It is then copied into the backend folder, where the backend application is started. The backend flask app then serves the APIs as well as the frontend to the web.

7.2 Highlight Service

The highlight service can generate preconfigured sentences from templates. Those templates are configured in the `config/sentences.yml` file. The highlight service can extract and automatically create all the sentences. If a new sentence was added and there is an issue, it will create an error during the launch of the application.

7.2.1 Sentence Configuration

The configuration design is constructed as follows: One sentence object represents a single sentence type. Multiple sentences can be generated from a single sentence type. More information about how one sentence type can generate multiple sentences can be found in the chapter Configs [OPTIONAL].

Example
This is an example from Interscriber. It is the configuration of the sentiment sentence type and results in the following sentences:

- The sentiment of the first half was quite positive.
- The sentiment of the second half was fairly negative.
- The sentiment of the meeting was fairly positive.

As can be seen, in this case, one sentence type generates three sentences. The following sections explain how and why this works.

All keys except *dynamic_config* are present in this configuration. For more information, see the chapter Dynamic Config [OPTIONAL]. The *user_filters* configuration does not work because the EQUAL comparator does not exist.

```yaml
- sentence: 'The sentiment of the $SENTIMENT_PART was $SENTIMENT.'
  sentence_id: SENTIMENT_PART
  importance: 0.5
  categories:
    - TEAM
  variables:
    - type: SENTIMENT_PART
    - type: SENTIMENT
  configs:
    - parameters:
        start: 0
        end: 0.5
      importance: 0.25
      config_id: FIRST_PART
    - parameters:
        start: 0.5
        end: 1
      importance: 0.25
      config_id: SECOND_PART
    - parameters:
        start: 0
        end: 1
      config_id: FULL_MEETING
  user_filters:
    - variable: SENTIMENT
      comparator: EQUAL
      value: negative
```

General sentence data

services/highlights/models/definitions/sentence.py

```yaml
- sentence: 'The sentiment of the $SENTIMENT_PART was $SENTIMENT.'
  sentence_id: SENTIMENT_PART
  importance: 0.5
```

sentence [String]: To begin, the sentence itself must be written. As can be seen, the sentence contains the variable names prefixed with the $ symbol (dollar sign). This syntax is required so that the frontend can replace the variable name with the actual value.

sentence_id [unique String]: This is a unique sentence identifier. It must be unique between all sentences in this configuration file.

importance [float 0-1]: The importance of the sentence type in relation to other sentence types. For example, is the sentiment sentence type more or less important than a sentence type that captures fights within the meeting?

Categories

services/highlights/models/definitions/category.py

```
categories:
  - TEAM
```

categories [Array]: This is an array of the enum *category*.

```
TOPIC
TEAM
USER
```

The categories array is mainly used in the frontend to display the sentences with the same category together.

Variables

services/highlights/models/definitions/variable.py

```
variables:
  - type: SENTIMENT_PART
  - type: SENTIMENT
```

variables [Array]: The variables configuration node is an array which can contain following variable types:

```
- type: SENTIMENT_PART
  count: 3 # (default: 1)
```

The count parameter, for example, allows the *MOST_IMPORTANT_TOPICS* variable to specify the number of topics to return. As a result, the variable can be used in both sentences:

- Most important topics were "frontend code", "backend architecture" and "design".
- Speaker 2 talked most about design.

Both of those sentences utilize the same variable, but one passes as additional parameter: *count: 3* and the other *count: 1*.

Implementation in python

Since the variables contain additional logic, they must be implemented in python. The mapping between the types and the implementation happens in this file:

services/highlights/models/definitions/variable_types.py.

```
class VariableType(enum.Enum):
    SENTIMENT_PART = SentimentPartVariable
```

The implementation itself extends from the *variable_interface.py*.

```
class Variable:
    @staticmethod
    def importance(value, confidence, parameters, transcript: Transcript):
        return confidence

    @staticmethod
    def generate(transcript: Transcript, parameters, count) -> (any, int):
        """This retrieves the data from api and maps it to the value"""

    @staticmethod
    def format_function() -> Callable[[any], str]:
        return lambda value: value
```

`importance() -> float[0-1]`: The importance method returns a number between 0 and 1.

Example: The sentiment sentence is more relevant to the user if the meeting was overall negative than positive. This means the importance method returns a higher number when the value is negative.

`generate()`: The generate method is required to be implemented. By default, the method returns a tuple of the value and the confidence of the value. However, it can also return an array of tuples (see `MOST_IMPORTANT_TOPICS` variable).

Note: If an array is returned the highlight service extends the variable names with the index of the array; `$MOST_IMPORTANT_TOPIC_0`, `$MOST_IMPORTANT_TOPIC_1`

`format_function()`: The format function is required because, in some cases, the value returned by generate must remain in a comparable format so that the importance can be calculated.

After the evaluation, format function will format the value, such as converting a datetime to a string or a number to a percentage.

Configs [OPTIONAL]

`services/highlights/models/definitions/sentence_config.py`

```yaml
configs:
  - parameters:
      start: 0
      end: 0.5
    importance: 0.25
    config_id: FIRST_PART
  - parameters:
      start: 0.5
      end: 1
    importance: 0.25
    config_id: SECOND_PART
  - parameters:
      start: 0
      end: 1
    importance: 0.5
    config_id: FULL_MEETING
```

`configs [Array]`: The configs node is an array consisting of following nodes:

```yaml
- parameters:
    start: 0
    end: 0.5
  importance: 0.25
  config_id: FIRST_PART
```

`parameters [Array]`: The node parameters can contain arbitrary key value pairs. Once the file is loaded into python the parameters are loaded into a dict. A variable can access the parameters, since the dict is passed into the generate and importance function.

`importance [float 0-1]`: The importance indicates how important one single configuration is.

Example: The sentiment over the full meeting is more relevant and important than the sentiment over only half.

`config_id [String]`: A unique identifier of the config. Needs to be unique within the config node.

Dynamic Config [OPTIONAL]

services/highlights/models/definitions/dynamic_config_types.py

```
dynamic_config: SPEAKER
```

dynamic_config: [Enum]: See chapter 7.2.3 for the list of available dynamic configurations.

Implementation in python

Since the dynamic_config contain logic, they must be implemented in python. The mapping between dynamic_config and implementations is found here:

services/highlights/models/definitions/dynamic_config_types.py.

```
class DynamicConfigType(enum.Enum):
    SPEAKER = SpeakerConfig
```

All dynamic_config implementations must extend from the *dynamic_config_interface.py*.

```
class DynamicConfig:
    @staticmethod
    def get(transcript: Transcript):  # pylint: disable=unused-argument
        return []
```

get(): The get method can create *SentenceConfigs* (see chapter Configs [OPTIONAL]).

User Filters [OPTIONAL]

services/highlights/models/definitions/user_filter.py

```
user_filters:
  - variable: MOST_DOMINANT_SPEAKER_PERCENTAGE
    comparator: SPEAKER_PERCENT_LARGER_THAN_AVERAGE_REST
    value: 200
```

variable [String]: As a value for this parameter, all variable of this sentence type can be used.

comparator [Enum]: The comparator defines how the variable is compared to the value.

services/highlights/models/definitions/comparator.py

```
LARGER_THAN = value1 > value2
SMALLER_THAN = value1 < value2
SPEAKER_PERCENT_LARGER_THAN_AVERAGE_REST =\
  how much more a speaker had to speak in comparison to others to be a dominant speaker
```

value [any - must match with variable value]: The value with which the variable value is compared to.

7.2.2 Available Variables

services/highlights/models/definitions/variable_types.py

All variable implementations can be found here: *services/highlights/variables/*

```
class VariableType(enum.Enum):
    LEAST_DOMINANT_SPEAKER = LeastDominantSpeakerVariable
    MOST_DOMINANT_SPEAKER = MostDominantSpeakerVariable
    MOST_DOMINANT_SPEAKER_PERCENTAGE = MostDominantSpeakerPercentageVariable
    MOST_IMPORTANT_TOPIC = MostImportantTopicVariable
    SENTIMENT_PART = SentimentPartVariable
    SENTIMENT = SentimentVariable
    SPEAKER_NAME = SpeakerNameVariable
    SPEAKERS = SpeakersVariable
    MOST_CONTRIBUTED_TOPIC = MostContributedTopicVariable
    TITLE_FROM_CONFIG = TitleFromConfigVariable
    DATE_FROM_CONFIG = DateFromConfigVariable
    DECISIONS = DecisionsVariable
    AGENDA_TOPICS = AgendaTopicsVariable
```

LEAST_DOMINANT_SPEAKER: Invokes the Transcript Analyser statistics API, which returns the speaker times and id. It then finds the smallest amount and gets with the speaker id the name. The speaker name is then returned.

MOST_DOMINANT_SPEAKER: Invokes the Transcript Analyser statistics API, which returns the speaker times and id. It then finds the largest amount and gets with the speaker id the name. The speaker name is then returned.

MOST_DOMINANT_SPEAKER_PERCENTAGE: Invokes the Transcript Analyser statistics API, which returns the speaker times and id. It calculates the percentage of the most dominant speaker: *100 / sum(speakin times) * max(speaking times)*.

MOST_IMPORTANT_TOPIC: Invokes the Transcript Analyser statistics API. The API returns the top 3 topics together with the speaker times. This variable can consume the *count* parameter from the variable config node (default=1). The count parameter defines how many values are returned from this variable. Additionally, the *speaker_id* in the parameter dict which is set by the *config/dynamic_config* can also be used (optional). The *speaker_id* parameter allows the API to only extract topics about this specific user.

SENTIMENT_PART: This variable must be used with either *config* or *dynamic_config* which must define the *start* and *end* parameters. The variable itself converts the start and end from numbers into words, which can be used in sentences.

```
0 - 1:      'meeting'
0 - 0.5:    'first half'
0.5 - 1:    'second half'
0.3 - 0.9:  'part between 30% and 90%'
```

SENTIMENT: Invokes the Transcript Analyser sentiment API. From the API the sentiment of each sentence is returned. Additionally, this variable can use the *start* and *end* parameter set by *config* or *dynamic_config* to only analyse a certain part of the meeting. Based on the result the general positiveness is calculated and returned. The confidence of the sentiment is also included into the calculation of the return value.

```
value < 0.3:   fairly positive/negative
value < 0.6:   quite positive/negative
value < 1:     very positive/negative
```

`SPEAKER_NAME`: The speaker name variable converts the `speaker_id` from the parameter dict which is set by `config` or `dynamic_config` into the speaker name. The speaker name is found in the transcript itself.

`SPEAKERS`: The speakers variable returns a string of all speaker names.

`MOST_CONTRIBUTED_TOPIC (Currently mocked)`: This variable should return the topic, where most speakers contributed something to.

`TITLE_FROM_CONFIG`: This variable is usually used in combination with a `dynamic_config`. When the `dynamic_config` invokes an NLP endpoint, the return value can be added to the `title` key in the parameter dict. The `TITLE_FROM_CONFIG` variable then retrieves and returns the `title` value.

`DATE_FROM_CONFIG`: This variable is usually used in combination with a `dynamic_config`. If there are any dates returned from the endpoint, it can be assigned to the `date` key in parameters. The `DATE_FROM_CONFIG` variable then calculates the importance from the proximity of the date - the closer the date to more important. In the end, the variable converts the date into an iso-format, which the frontend can understand.

`DECISIONS (currently mocked)`: This variable should return the main decision taken in the meeting.

`AGENDA_TOPICS (currently mocked)`: This variable should return the agenda points of the meeting.

7.2.3 Dynamic Configurations

When to use Dynamic Configurations

Dynamic configurations are used when the number of sentences that should be created for a specific sentence type varies. For instance, if the requirement is to generate sentences that highlight the user's most-discussed topic, dynamic configurations are used to implement this. This is because it is unknown at the time of sentence configuration who will attend the meeting. And therefore, it is impossible to manually create sentence configs as described in the chapter Configs [OPTIONAL]. Conclusion: It must be possible to create configurations during runtime.

What it does

A dynamic configuration generates an array of `SentenceConfig` (see chapter Configs [OPTIONAL]).

Available Dynamic Configurations

`services/highlights/models/definitions/dynamic_config_types.py`

All `dynamic_config` implementations can be found here: `services/highlights/dynamic_configs/`

```
class DynamicConfigType(enum.Enum):
    SPEAKER = SpeakerConfig
    EVENTS = Events
    ACTIONS_WITH_DEADLINE = ActionsWithDeadline
    SPEAKER_ACTIONS = SpeakerActions
    CRITICAL_TOPICS = CriticalTopics
    MEETING_FIGHTS = MeetingFights
    NONE = DynamicConfig
```

`SPEAKER`: This dynamic configuration generates a unique configuration for each speaker in the meeting. It adds the `speaker_id` into the parameter dict. This can be used with:

- `MOST_IMPORTANT_TOPIC`
- `SPEAKER_NAME`

`NONE`: This is the default dynamic config. It returns an empty array.

`EVENTS (currently mocked)`: This dynamic configuration fetches from a mocked API. The return values are added to the parameters dict (`title` and `date`).

`ACTIONS_WITH_DEADLINE (currently mocked)`: This dynamic config fetches a mocked API. The return values are added to the parameters dict (`title` and `date`).

`CRITICAL_TOPICS (currently mocked)`: This dynamic config fetches a mocked API. The return values are added to the parameters dict (`title` and `speaker_id`).

`MEETING_FIGHTS (currently mocked)`: This dynamic config returns an empty array. This should demonstrate that empty configurations will result in no generated sentence.

7.2.4 API Endpoints

If the transcript language is not English, all endpoints return a 400 error response code. Because the highlight service and Transcript Analyser APIs only support English transcripts at this time.

Update

The update endpoint updates the `selected` state of the sentences, which are passed in the request.

Example URL: `/highlights/update?transcript_id=1`

URL: `/highlights/update`
Method: `POST`
URL parameter: `transcript_id`
Request body: An array of the sentence id and the selection state.

```
[
  {
    "id": "sentence id",
    "selected": "true/false"
  }
]
```

Response: Empty response
Response codes:

- 200: successful
- 400: transcript language is not english
- 404: no highlight sentences for this transcript_id were found in the database

GET
The get endpoint loads all the previously generated sentences from the database and returns it.

Example URL: `/highlights/get?transcript_id=1`

URL: `/highlights/get`
Method: `GET`
URL parameter: `transcript_id`
Response: The response is an object of following values

```
{
  "stale": "true/false",
  "createdAt": "last update in ISO 8601 format",
  "highlights": [
    {
      "id": "unique sentence id (SENTENCE_ID%CONFIG_ID)",
      "sentence": "template sentence",
      "categories": ["CATEGORIES"],
      "importance": "0-1",
      "selected": "true/false",
      "variables": [
        {
          "name": "variable name",
          "value": "variable value"
        }
      ]
    }
  ]
}
```

`stale`: whether the data from the database is outdated or not
`createdAt`: the last time the sentences were created
`highlights`: all the previously generated sentences from the database

Response codes:

- 200: successful
- 400: transcript language is not english
- 404: no highlight sentences for this `transcript_id` were found in the database

Generate
The generate endpoint creates all sentences which are defined in the `sentences.yml`. It then compares them to the previously generated values and updates them if necessary.

Example URL: `/highlights/generate?transcript_id=1`

URL: `/highlights/generate`
Method: `GET`
URL parameter: `transcript_id`
Response: The response object is very similar to the one from the get request.

```
{
  "createdAt": "last update in ISO 8601 format",
  "highlights": [
    {
      "id": "unique sentence id (SENTENCE_ID%CONFIG_ID)",
      "sentence": "template sentence",
      "categories": ["CATEGORY"],
      "importance": "0-1",
      "selected": "true/false",
      "is_new": "(OPTIONAL) true/false",
      "variables": [
        {
          "name": "variable name",
          "value": "variable value",
          "old_value": "(OPTIONAL) old value"
        }
      ]
    }
  ]
}
```

The differences are:

- stale, since the data cannot be outdated immediately after generation
- a sentence object contains an is_new value, indicating that the sentence was never generated before
- a variable object contains an old_value value, indicating that since last creation the value changed

Response codes:

- 200: successful
- 400: transcript language is not English

7.3 Example: How to add a new Sentence

In this example, we look at how *Klaus talked most about his barking dog*. would be implemented. This sentence consists of a speaker name and a topic. However, it might be interesting to know what other speakers than Klaus said. Hence, this sentence should be created for each meeting attendee.

To achieve this a dynamic configuration must be used, then during the configuration of this sentence it is not known who participates in the meetings. So, this needs to be done during runtime.

1. As a first step, we will add the configuration to the *sentences.yml* file:

```yaml
- sentence: '$SPEAKER_NAME talked most about $MOST_IMPORTANT_TOPIC_0.'
  sentence_id: SPEAKER_MOST_IMPORTANT_TOPIC
  importance: 1
  categories:
    - TOPIC
    - USER
  variables:
    - type: SPEAKER_NAME
    - type: MOST_IMPORTANT_TOPIC
  dynamic_config: SPEAKER
```

   The important data that we added is the *sentence*, *variables* and *dynamic_config*. It is important that the variable types are written the same as it was in the sentence itself.

   > *Note: The _0 in the sentence is there because the* `MOST_IMPORTANT_TOPIC` *variable returns more than one value. > In our case only the first one is used.*

2. Once the sentence was added to the configuration, we can add the dynamic config. The implementation of the dynamic configuration has to get all the speaker ids and create for each of them a *SentenceConfig* with a unique id. The speaker id is then assigned into the parameters dict of this *SpeakerConfig*.

```python
class SpeakerConfig(DynamicConfig):
    @staticmethod
    def get(transcript: Transcript) -> list[SentenceConfig]:
        configs: list[SentenceConfig] = []
        for speaker_info in transcript.speaker_info:
            speaker_id = speaker_info.get('id')
            config = SentenceConfig(config_id=f"speaker_{speaker_id}")
            config.parameters = {
                "speaker_id": speaker_id
            }
            configs.append(config)
        return configs
```

   Now, if the application starts it crash immediately. Because the dynamic configuration with the *SPEAKER* name is not mapped to the *SpeakerConfig* class. This must be done in *models/definitions/dynamic_config_types.py* as follows:

```python
class DynamicConfigType(enum.Enum):
 SPEAKER = SpeakerConfig
```

3. As the last step, the variables must be implemented. In some cases, already existing variables can be used. But for the sake of this example, new ones are implemented. The *SPEAKER_NAME* variable should simply get from the transcript the *speaker_id* which was added to the parameters dict.

```
class SpeakerNameVariable(Variable):
@staticmethod
    def generate(transcript: Transcript, parameters, count):
        speaker_id = parameters.get("speaker_id")
        speaker = next(filter(lambda x: x.get("id") == speaker_id,
        transcript.speaker_info), {})
        return speaker.get("name", f"Speaker {speaker_id}"), 1
```

The *MOST_IMPORTANT_TOPICS* variable should invoke the Transcript Analyser statistics API. Because we set the *speaker_id* parameter, the transcript and the *speaker_id* are passed to the API. As a return value, we receive the statistics, including the top three topics of specified user. Now we extract the top topic from said return value. This variable can extract more than one topic, but we never specified the count variable, which means it is by default set to one. In the importance method, since the parameters are set, the importance is calculated by the amount of time the speaker talked in total.

```
class MostImportantTopicVariable(Variable):
    @staticmethod
    def importance(value, confidence, parameters, transcript: Transcript):
        if parameters:
            statistics =
            transcript_analyser_api_service.statistic_api(transcript)
            speaker_times = highlight_api_util.get_speaker_times(statistics)
            return speaker_times.get(str(parameters['speaker_id'])) /
            sum(speaker_times.values())
        return confidence

    @staticmethod
    def generate(transcript: Transcript, parameters, count):
        if parameters:
            statistics =
            transcript_analyser_api_service.statistic_api(transcript,
            [parameters['speaker_id']])
        else:
            statistics =
            transcript_analyser_api_service.statistic_api(transcript)
        if statistics is None:
            return None
        topics = highlight_api_util.get_topics(statistics)[0:count]
        if count > 1:
            return list(map(lambda s: ('"' + s + '"', 1), topics))
        return list(map(lambda s: (s, 1), topics))
```

Now, if the application starts it crash immediately. Because the variables are not yet mapped to the corresponding types.
This must be done in *models/definitions/variable_types.py* as follows:

```
class VariableType(enum.Enum):
     MOST_IMPORTANT_TOPIC = MostImportantTopicVariable
     SPEAKER_NAME = SpeakerNameVariable
```

7.4 Technical Enhancements

7.4.1 Fix timeout Bug

We are aware of a bug in our implementation. When a Interscriber user wants to generate highlights for a lengthy transcript, the browser may timeout the request to the backend. Since the analysis of the transcript is time-consuming. The frontend therefore interprets the timeout as an invalid request and displays an error to the user.

The generation of highlights will continue in the background, so this is merely a visual inaccuracy. The generated highlights will be displayed the next time the user visits the Highlights Page.

The communication between the frontend and backend should be improved so that this no longer occurs.

A potential solution for the generate API endpoint is to immediately respond after receiving the request and then continue with the generation. And once the generation is done an email will be sent. The frontend could then display a message indicating that the generation will take some time and that an email will be sent once it is complete.

This would prevent the frontend from receiving a timeout and the user from becoming confused. In addition, the top five highlighted sentences could be included in this email, thereby enhancing the utility of this function.

7.4.2 More Variable Configurations

During the implementation of the sentence

```
There was a fight between $SPEAKER_0 and $SPEAKER_1 for $FIGHT_MINUTES
```

it was discovered that there is room for improvement. Because in the case of this sentence the `SPEKAER_NAME` variable should be used twice to display two different speaker names.

The dynamic configuration for this sentence receives two speaker ids, which are added to the parameters dict. But the variable can only consume one with the key `speaker_id`.

Currently, this issue would be resolved with a workaround by implementing a second variable, that does the same as the `SPEKAER_NAME` variable, but uses a different key to load the value from the parameter dict.

A better **solution** to this problem is to improve the variable configuration setup. Meaning, it should be possible to pass additional parameters besides `count` in the `sentences.yml`. In the case of `MEETING_FIGHTS` a parameter that allows to override the default `key` of the parameter could be implemented.

```
- type: SPEAKER_NAME
  key_for_value: speaker_1
```

All dynamic variable configuration

The `TITLE_FROM_CONFIG` variable could be generalized and renamed with `KEY_FROM_CONFIG`. The variable configuration could take `key` as an additional parameter, which is then used to load the specified key from the parameters.

```
- type: KEY_FROM_CONFIG
  key: sentiment
```

The current implementation of this variable looks like this

```
class TitleFromConfigVariable(Variable):
    @staticmethod
    def generate(transcript: Transcript, parameters, count):
        return parameters.get('title'), 1
```

The proposed implementation would not only allow more flexibility with other sentences. It would also improve upon the importance of the variable value. The importance would be calculated in the `dynamic_config` and directly influences the ranking.

```
class KeyFromConfigVariable(Variable):
    @staticmethod
    def generate(transcript: Transcript, parameters, variable_params):
        key = variable_params.get('key')
        return parameters.get(key), parameters.get('importance')
```

8 Feature Evaluation

Throughout the initial phase of the project, we discussed various design concepts with our supervisors, which are also Interscriber stakeholders. This was conducted during a weekly meeting to iteratively adapt and refine our concepts. In addition, they suggested further ideas to improve the highlights feature both visually and functionally.

8.1 First Feedback

After implementing the first version of the feature, it was evaluated by presenting it to selected reviewers. The received verbal feedback and a extensive written feedback regarding UI design modifications were directly converted into tasks and incorporated into the final solution.

The written feedback is addressed in the chapter 5.1.3 and appended in the chapter 15.1.

8.2 User Survey

For the final user evaluation, we created a survey. The results can be found in chapter 15.6 in the Appendix. Participants were granted access to the new highlights feature, allowing them to understand how it functions so that they could complete the survey.

In addition to requesting an evaluation of the feature, we also posed more general inquiries. We questioned as to whether a meeting recap comprised of highlight sentences is useful. In addition, we asked them to assess the relevance of the proposed highlight sentences.

8.2.1 Survey Result

Due to the extensiveness of the survey, we only received few results. Nevertheless, they can be used as indication for further development. The detailed results can be found in the Appendix chapter 15.6.

Usability of Highlight Sentences – Are highlight sentences useful as a meeting summary?

The responses indicate that the majority of users would benefit from meeting summaries in the form of highlight sentences. It verifies that the concept of the Interscriber highlight feature is valid. Nevertheless, most respondents would also find flow-text summaries helpful (Appendix: 15.6.1).

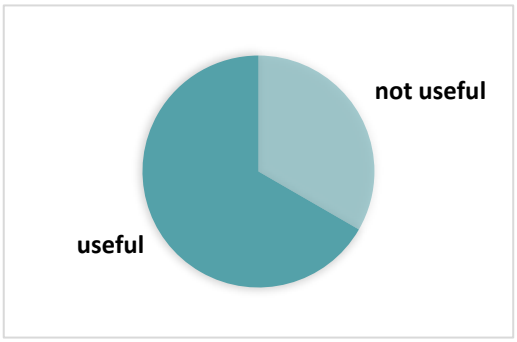


Figure 18: Survey result showing that highlight sentences are useful. (Appendix 15.6.1)

Usability of the Highlights Feature – Is the Highlight feature Easy to Use?

The answers show that the UI was intuitive for most users.

We asked participants to provide their feedback. The main negative feedback was because of an old browser version, which breaks the UI design for the **add sentence** dialog. Even though this was factored into the rather poor result, it should not be further considered, since the operating system they used is over five years old.

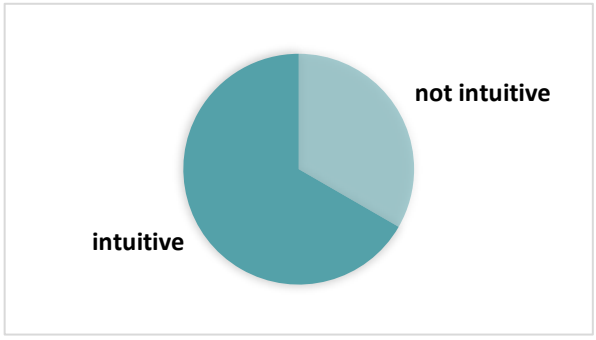


Figure 19:Survey result showing that the highlights feature is intuitively designed. (Appendix 15.6.2)

Importance of the proposed highlight sentences

To refine the ranking of the highlight sentences, the participants were asked to grade the following sentences by their importance.

Each highlight sentence was assigned a value between one and five. According to their assignments, the sentences in the following list are arranged in descending order. The score consists of the average importance derived from the responses.

Score	Sentence Type
4.8	Most important topics were **X**, **Y** and **Z**.
4.8	The main decisions were **A** and **B**.
4.8	Present in the meeting: **Speaker1**, **Speaker2** and **Speaker3**
4.7	The topics on the agenda were **A**, **B** and **C**.
4.7	**Speaker1** will do **ActionX** by **15.06.22**.
4.6	The deadline for **ActionX** is set to **15.06.22**.
4.6	**EventX** will take place on the **15.06.22**.
3.8	**X** was the topic where most participants contributed something.
3.4	**Speaker2** spoke critical of **TopicY**.
3.1	The sentiment of the meeting was **very positive**.
3.1	The sentiment of the meeting was **very negative**.
2.9	**Speaker2** was very dominant, speaking more than **60%** of the time.
2.8	The sentiment of the meeting was **fairly negative**.
2.8	**Speaker1** talked most about **TopicX**.
2.7	**Speaker2** spoke the most, **Speaker3** spoke the least.
2.6	The sentiment of the meeting was **fairly positive**.
2.3	There was a fight between **Speaker1** and **Speaker2** for **3 Minutes**.

Table 8: Survey result showing the importance of the sentence types according to the respondents. The score consists of the average importance derived from the responses. (Appendix 15.6.4)

With this ranking, the importance calculation within Interscriber can be refined.

9 Discussion

The new highlight function enables Interscriber users to generate highlight sentences for their transcribed meeting. The user receives a pre-selected list of highlight sentences, but may modify this collection by selecting from a variety of sentences organized by category and importance. The selection can then be emailed to interested parties. The highlight sentences include information about the speakers, topics, and sentiments. In addition, sentences about action items, decisions, events, and fights are mocked to demonstrate the potential of our templating engine, if corresponding NLP services are added in the future. Unfortunately, English is currently the only supported language.

The creation of fully dynamic meeting highlights and summaries using AI solutions is the subject of extensive research. Even though some of them appear promising and will likely replace our method in the future, we believe our method is a good way to create usable meeting highlights for the time being.

Using NLP algorithms and static templates to create meeting summaries is a simple and reliable method. There are a variety of algorithms and AI frameworks that can extract specific meeting characteristics. Our solution enables the aggregation of such data into comprehensible sentences. Additionally, it is simple to add new or exchange existing NLP services.

10 Outlook

Even though our tool is a useful solution for basic highlight generation, much work must be done before it can compete with other comparable solutions.

Throughout all phases of this thesis, the following improvements and features were collected. Due to time constraints, we were only able to implement a subset of our ideas. However, we hope to see some of them added in the future.

10.1 Remove Mocked Sentences

Table 9: Mocked sentences contains a list of sentences with mocked dynamic data. They are mocked since there are no data endpoints capable of analyzing the transcript and extracting the required information.

We added them to test the extensibility of our solution and to demonstrate the potential of our templating engine.

Topic 3 was the topic where most participants contributed something.
Event 1 will take place on the **21.08.2022**.
The deadline for **Action Item 1** is set to **22.08.2022**.
Klaus will **Action Item 1** by **22.08.2022**.
The main decisions were **Decision 1, Decision 2**.
Klaus spoke critical of **Topic 1**.
The topics on the agenda were: **Agenda Topic 1, Agenda Topic 2**.
There was a fight between **Klaus** and **Hans** for **4 minutes**.

Table 9: Mocked sentences

We expect there to be a much larger number of sentences that could be interesting for the Interscriber users and those who were unable to attend a meeting.

10.2 Adding Multi-Language Support

Since Interscriber is a tool that supports multiple languages, it is essential to incorporate this feature for the highlight sentences as well.

Two areas must be covered:

1. The templates must be localizable.
2. These sentences must have translatable variables.

This enables the highlight sentences to be viewed in any supported language.

However, the transcript analyzer is only capable of analyzing English-language transcripts. Adapting the algorithms may be necessary to enable multilingual support there. For some algorithms, it is merely a matter of dynamically loading different models that have been trained for exact language. Some implementations may require a more extensive refactoring.

10.3 Improve "Send"

Currently, the *send* button on the *highlights overview page* launches the default email client and inserts the highlighted sentences into the body of the email. Unfortunately, doing so prevents us from customizing the formatting the look and feel of it.

To incorporate the styling of Interscriber into the email, it would be necessary to send the message directly from the Interscriber backend. This allows for the addition of images and text to promote Interscriber.

However, the user will no longer be able to modify the message. This may be problematic for some users, as they may wish to add their own summary to the email.

10.4 Addition of Presets

In a professional setting, it is not uncommon for meetings to occur weekly or daily. However, in the current version of Interscriber, the same few sentences must be selected each time meeting highlights are generated. And this may become tedious. Particularly when the meeting occurs daily.

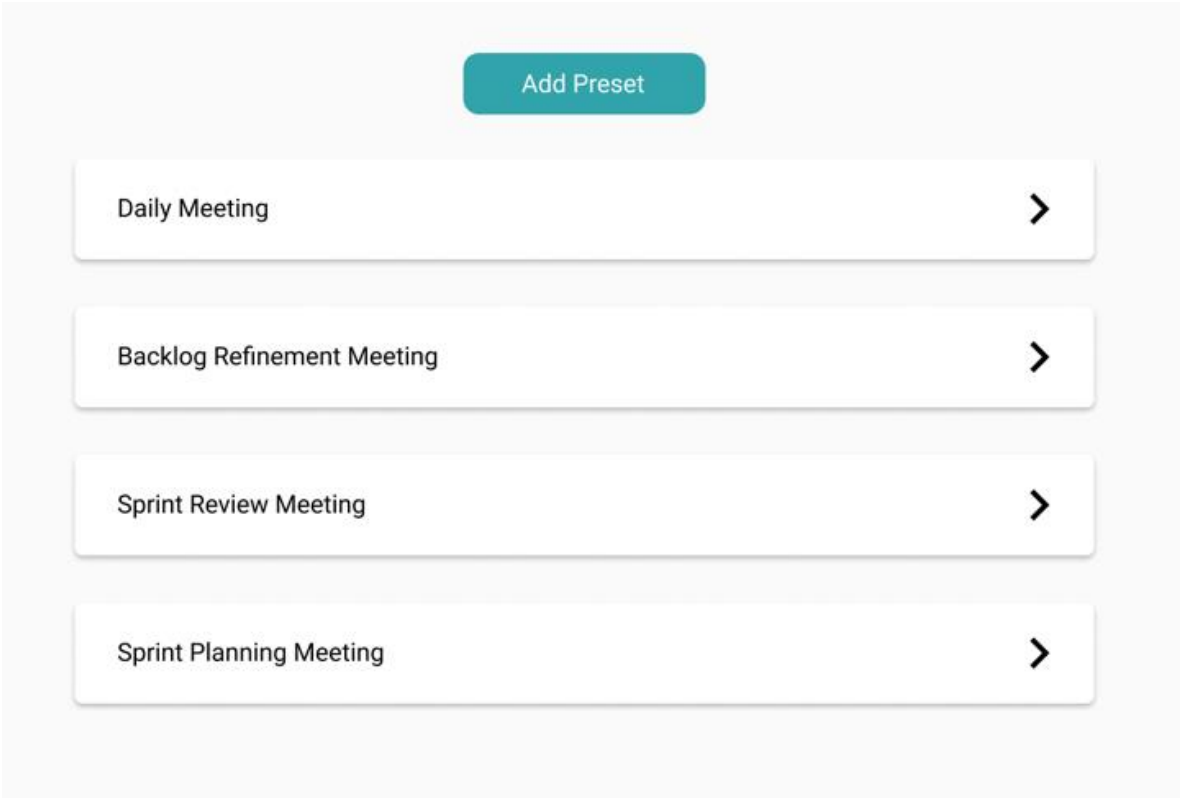


Figure 20: Presets list mockup

A solution to this problem is implementing a feature we call presets. A preset contains user groups, like *Developer* or *Manager*. Each user group includes a list of highlight sentences that can be maintained by the Interscriber user himself, like the current highlight selection procedure. To characterize each user group, keywords can be added. If a highlight sentence contains this or a related keyword, it is ranked higher automatically.

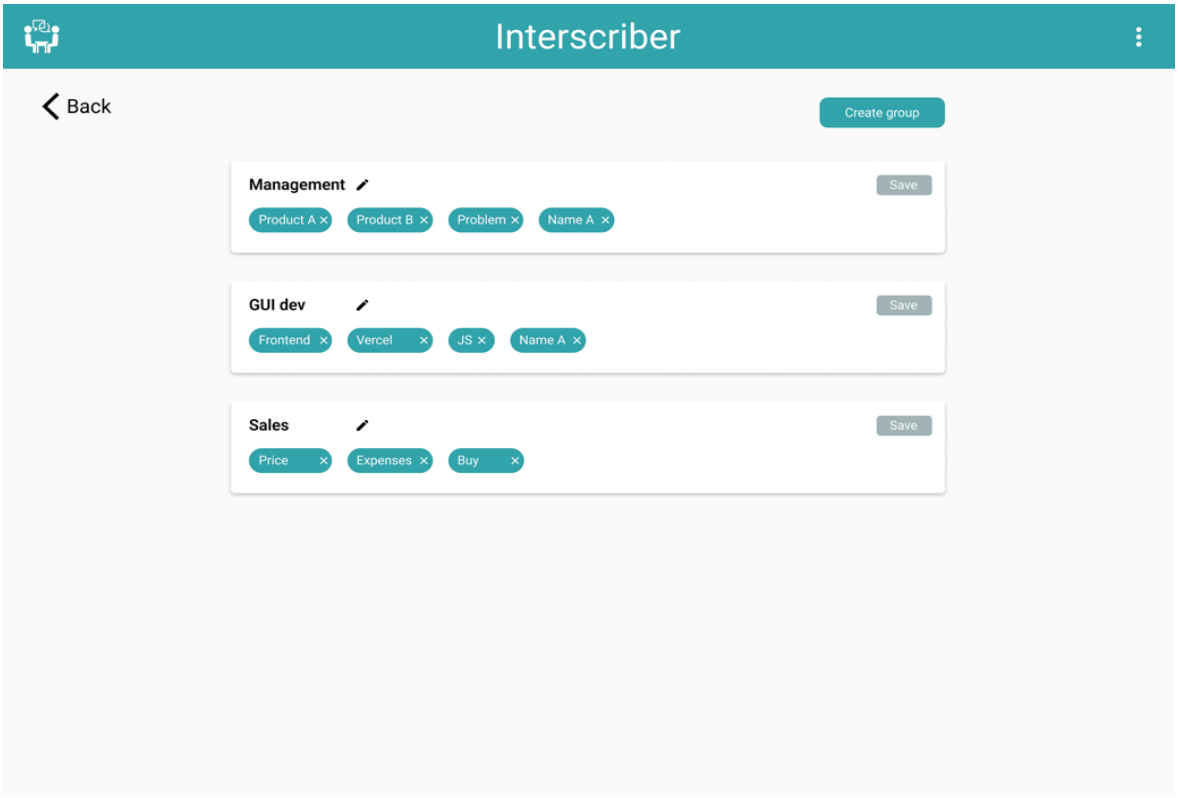


Figure 21: User groups of preset mockup

In addition, we recommend extending the filtering system to the frontend. Currently, sentences can be filtered via backend configuration. This feature should be expanded so that the Interscriber user can specify when a sentence should be displayed and when it should not.

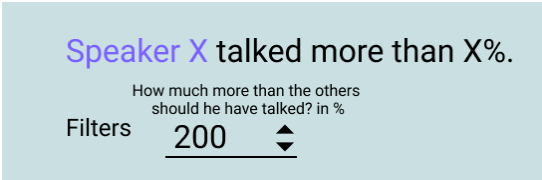


Figure 22: Sentence filtering mockup

> *Example: The sentence "**Speaker X** talked more than **Y%** of the time." should be configurable by the user. A value of 200% indicates that Speaker X must have spoken for more than twice as long as the average speaker. And if the value of a meeting is less than 200%, this sentence is hidden.*

This was not implemented into the current solution because it does not make sense at this time. If the Interscriber user can simply deselect a sentence, they will not configure a filter to hide it.

As a final feature for the presets, we propose allowing the user of Interscriber to manually rank the sentences. They should be able to manually drag and drop the sentence into the order they deem most logical. On top are the most significant sentences, and at the bottom are the least. This will not replace the existing ranking system; rather, it will enhance it. The manual ranking will add a new variable to the equation in order to improve the algorithmic ranking.

10.5 Interscriber User can Manage their own Sentences

Since most sentences are unchangeably written in the backend (modifying a sentence requires a redeployment), a new feature should allow Interscriber users to modify existing sentences or even add new ones. Thus, the configuration in the backend serves as the base implementation upon which users can expand.

The UI should allow the user to select the type of variable to be added to the sentence, add filters or parameters, and then write the template sentence in a text field.

A live preview supplemented by a rudimentary compilation step – essentially analyzing the availability of all variables – is displayed. The variables use mocked data that they provide to demonstrate how a potential sentence might look.

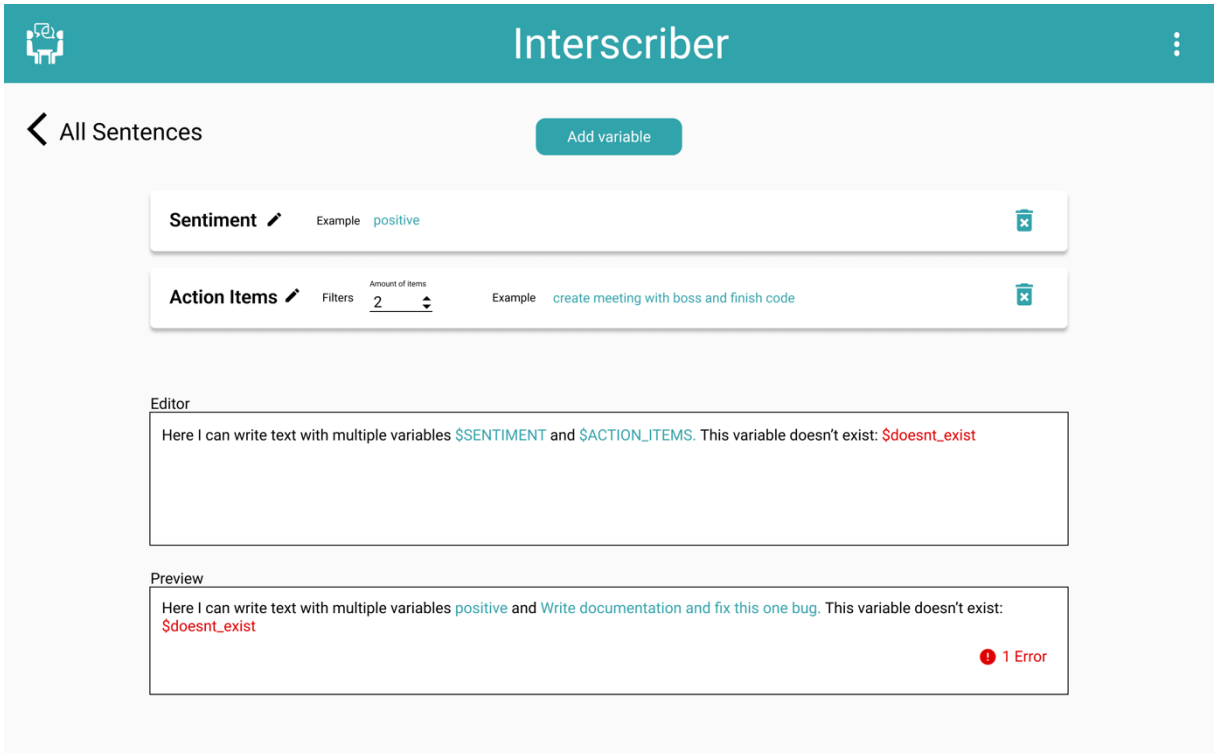


Figure 23: Sentence editor page mockup

10.6 Linking of Highlight to Transcript

In the current implementation, the source of the extracted data is unknown. To improve this untransparent process, each generated sentence should contain a link to the text segment from which it was extracted.

*Example: When clicking on the sentence "There was a fight between **Hans** and **Klaus** which lasted for **3 minutes**." the text part and the audio snippet are displayed to the user.*

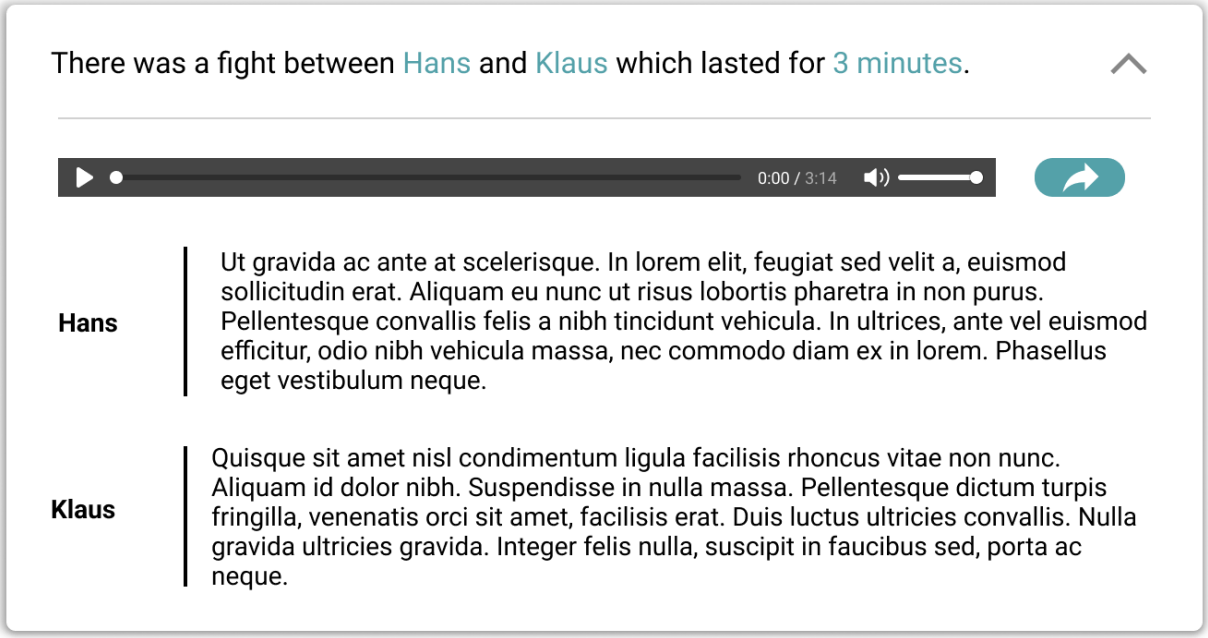


Figure 24: Linking of highlight to transcript mockup

10.7 Utilizing the Dynamic Data to Create a Flow Text

Since the generation of the highlight sentences requires to gather many different types of information, this information could be used to create a summary.

We propose using a data-to-text generation (D2T) system that can produce text in natural language from structured inputs such as JSON.

10.8 Dynamic Categories

Currently, there are only three types of sentence categories. This is limiting and ought to be expanded so that sentences can define dynamic categories. The translation must be provided by the dynamic categories; otherwise, it will not be displayed in the add sentence dialog.

10.9 Rollback of the changes

Currently, all changes made in the add sentence dialog are transmitted to the backend. It would be a useful addition if there was an additional cancel button that resets all changes and retains the previous selection.

This simple change is expected by the user, as it has long been the default.

10.10 Automatic Calibration of the Ranking

In addition to the proposed manual ranking described in chapter 10.4, this feature should automate this procedure.

Each Interscriber user selects the most relevant phrases for each meeting. Based on these statistical data, the importance values of the sentences can be adjusted. There may also be a relationship between the variable value and the deselection/selection of a sentence. Consequently, variable importance can also be adjusted.

This could improve the automatic selection of the top five sentences after the initial highlight generation.

11 References

[1] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871-7880, 2020.

[2] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2018.

[3] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," OpenAI, 2018.

[4] P. Schmid, "bart-large-cnn-samsum," Hugging Face, [Online]. Available: https://huggingface.co/philschmid/bart-large-cnn-samsum. [Accessed 8 June 2022].

[5] A. See, P. J. Liu and C. D. Manning, "Get To The Point: Summarization with Pointer-Generator Networks," *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073-1083, 2017.

[6] B. Gliwa, I. Mochol, M. Biesek and A. Wawer, "{SAMS}um Corpus: A Human-annotated Dialogue Dataset for Abstractive Summarization," *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 70-79, 2019.

[7] OpenAI, "OpenAI," 2022. [Online]. Available: https://openai.com/api/. [Accessed 07 06 2022].

[8] K. Elkins and J. Chun, "Can GPT-3 pass a writer's Turing Test?," *Journal of Cultural Analytics 5 (2)*, p. 17212, 2020.

[9] L. Floridi and M. Chiriatti, "GPT-3: Its Nature, Scope, Limits, and Consequences," *Minds and Machines*, pp. 1572-8641, 2020.

[10] Sembly AI, "SEMBLY AI," [Online]. Available: https://www.sembly.ai/. [Accessed 6 June 2022].

[11] QuillBot, "QuillBot," [Online]. Available: https://quillbot.com/summarize. [Accessed 6 June 2022].

[12] Wordtune, "wordtune," [Online]. Available: https://app.wordtune.com/read. [Accessed 6 June 2022].

[13] Summari, "summari," [Online]. Available: https://www.summari.com/. [Accessed 6 June 2022].

[14] Wikipedia, "Switzerland," 5 June 2022. [Online]. Available: https://en.wikipedia.org/wiki/Switzerland. [Accessed 6 June 2022].

[15] WakeUpNYCTV, “Youtube, Weekly Meeting Example,” 19 November 2013. [Online]. Available: https://www.youtube.com/watch?v=3WrZMzqpFTc. [Accessed 6 June 2022].

[16] P. Aigner, M. Gerber and B. Rohr, *Evaluation of Approaches to Summarize Dialogue Transcripts with Focus on User- friendliness,* 2021.

[17] S. Rose, D. Engel, N. Cramer and W. Cowley, "Automatic Keyword Extraction from Individual Documents," *Text Mining: Applications and Theory*, pp. 1-20, 2010.

[18] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes and A. Jatowt, "YAKE! Keyword extraction from single documents using multiple local features," *Information Sciences,* vol. 509, pp. 257-289, 2020.

[19] M. Grootendorst, A. Matsak, Y. Ogura and V. D. Warmerdam, "MaartenGr/KeyBERT: v0.5.1," 05 May 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6517936. [Accessed 06 06 2022].

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[21] T. Schopf, "KeyphraseVectorizers," 03 06 2022. [Online]. Available: https://github.com/TimSchopf/KeyphraseVectorizers. [Accessed 07 06 2022].

[22] B. &. C. Careers, "Youtube, Mock Consulting Interview," 01 10 2019. [Online]. Available: https://www.youtube.com/watch?v=psuKyBsH1oo. [Accessed 07 06 2022].

[23] "Youtube, Mock Interview with Feedback," 04 03 2015. [Online]. Available: https://www.youtube.com/watch?v=R_dxlajqA4s. [Accessed 07 06 2022].

[24] M. Rehkopf, "User stories with examples and a template," [Online]. Available: https://www.atlassian.com/agile/project-management/user-stories. [Accessed 08 06 2022].

12 Glossary

API (Application Programming Interface) The interface of an application called by other applications or components.

BART Is used to train models for different NLP tasks. 4

GPT A model used for different NLP tasks. GPT-3 is the latest version. 4

Highlight Sentence A sentence generated by the highlights feature using sentence templates.

Highlights A list of highlight sentences representing the summary of a meeting.

Importance The value after which the highlight sentences are ranked. Describes how relevant a sentence is for the user.

Interscriber An application to transcribe meetings. The highlights feature is implemented in Interscriber. 2

JSON A markup language used for sharing information between applications.

KeyBERT An algorithm that extracts keywords. 11

Mockup The design of a user interface.

NLP (Natural Language Processing) The science of letting AI (Artificial Intelligence) understand, interpret, and manipulate natural language.

RAKE An algorithm that extracts keywords. 10

Sentence Config A subcomponent of the sentence type, which describes different manifestations of the sentence.

Sentence Configuration The artifact containing all different sentence types.

Sentence Type The definition of a sentence. One sentence type exists for each template.

Transcript Analyser An application used by the highlights feature to analyze transcripts. 3

UI User Interface

UX User Experience

YAKE! An algorithm that extracts keywords. 11

YAML A markup language used for configurations.

13 List of Figures

14 List of Tables

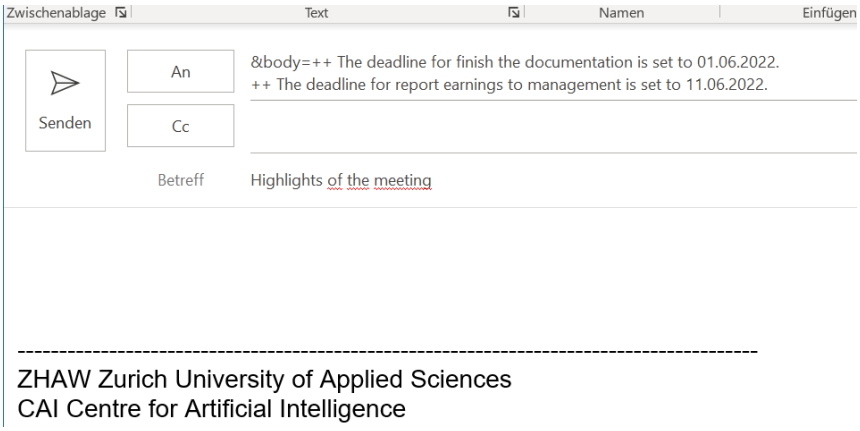
15 Appendix

15.1 User Feedback for the First Iteration of the UI

15.1.1 Mark Cielback Feedback Email

Hallo Zusammen
Ich konnte jetzt euer Tool selber ausprobieren, und es funktioniert! Das ist super, es läuft und ist bis jetzt nicht abgestürzt.
Hier sind ein paar Sachen die mir beim Testen aufgefallen sind:

- Die Schrift ist klein, sogar kleiner als die Texte vom Transkript.
- Wenn in einem Satz Topics stehen, könnte man die in " .." setzen, damit sie besser erkennbar sind
- Es wirkt so als wären in den Topics die Stopwords entfernt. Das ist für die Topic Detection vermutlich sinnvoll, aber zum Lesen eher schwierig. Vielleicht als Feature Request an die API von Zhivar stellen dass man den "most representative text" eines Topics mit Stopwords bekommt.
- Wenn ich auf "send" drücke, werden die Highlights als Empfänger eingefügt:



- Wenn man das erste Mal die Highlights für ein Transkript öffnet, erscheint ein Overlay "New Sentences". Hier ist unklar was das ist. Warum sind die "New"? Ich glaube das ist gedacht falls man später nochmal highlights öffnet und dann neue Sätze im Backend erzeugt wurden - aber beim ersten Mal ist das verwirrend. Vielleicht beim ersten Öffnen einfach nicht anzeigen? Oder sonst einen kleinen Text dazu, was das ist -> und auch dass davon die TOP-5 ausgewählt werden, und man später die anderen Sätze via Edit hinzufügen kann.
  Unabhängig davon: Im Titel vielleicht "Sentences" durch "Highlights" ersetzen. Und um das Fenster zu schliessen muss man auf das Kreuz klicken -> es ist unklar was dann passiert. Vielleicht enen Button "close" einfügen?

- Edit: hier erscheinen zunächst die Sätze die man aktuell hat, und man kann sie nur löschen. Erst wenn man auf "Add" klickt kann man weitere Sätze hinzunehmen.
  Ich fänd's intuitiver wenn man direkt auf den Modus von "Add" kommt, also grad alle Sätzen mit den Checkboxen sieht. Die Liste wo man nur löschen kann braucht es meiner Meinung nach gar nicht (oder hat die noch einen anderen Sinn?).

Falls man das erste Delete-Fenster weglässt, kann man auch oben die Buttons Add und Finish entfernen.

- Add-Fenster: hier fehlt ein Button "ok" mit dem man die Aenderungen abschliesst. Im Moment muss man auf das Kreuz oben links klicken, und es ist unklar ob das heisst dass die Änderungen gespeichert werden oder nicht.
- New Featuer im Add-Fenster: man könnte in den 4 Buttons oben anzeigen wieviele Sätze jeweils ausgewählt wurden, also " All (3 of 23)", "Speaker (1 of 7)" etc. wobei die Zahlen in klein/grau sein könnten (vielleiht in zweiter Zeile)
- Der Reload-Button neben "Edit" sieht im Moment nicht aus wie ein Button (diese sind ja immer weiss auf blauem Hintergrund). Evtl einfach einen echten Button "Reload" machen, und dann den Spinner unabhängig davon?
- " Created: 30.5.2022, " ist zweideutig, vielleicht besser "last update: "

- Den Link zurück "< Transcript" mit blauem Pfeil-Button darstellen
- Ich hab's mal für ein 30min Transkript ausprobiert. Beim erstem Mal kam nach ca. 30sek ein Fehler "Sentences couldn't be generated" (ich nehme an da gibt’s einen Timeout?), beim zweiten Mal hat's geklappt.

Soweit mal das erste Feedback. Ich denke die meisten Sachen sind Kleinigkeiten und optische Improvements, generell funktioniert es sehr gut. Die Liste ist auch nicht so gemeint dass ihr das alles machen sollt – eher als Vorschläge, die ihr auch ggf. als Ideen für nächste Schritte in eure Arbeit aufnehmen könnt.

Kleine Story am Rande: Ich habe zweimal auf ein Topic in einem Satz geklickt, um zu sehen was das genau ist. Das ist vermutlich ein Feature was dann irgendwann kommt, dass man zu einem Topic zB die entsprechenden Textstellen anzeigen kann (aber nicht mehr in dieser Arbeit 😜).

Viele Grüsse
Mark

ZHAW Zurich University of Applied Sciences
CAI Centre for Artificial Intelligence

Prof. Dr. Mark Cieliebak

15.2 Use Cases

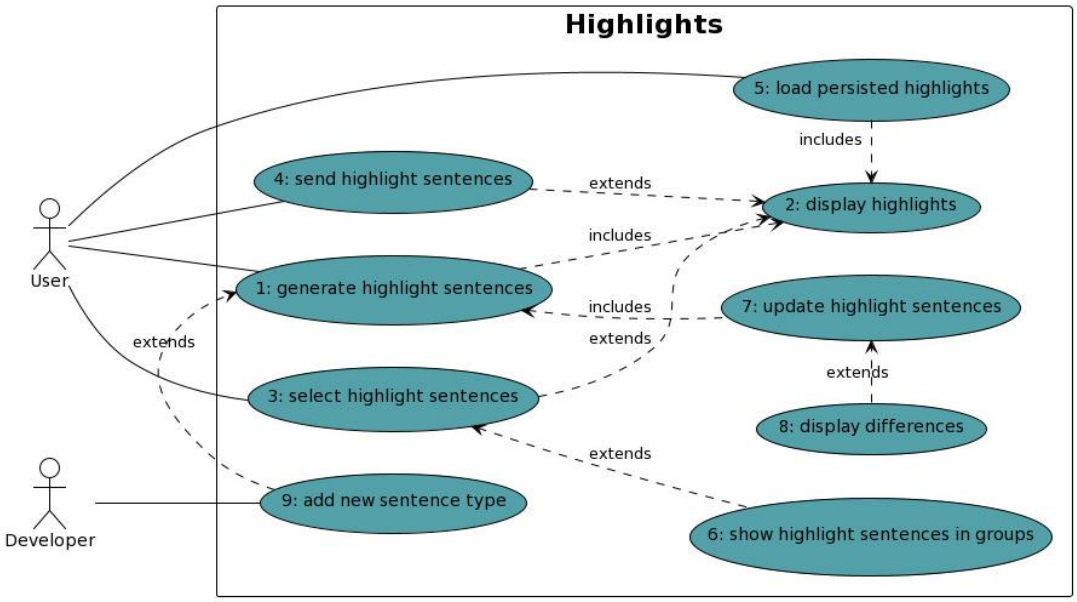


Figure 25: Highlights Use Case Diagram

15.2.1 Generate Highlight Sentences

ID:	**1**
Title:	generate highlight sentences
Description:	The user can generate highlight sentences of his transcribed meeting by clicking on the highlights button.
Primary Actor:	Interscriber User
Preconditions:	The user has transcribed his meeting. To improve the results, they also refined the transcription and added information about the attendees. There are no highlights saved.
Postconditions:	The user is shown an ordered set of the most important highlight sentences, that describe the meeting.
Main Success Scenario:	1. The user clicks on the highlights button. 2. The system loads defined template sentences and sends requests to an nlp-api to get needed information for those sentences respective to the transcript. 3. The nlp-api responds with the correct information. 4. The system fills the information into the template sentences which are then saved and ordered by importance. 5. The system shows the most important sentences to the user.
Extensions:	3a. The nlp-api sends an error or does not respond in time 3b. The system ignores the related sentences and generates the highlights with the other sentences. 3c. The system shows a warning to the user, that not all sentences could be generated.
Priority:	1

Table 10: Generate Highlight Sentences (Use Case 1)

15.2.2 Display Highlights

ID	**2**
Title	Display highlights
Description	Display highlight sentences on the highlights page
Primary Actor	Interscriber User
Preconditions	The Meeting was transcribed and the highlights were generated or loaded.
Postconditions	The user gets an ordered set of the most important highlight sentences, that describe the meeting.
Success Scenario	The system shows the most important sentences to the user.
Extensions	-
Priority	1

Table 11: Display Highlights (Use Case 2)

15.2.3 Select Highlight Sentences

ID:	**3**
Title:	select highlight sentences
Description:	The user can select his own choice of the most important highlight sentences.
Primary Actor:	Interscriber User
Preconditions:	The user has generated the highlight sentences for a meeting.
Postconditions:	The user sees an overview of his selected highlight sentences.
Main Success Scenario:	1. The user clicks on the Edit button. 2. The system shows all the highlight sentences ordered by importance. 3. The user selects highlight sentences and confirms his selection. 4. The system shows an overview of the selected sentences.
Extensions:	-
Priority:	1

Table 12: Select Highlight Sentences (Use Case 3)

15.2.4 Send Highlight Sentences

ID:	**4**
Title:	send highlight sentences
Description:	The user can send his highlight sentences by email.
Primary Actor:	Interscriber User
Preconditions:	The user has generated the highlight sentences for a meeting and selected the highlights they want to send by email.
Postconditions:	The user's primary email program has opened a new email with the formatted highlight sentences as content.
Main Success Scenario:	1. The user clicks on the send email button 2. The system sends a command to open a new email with the formatted highlight sentences as content. 3. The email program has opened a new email with the correct content
Extensions:	3a. The email program responds with an error. 3b. The system forwards the error to the user. 3c. The user sees the error, can act on it and try again with step 1.
Priority:	2

Table 13: Send Highlight Sentences (Use Case 4)

15.2.5 Load Persisted Highlights

ID	**5**
Title	Load persisted highlights
Description	Load and display previously created and persisted highlights
Primary Actor	Interscriber User
Preconditions	The Meeting was transcribed and the highlights were generated or loaded.
Postconditions	The user gets an ordered set of the most important highlight sentences, that describe the meeting.
Success Scenario	1. The user clicks on the highlight button. 2. The system loads the persisted highlights sentences. 3. The system shows the most important sentences to the user.
Extensions	2a. There are no persisted highlights sentences. 2b. The system generates the highlights instead.
Priority	2

Table 14: Load Persisted Highlights (Use Case 5)

15.2.6 Show Highlight Sentences in Groups

ID:	**6**
Title:	show highlight sentences in groups
Description:	The user can select the highlight sentences from groups formed by categories like“Speaker”, “Team” or “Topic”. .
Primary Actor:	Interscriber User
Preconditions:	The user has generated the highlight sentences for a meeting.
Postconditions:	The user sees the generated highlight sentences in groups when editing
Main Success Scenario:	1. The user clicks on the Edit button. 2. The system shows the highlight sentences in groups defined in the template. 3. The user sees the highlight sentences in groups
Extensions:	-
Priority:	2

Table 15: Show Highlight Sentences in Groups (Use Case 6)

15.2.7 Update Highlight Sentences

ID:	**7**
Title:	update highlight sentences
Description:	The user sees updated highlight sentences after changing the transcript.
Primary Actor:	Interscriber User (Person that wants to send Highlights to the persons that did not attend the meeting)
Preconditions:	The user has already generated highlight sentences. The user afterward made changes to the transcript.
Postconditions:	The user sees updated highlight sentences based on the changed transcript.
Main Success Scenario:	1. The user clicks on the highlights button. 2. The system checks if the transcript has changed 3. The transcript has changed. 4. The system generates new highlight sentences. 5. The user sees the new highlight sentences. The selection of sentences is the same as in the previous version.
Extensions:	3a. The transcript has not changed. 3b. The system shows the same highlight sentences that have already been generated. 3c. The user sees the same highlight sentences.
Priority:	2

Table 16: Update Highlight Sentences (Use Case 7)

15.2.8 Display Differences

ID	**8**
Title	Display changes between the last set of highlight sentences and the new one
Description	The sentences are updated regularly. It is important to show the user what changed. Like this, they can see in one step if they like the new changes and does not need to look for them individually.
Primary Actor	Interscriber User
Preconditions	The user has already generated highlight sentences. The user afterward made changes to the transcript. The user is located on the meetings highlight page and the old data is displayed.
Postconditions	The user sees differences between the old and new highlight sentences.
Success Scenario	1. The user clicks on the Update button 2. The system generates the new highlights and compares them to the previous transcript. 3. The system shows all differences between the old and the new highlights in a dialog. 4. The user sees the differences and closes the Difference dialog. 5. The system shows the newly generated highlight sentences with the same sentences selected as in the previous version.
Extensions	2a. The system detects no changes 2b. The system shows the newly generated highlight sentences. The Difference dialog is not shown.
Priority	3

Table 17: Display Differences (Use Case 8)

15.2.9 Add new Sentence Types

ID	**9**
Title	add new sentence types
Description	New sentence types can be added by changing a configuration file. The APIs are well detached and only the most necessary code must be added if a new API or new variables are needed.
Primary Actor	Interscriber Developer
Preconditions	
Postconditions	
Success Scenario	1. The Developer adds or changes a sentence configuration in the configuration file. They then deploy a new version of Interscriber with the new configuration file included. 2. When called, the system generates the new version of sentences.
Extensions	1a. There are changes needed that concerning variables or APIs 1b. The Developer makes the changes in the dedicated areas inside the highlights generation. 1c. He then deploy a new version of Interscriber with the new configuration file included.
Priority	3

Table 18: Cache for APIs (Use Case 9)

15.3 Summarizing Tools Comparison

15.3.1 Switzerland Wikipedia Results

To compare different summarization tools, each tool was given the same text to summarize. The text originates from the Wikipedia article on Switzerland. [14]

Sembly AI needs an acoustic version for being able to create its summary. This is not available in audio; hence it was skipped.

Switzerland Wikipedia
Source: [14]

Switzerland, officially the Swiss Confederation, is a landlocked country at the confluence of Western, Central and Southern Europe. The country is a federal republic composed of 26 cantons, with federal authorities based in Bern. Switzerland is bordered by Italy to the south, France to the west, Germany to the north and Austria and Liechtenstein to the east. It is geographically divided among the Swiss Plateau, the Alps and the Jura, spanning a total area of 41,285 km2 (15,940 sq mi) and land area of 39,997 km2 (15,443 sq mi). Although the Alps occupy the greater part of the territory, the Swiss population of approximately 8.5 million is concentrated mostly on the plateau, where the largest cities and economic centres are, among them Zürich, Geneva and Basel. These three cities are home to several offices of international organisations such as the WTO, the WHO, the ILO, the headquarters of FIFA, the UN's second-largest office, as well as the main office of the Bank for International Settlements. The main international airports of Switzerland are also located in these cities.

The establishment of the Old Swiss Confederacy in the Late Middle Ages resulted from a series of military successes against Austria and Burgundy. Swiss independence from the Holy Roman Empire was formally recognised in the Peace of Westphalia in 1648. The Federal Charter of 1291 is considered the founding document of Switzerland, which is celebrated on Swiss National Day. Since the Reformation of the 16th century, Switzerland has maintained a firm policy of armed neutrality; it has not fought an international war since 1815 and did not join the United Nations until 2002. Nevertheless, it pursues an active foreign policy. It is frequently involved in peace-building processes worldwide. Switzerland is the birthplace of the Red Cross, one of the world's oldest and best known humanitarian organisations. It is a founding member of the European Free Trade Association, but notably not part of the European Union, the European Economic Area or the Eurozone. However, it participates in the Schengen Area and the European Single Market through bilateral treaties.

Switzerland occupies the crossroads of Germanic and Romance Europe, as reflected in its four main linguistic and cultural regions: German, French, Italian and Romansh. Although the majority of the population are German-speaking, Swiss national identity is rooted in a common historical background, shared values such as federalism and direct democracy, as well as Alpine symbolism. This identity stretching across languages, ethnic groups, and religions has led many to consider Switzerland a Willensnation ("nation of volition"), as opposed to a nation-state.

Due to its linguistic diversity, Switzerland is known by a variety of native names: Schweiz [ˈʃvaɪts] (German); Suisse [sɥis(ə)] (French); Svizzera [ˈzvittsera] (Italian); and Svizra [ˈʒviːtsrɐ, ˈʒviːtsʁɐ] (Romansh). On coins and stamps, the Latin name, Confoederatio Helvetica

– frequently shortened to "Helvetia" – is used instead of the four national languages. A developed country, it has the highest nominal wealth per adult and the eighth-highest per capita gross domestic product. It ranks highly on some international metrics, including economic competitiveness and human development. Its cities such as Zürich, Geneva and Basel rank among the highest in the world in terms of quality of life, albeit with some of the highest costs of living in the world. In 2020, IMD placed Switzerland first in attracting skilled workers. The WEF ranks it the fifth most competitive country globally.

Quillbot

Switzerland is a landlocked country at the confluence of Western, Central and Southern Europe. It is bordered by Italy to the south, France to the west, Germany to the north and Austria and Liechtenstein to the east. The largest cities are Zürich, Geneva and Basel. Switzerland occupies the crossroads of Germanic and Romance Europe. It has four main linguistic and cultural regions: German, French, Italian and Romansh. A developed country, it has the highest nominal wealth per adult and eighth-highest per capita gross domestic product. [11]

Wordtune

Switzerland is a landlocked country at the confluence of Western, Central and Southern Europe, composed of 26 cantons, with federal authorities based in Bern. It has a total area of 41,285 km2 (15,940 sq mi) and a land area of 39,997 km2 (15,443 sq mi).

Switzerland was founded in the Late Middle Ages, and became independent from the Holy Roman Empire in 1648. It is a founding member of the European Free Trade Association and participates in the Schengen Area and the European Single Market.

Switzerland is a crossroads of Germanic and Romance Europe, with German as its main language. Its national identity is rooted in shared values and Alpine symbolism.

Switzerland is a developed country with the highest nominal wealth per adult, the eighth-highest per capita gross domestic product, and some of the highest costs of living in the world. [12]

Summari

Switzerland is a landlocked country bordered by Italy to the south, France, and Germany to the west, and Austria and Liechtenstein to the east.

- *It is situated at the confluence of Western, Central, and Southern Europe, and is a federal republic composed of 26 cantons, with federal authorities based in Bern.*
- *Since the Reformation of the 16th century, Switzerland has maintained a strong policy of armed neutrality; it has not fought an international war since 1815 and did not join the United Nations until 2002.*
- *It pursues an active foreign policy and is frequently involved in peace-building processes around the world.* [13]

GPT-3 (Tl;dr Summary)

Version 1:

Switzerland is a federal republic located in Central Europe, bordered by Italy, France, Germany, Austria and Liechtenstein. Its capital is Bern and its largest city is Zurich.

Version 2 (High Temperature):

The country is a federal republic permeated by several cantons, with federal authorities based in Bern. Switzerland is bordered by Italy to the south, France to the west, Germany to the north and Austria and Liechtenstein to the east. [7]

15.3.2 Slogan Meeting Results

This test is based on a meeting protocol handwritten by us.

Slogan Meeting

Tim: Hi Silvy, how are you?

Silvy: Thanks, how was your weekend?

Tim: Ou, very good, we, Holly and I, went hiking and it was simply marvelous.

Greg: Hi

Silvy: Hi Greg

Tim: Hi Greg, so we can start. Am, I'm sharing my screen, do you see it?

Greg: Yes

Tim: So. Last time we defined a new slogan but the management wasn't happy about it. They think it's too provocative and I think it's best we find a way to soften it a bit.

Greg: Really? But it was thought to be a bit provocative, wasn't it?

Tim: Yes, yes. But I see their point, when they say it could be seen as offending. We just need to soften it a little bit. It should still bring people to think, right?

Greg: Right, so whats the plan?

Tim: Yes, you and Silvy, please prepare a new version of a slogan for our meeting at Thursday, ok?

Silvy: Ah, I'm not working tomorrow. Could we not do it for the the weekly at friday?

Greg: Yes, friday sounds good.

Tim: Ok, friday it is. Thanks for your effort, hear you then. Bye.

Silvy: Thanks Bye

Greg: Bye.

Quillbot

Tim and Greg are working on a new slogan for their company, but the management think it's too provocative. They need to soften it a bit, how do you soften a provocative slogan? [11]

Wordtune

Tim and Silvy prepared a new slogan for the weekly meeting, but the management wasn't happy with it. Tim thought it should be softer, so they prepared a new slogan for the meeting on friday. [12]

Summari

Tim: So. Last time we defined a new slogan but the management wasn't happy about it.

- *So, please prepare a new version of a slogan for our meeting at Thursday, ok?*
- *Silvy, I'm not working tomorrow. Could we not do it for the weekly at friday?* [13]

BART (bart-large-cnn-samsum)
Tim, Holly and Tim went hiking with Silvy and Greg at the weekend. Tim, Greg and Silvy need to come up with a new slogan for the weekly meeting on Friday as the previous one was considered too provocative and they need to soften it a bit. [4]

GPT-3 (Tl;dr Summary)
Tim asks Silvy and Greg to come up with a new, softer version of their current slogan for the company they work for. They agree to do it and set a deadline for Friday. [7]

15.3.3 Weekly Meeting Comparison Results

As a sample meeting we used a mock meeting from Youtube [15]. The following transcription was created and edited with the help of Interscriber. The speaker names are fictional.

Weekly Meeting
Source: [15]

Anna: Hello everyone thank you guys for coming to our. Weekly Student Success meeting and let's just get started. So, I have our list of chronically absent students here, and I've been noticing a troubling Trend. A lot of students are skipping on Fridays. Does anyone have any idea? What's going on?

Julie: I've heard some of my mentees talking about how it's really hard to get out of bed on Fridays it might be good if we did something like a pancake breakfast to encourage them to come.

Anna: I think that's a great idea. Let's try that next week.

Nancy: It might also be because a lot of students have been getting sick now that it's getting colder outside i've had a number of students, come by my office with symptoms like sniffling and coughing we should put up posters with tips for not getting sick since it's almost flu season like, you know, wash your hands after the bathroom stuff like that.

Anna: I think that's a good idea and it'll. Be a good reminder for the teachers as well so one of the things I wanted to talk about, there's a student i've noticed here, John. Smith. He's missed seven days already and it's only November. Does anyone have an idea what's going on with him?

Nancy: I might be able to fill in the gaps there i talked to John today and he's really stressed out he's been dealing with helping his parents take care of his younger siblings during the day it might actually be a good idea if he spoke to the guidance counselor a little bit.

Mike: I could talk to John today if you want to send them to my office, after you meet with him it's a lot to deal with, for middle Schooler.

Andy: Great thanks. And, and I can help out with the family Child care needs i'll look for some free or low-cost resources in the community to share with John and he can share them his family.

Anna: Great will some really good ideas here today. Thanks for coming. And if no one has anything else, I think we can wrap up.

Sembly AI
Source: [10]

Tab Glance:

Attendees discussed absent students

So I have our list of chronically absent students here, and I've been noticing a troubling trend. A lot of students are skipping on Friday.

Requirement (0:30)
With symptoms like sniffling and coughing, we should put up posters with expl for not getting sick.

Overview

The meeting was 2 minutes of duration. A total of 1 participant attended this meeting.

Tab Minutes:

*Attendees discussed **absent students***

So I have our list of chronically absent students here, and I've been noticing a troubling trend. A lot of students are skipping on Friday.

Requirement (0:30)
With symptoms like sniffling and coughing, we should put up posters with expl for not getting sick.

Tab Key Items:

Actions

Action (1:25)
I'll look for some free or lowcost resources in the community to share with John, and you can share them with his family.

Requirements

Requirement (0:30)
With symptoms like sniffling and coughing, we should put up posters with expl for not getting sick.

QuillBot
A lot of students are skipping class on Fridays. A number of students have come in with symptoms like sniffling and coughing. It's almost flu season so we should put up posters with tips for not getting sick since it's almost that time of the year. [11]

Wordtune
Anna has noticed that a lot of students are skipping on Fridays. She thinks we should try a pancake breakfast or put up posters with tips for not getting sick since it's almost flu season.

Anna suggested that the teachers remind students to come to school on time, and Nancy suggested that John Smith, who has missed seven days of school, speak to the guidance counselor. [12]

Summari

Weekly Student Success Meeting

- *Anna: Hello everyone, welcome to our weekly meeting*
- *She has a list of chronically absent students*
- *A lot of students are skipping on Fridays, and no one seems to know what is causing this*
- *Maybe a pancake breakfast or something to encourage students to come to school on Fridays*
- *John Smith, a student who has missed 7 days in a row, has been struggling with his parents taking care of his younger siblings* [13]

BART (bart-large-cnn-samsum)

Anna, Julie, Nancy, Mike, Andy and Nancy are discussing the problem of chronically absent students skipping on Fridays. John Smith has missed seven days already and it's only November. Nancy talked to John today. John is stressed out and needs to talk to the guidance counselor. Nancy will send John's missing days to Mike's office. Andy will look for free or low-cost resources for John. [4]

BART (Transcript Analyser)

The implementation in the Transcript Analyser uses the pre-trained model bart-large-cnn-samsum [4].

A lot of students are skipping on Fridays. John Smith has missed seven days already and it's only November. He's stressed out because he's been taking care of his younger siblings during the day. John should talk to the guidance counselor. I will help out with the family Child Care needs.

GPT-3 (Tl;dr Summary)

There is a trend of students skipping school on Fridays, and it is believed to be because they are either tired or sick. A pancake breakfast and flu season posters are being considered as solutions. One student is already being helped by his peers. [7]

15.4 Keyword Extraction Experiment Source

15.4.1 Mary Susan Interview

Mary: Uh huh mary.

Susan: Hi hello, I'm Susan Thompson Resource manager.

Mary: Hi, I'm mary Hanson and I'm applying for one of your kitchen jobs.

Susan: Great.

Mary: Here's a copy of my resume.

Susan: Great, have a seat mary.

Mary: Thank you.

Susan: Mary, do you have any experience working in the kitchen?

Mary: No, but I want to learn, I work hard and I cook a lot at home. I love this.

Susan: Well tell me about yourself.

Mary: Well I love to learn new things i'm very organized and I follow directions. That's why my boss at my last job made me a trainer and the company actually gave me a special certificate for coming to work on time every day for a year and I'm taking an english class to improve my writing skills.

Susan: That's great why did you leave your last job?

Mary: It was graveyard and I need to work days.

Susan: Oh I see well what hours can you work?

Mary: From eight am until five pm.

Susan: Okay well do you have any questions for me mary?

Mary: Yes what kind of training is needed?

Susan: Not a lot most new workers can learn everything the first day do you have any other questions?

Mary: No, I don't think so, but I've heard a lot of good things about your company and I would really like to work here.

Susan: Well I have a few more interviews to do today but I will call you tomorrow if you get the job.

Mary: Okay, you are sure. Nice to meet.

Susan: You Nice meeting you.

Mary: Too thank you so much for your time.

Susan: Yes, good luck.

Mary: Thank you.

15.4.2 Consulting Interview

Source: [22]

Speaker 2: Hi. Hi. Welcome to Maine. Thank you. I'm Emily. Nice to meet you. Nice to meet you too. Okay, follow me? Okay. So I propose that we just jump right into the case and get started. How does that feel? Yeah, let's do it. Okey-dokey. So I'm going to keep my computer screen open and at there's a few pieces of data and some graphs that I I might want to show you but otherwise, I'm ready when you are. Yeah, I'm ready. Great. Our client is next-gen Tech and they've created a wearable device called the Theo and Theo is essentially a little computer on your wrist. It tracks activity. So it tells you your calories, your steps. It also can accept text messages is basically fully connected device. Next gen technology. Wants to go to make 1 billion dollars over the next two years in the u.s., They Brought us in because they have a bit of a problem. Theo has Wi-Fi built in, which is great. But that connectivity is only available currently when it has Wi-Fi. So they want to partner with a cellular carrier, to provide that connectivity, and they've talked to Verizon and AT&T the two biggest players in the US and Verizon and AT&T are excited about the opportunity, but they have said that they will only work with next gen Tech and with Theo, They are the only two providers. So, what next gen tech asked us to come in and help them answer is, should they go with AT&T and Verizon and an exclusive relationship? Or should they pursue a partnership with other.

Speaker 1: Carriers? Great. So, let me summarize the case for you to make sure I got all the details. Our client is next-gen Tech. Our client has developed this wearable device called Theo, which is like a computer on your wrist. It tracks your Fitness calories. You can fax from it. And our goal here is to make the you a 1 billion dollar revenue. Is that correct? Yeah, revenue and in the US only.

Speaker 2: Yes, we're only talking about the u.s. Right now. Okay.

Speaker 1: When complications, that, the device has why? Five-year-old in but our client wants to wants it to work anywhere even when Wi-Fi is not available. So they want to partner with some set of cellular carriers and the options. We have are basically one AT&T and Verizon partnering with AT&T and Verizon, but they demand exclusivity or following order carriers. You got it. Can I ask you one more question? Sure. Are we planning to sell this device? Only through like the carrier's stores.

Speaker 2: Yeah, for now. That's our. Distribution channel is going to be through AT&T and Verizon or other carriers stores.

Speaker 1: Great. Do you mind if I take some some Taiga to organize, my thoughts, go for it? All right. So I'll go with you mate. The old 1 billion dollar business in the u.s. In revenue and the next two years. Yeah, and the options, the question I have to answer is to whether go with AT&T and Verizon and sine X and exclusivity contract or follow other carriers. You got it to assess, thus this problem. I would like to look to Market size of the carriers to the customer base and the retail footprint of the options. Okay, so under Market size. Understand. Since the, the carriers will be the only Logistics, the footprint that destroys will be the only way of selling product. I like to understand the number of customers. We currently have for AT&T Verizon and the other option. What is the price? We were selling to describes, sure. And what is the growth expected between year one and two other customer base? I would like to understand the demographics of the up, the customers, understand their price sensitivity. Sensitivity and

Joe says, the likelihood of the customer base to to purchase our product make sense. And finally, I would like to look at the retail footprint, see what is the number of stores? We currently have for the options. And what is the sales force.

Speaker 2: Available? Okay, I like that. Where do you wanna.

Speaker 1: Start? I'll let you start by the market side. That's fine, that works for me. So first of all, like to understand what is the.

Speaker 2: Price with AT&T and Verizon are Tation is that will sell the Theo for $400. Hmm. And with other carriers, our expectation is that we would sell it for 500.

Speaker 1: Dollars. Is there a reason for that?

Speaker 2: Good question. So part of what makes 18, Verizon AT&T, and Verizon good partners is that they would be willing to throw in a little bit of money to bring down the cost. That customers would have to pay walking out the door with you. So, AT&T and Verizon are essentially helping to subsidize that four hundred dollar cost makes.

Speaker 1: Sense. I would like to know the current number of customers. We have.

Speaker 2: Have okay, over. Why do you understand that?

Speaker 1: So a nice ass, like the total Market? Thank you, Margot, and calculate the penetration rate. So can see how feasible it is to get into one video. Revenue. Got it, each of the.

Speaker 2: Options. Okay, so I have a little bit of data for you then. Let me share that with you.

Speaker 1: Okay, so I have the customer base for AT&T and Verizon and other carriers, which includes Sprint, T-Mobile US Cellular Cricket. It's pretty granular. So this is for the current customer base.

Speaker 2: Right? Right. Now.

Speaker 1: We have the expected customer base next year. Why do you want understand that? Because we want to like our goal is to get to 1 billion Revenue in year. Two. Hmm, and I would like to calculate that the penetration.

Speaker 2: Yeah, next year. So we do have some growth rates. We know the growth rate for both 18, Verizon as well. As for other carriers is about.

Speaker 1: 20%. Give this.

Speaker 2: Back to you kissing you, these numbers.

Speaker 1: So what I would do to calculate the penetration rates is first of all, I would estimate the potential Market by multiplying the number of customers. And in year 2, which means I will calculate the number of customers with plenty percent growth rate. Okay, and divide 1 billion by this number, which is the penetration rate doesn't make.

Speaker 2: Sense you, yeah, I think so. Why don't you just start? Trying them. You can keep going.

Speaker 1: The number of customers for option one, which is AT&T, and Verizon and option two, which is, which are the other characters. So, the growth rate is 20% We get to the number of customers in you too. Okay, which is 24 million for AT&T Verizon and 36 matches at a 20% to each of the.

Speaker 2: Okay. So you didn't twenty times 1.2.? Yeah. Okay.

Speaker 1: Now we have the Price. Per device for each of the scenarios. And here, we have four hundred dollars, fourteen, to Verizon, and $500 for option two. All right, by multiplying these two, I'll get you that potential total Market size. Assuming you have 100% penetration. Okay. So this is the total Mark not nine. Point six billion and 1.18. Okay, option. Since we want 1 billion Revenue in YouTube, I will divide 1 billion by 9.6 and my final. So in this case, I will approximate 10% if that's fine with you and he is 1 billion by 18 Again approximately 5%. If that's fine. That sounds good. So both both penetration rates, seem relatively low. Mmm, of course option to seems easier to achieve. However, since We have more carriers. It's more granular. He's harder to probably standardized, go to market strategy, they have. So I would.

Speaker 2: Do it in a vacuum 5% seems better than ten percent penetration, but because, but because option two is the other carriers. And that's really fragmented. That could be potentially really excitedly. Got it. Okay.

Speaker 1: So we're back to my framework, the next step of like to to take it to look at the customer base. Hmm and understand who our customer. And our customers are. What are they what they value? What about when acquiring a product from from there? They're carriers the way to do that. Be probably to run a survey customer survey.

Speaker 2: Huh? We definitely did do a customer survey and I'm excited to share it with you. So you.

Speaker 1: Can see here that this tells us, what is the most important selection criteria for AT&T Verizon, and the other carriers. So, I'm assuming these numbers here are like the sample size, which means the number of I was interviewed in the survey. You're right. Yep. Hey, 14th. In Verizon. The two, most important selection criterias are reliability of the network and customer service. So I'm assuming AT&T Verizon provide, a better Network and a better customer service. On the other hand. The other carriers are very concerned about price. They're very price-sensitive, which tells me that since our product is like, a new product and we're looking for early adopters earlier. Adopters are probably not very price-sensitive. So, It seems like AT&T Verizon customers have a better fit for our.

Speaker 2: Product. Okay, I thought that makes sense to me. And so do you feel like you have enough.

Speaker 1: Information? Yeah, since the the distribution channels were talking about are the carrots. Stores a light to understand, like take a look at the route, 80 foot print available, sure and see what, what is the number of stores? Each of the options have how good is the Salesforce? And at the app? You have any data on that? I.

Speaker 2: Do, in fact Here we go.

Speaker 1: So he would have the number of stores in the US and outside of the US focusing. First on the u.s., I can see that 18 to 80 and Verizon combine have 4,000 stores and all the

other carriers combined have only 3,000 stores. So we're talking about for option, one AT&T Verizon. We had 33 percent more stores, and outside of the US, that's not true. However, since our goal is in the next two years, at least is to focus, only in the US. I don't think it's a problem right now, seems fair. Yes. So again, 18 Verizon seemed a better fit for, okay.

Speaker 2: Okay, so great. So we have a meeting with the CEO tomorrow and she asked us to come in and let her know what we've learned so far. What our recommendation is based on what we've learned any risks that we perceive and if we have any next steps.

Speaker 1: Hey, do you mind if I take some second thoughts? So, my final recommendation would be to go identity. Verizon. Okay, option one. And the reason for that is that the penetration rates a reasonable for both options. However, does the customer base for AT&T and Verizon customers Simply Better fit to our product because the our customers are the customers are less price, sensitive and are more likely to be early adopters of this new device. Okay, the main solution Channel be the carrier's. AT&T Verizon, they have more stored like 32% more stores. And since this device, I feel like customers will need to like try it on and make sure it fit and look. Well. It's a good. It's a good reason to have a it's good to have more stores. Yeah. Next steps are. First of all, follow up on the risks and since then and finally get it, LOL, Team 2 to write the contract and get the best terms possible for us.

Speaker 2: Great. You're done. Good job. Thank you. Yeah, so next up is on our side is that we will let you know if you make it through to the next round and you should get a call from us. Thanks for being excited about beIN and thanks for coming in and.

Speaker 1: Doing a case with me. Thank you. Thank you for having it's really good to meet.

Speaker 2: You. Danielle did a great job on that case. There were a few things that I really appreciated about what he did first. He listened really intently as I was setting up the case and the complications and the situation, he wrote great notes and then he asked really important clarifying questions at all. So allowed him to set up a framework where he then was able to drive the rest of the case. And so all along the way, he was referring back to his framework. I was able to refer back to it, to keep us on track and stay structured. There often isn't a perfect answer to a case. And what Danielle did a great job of is defending his point of view. He found the data points from throughout the case and brought them into his final recommendation. The last thing I want to mention is that Danielle did a great job of making the interview feel just like a business conversation he had. Ask great questions. He shared of his experience. He allowed me to go back and forth with him and I think we had a little bit of fun. So Bravo Danielle you did a great job.

15.4.3 Interview with Feedback

Source: [23]

Speaker 1: Thank you for coming in today and I'm happy to help you with your preparing for your interview. Why don't we start by having you tell me a little bit about yourself. So I'm a senior Communications major here at Curry.

Speaker 2: Graduating anticipated.

Speaker 1: Graduation is May of.

Speaker 2: 2015. I have very strong points in working with.

Speaker 1: People. I am very involved here at Curry. So I'm a part of.

Speaker 2: The Student Activities office. I'm a student worker. There is an orientation leader as well. I have a background in theater. I've spent almost half of my life doing theater. I have.

Speaker 1: Vocal talents of taking voice lessons for nine years.

Speaker 2: I don't know what else to say. I don't know what else to talk about. I'm from Franklin. Bats.

Speaker 1: Overall, really good. I think you did a really nice job of showing both parts of your background, but also some interest, it really helps to paint a picture of who you are as a person. I think what you need to be careful of. And what is a common mistake is to not just reiterate everything that they're looking at, on your resume. So once you start to get into that, I'm at this type of major and that I did this job, and then the two before that, I did that job. All you're doing is reading your resume to them and they've already read your resume. That's why they brought you in as a candidate. So you kind of think about what your pitch is and what your story is and what is it that I want them to know within the first few minutes of meeting me and that's really hard. This isn't something that's an easy thing to do, but you want to just try to stay away and you actually did. Good job. I would say you weren't. I've had, you know, I well, I'm from Franklin Mass and I went to such and such high school and then I went to Curry College and then I did this job and you didn't do that, which was good. But you want to try to stay away from what I kind of would describe as like the chronology of your background and more as a summary of who you are, as a candidate, what you're looking for where your passion lies as it relates to the.

Speaker 2: Position. What is a weakness that you have our weaknesses when things are going aren't going my way. I tend to, like, stress, myself out to the max. And I just don't know where to go from there. But then when it all comes down to it, I can pick myself back up. And so it's a weakness that builds into a string.

Speaker 1: I have a little bit of a formula that I like students to follow, when it comes to answering this question because this is a tough one. It's a three-part Formula 1 pick something that's honest. And authentic. So, So, you did that. I didn't get the oh, I'm just I work too hard, right? That's not very authentic. So, you did you picked an authentic weakness. Step two is to not pick a weakness that's going to throw yourself under the bus. So you don't want to say, for instance, you're going into PR. I really don't like working with people. Like that's a weakness that's going to be a deal breaker for that. Kind of industry, three, you then take it and you talk about how you mitigate that weakness. So, that's where the step the third step. I'd want you to take. So, I think you hit one and two but three would be to then say. So typically What I do when I feel myself. Getting really overwhelmed, is, I stop. I take a breather or I begin to write lists or I figure out a way to organize my day. So that if you were to ask my my co-workers at Outback, they probably have no idea that I'm like this, but internally, this is something that I go through. And so these are the things that I do to make sure that it doesn't affect my.

Speaker 2: Work.

Speaker 1: Can you give me an example of an accomplishment you had and what that tells us about you.

Speaker 2: So I think one of my greatest accomplishments is the general focus of me going to college. Because initially graduating high school. I didn't, I wasn't planning on going to college. I was a c student. High School wasn't involved in anything. Basically, just went to school and left and didn't really do much and then coming to college. I had that mindset of, I'm just gonna Like, do whatever. And then I got involved on campus. I ended up getting Dean's List, both semesters of my freshman year and then both semester sophomore year. And then it continued up until my senior of made dean's list every almost every semester a couple semesters. I haven't, but close enough. I'm extremely involved on campus. I was an orientation leader. I have been involved in Korea, theater and I'm participating in all of these specific things that go on at Curry, which.

Speaker 1: Initially, I had no idea. No, No idea or.

Speaker 2: Anticipation of me, doing anything that I've done at Curry so far. So, I think that right there is a great aspect of my life that I initially had never expected to happen. Sorry.

Speaker 1: I think again, very good. Very good sort of, common way to answer an interview question. I think taking it up a notch is to complete the question, which I kind of fed you, which was in that. What does that tell us about you as a worker? And that's usually not how that question is going to be asked to say, what's can you? Give me a sense of accomplishment? You've had but what you want in your brain to do is to translate that for them. So your example of making dean's list, so I didn't think I was, you know, going to go to college and then I went to school and I make Dean's List immediately. And what that can tell you about me. Is that once I get myself going, I'm really committed that I've got great time management skills. You know, it took me a little while to really put a commitment to education, but once I did, I was committed and that's where I am now. And you can start to think about what those, what is it that made you get that accomplishment. So what skills? It's because those are transferable skills. Those skills that you might, that helped you to get dean's list or the same type of skills that employer wants that time management, the attention to detail, the getting things done, getting things done on time. Those are the things that help you to get Dean's. Those are also the things that help you be a productive employee. So any time you have an opportunity to talk about an accomplishment you want to tie it immediately, back to what attributes about you helps. You get that accomplishment because that's going to paint a picture for them of. What kind of work are you going to be? You think sets you apart from other candidates?

Speaker 2: This sort of ties until like my strengths. I'm.

Speaker 1: Incredibly well at communicating with people and just being with.

Speaker 2: People in general. I tend to have people gravitate towards me when I, when I go places. I'm respectful.

Speaker 1: I am able to put forth any effort to help.

Speaker 2: Others including myself but others as well, and I'm just very creative as well. And I'm always able to bring like a Creative aspect into a situation. So.

Speaker 1: So again, I would encourage you to think about what sets you apart, what makes you truly different. And oftentimes, look at your combination of experiences because most people won't have the exact same background as you. So if you piece together the different areas that make you different so maybe the combination of event planning plus the theater or

something that is a real interesting cross-section that then brings a certain caliber to the workplace and kind of figure out what that is, and I think the other part of it is Anytime you have an opportunity to talk about a story or an example, that that could have been a good another opportunity to to provide an example to support what you're saying. And when you answer that.

Speaker 2: Question, do you have any questions for us? I don't have any questions. Thank you.

Speaker 1: We always want to have questions prepared prior to coming into an interview. And oftentimes what happens is just through the back and forth of the interview. Those questions get naturally answered. That said, you do want to have some additional questions, or have questions that are most likely not. Going to get answered, unless you ask them. And so the best way to do that is through company research, research, that organize it organization, research, how they've been covered in the news. If you ask a question related to, you know, PR agency or interview with the pr agency and you saw on the news that they just had a new client. Come on. That's a great opportunity to show that you've been really keeping track of them as an organization. So, you know, I noticed that you are now PR Agency for XYZ company key. Tell me a little bit about how that might impact the work of this area that I'm interviewing for. So, that's a way to show that you've I've been engaged and even following them and that you're really interested in them as an organization and it also gives you a chance to ask a really good question.

Speaker 2: But I.

Speaker 1: Would say overall very good. I think you are. You're interviewing level is at above average of a typical college student. I think this feedback is intended to take you to the next level and to be more on the professional seasoned professional level of interviewing. So it's not an oral exam. I think a lot of times students think it's just answering the questions and there's a right way to answer wrong way to answer and it's an ABC. It's not that it's really a back and forth conversation and you should be driving the interview as much as them. So it's your opportunity to learn about the position. The Asian, your anytime you're on an interview, you're trying to figure out whether this organization is a good match for you as much as they're trying to figure out whether you're a good match for.

Speaker 2: Them.

15.5 Keyword Extraction Experiment Result

The following results were generated with Transcript Analyser.

15.5.1 Mary Susan Interview

RAKE

uh huh mary hi hello	Susan thompson resource manager
would really like	Company actually gave
time every day	

YAKE

Susan Thompson Resource	Thompson Resource manager
Susan Thompson	Thompson Resource
resource manager	Susan
Thompson	Resource
huh mary	mary Hanson

Keybert with part-of-speech pattern

kitchen jobs	susan thompson resource manager
writing skills	resume
training	

Keybert with n-grams

experience working kitchen	mary experience working
kitchen want learn	applying kitchen jobs
working kitchen	

15.5.2 Consulting Interview

RAKE

asked really important clarifying questions	four hundred dollar cost makes
old 1 billion dollar business	next gen tech asked us
1 billion dollar revenue	

YAKE

Verizon	customers
customer	Verizon customers
carriers	customer base
Verizon customers Simply	number
Yeah	Great

KeyBERT with pos

cellular carrier	wearable device
cellular carriers	next gen technology
at&t	

Keybert with n-grams

theo wi fi	partner cellular
want partner cellular	provide connectivity
partner cellular carrier	

15.5.3 Interview with Feedback

RAKE

list every almost every semester	part formula 1 pick something
professional seasoned professional level	got great time management skills
questions get naturally answered	

YAKE

Dean List	good
questions	Curry
things	Dean
question	interview
College	weakness

KeyBERT with part-of-speech pattern

interview question	interview
resume	senior communications major
voice lessons	

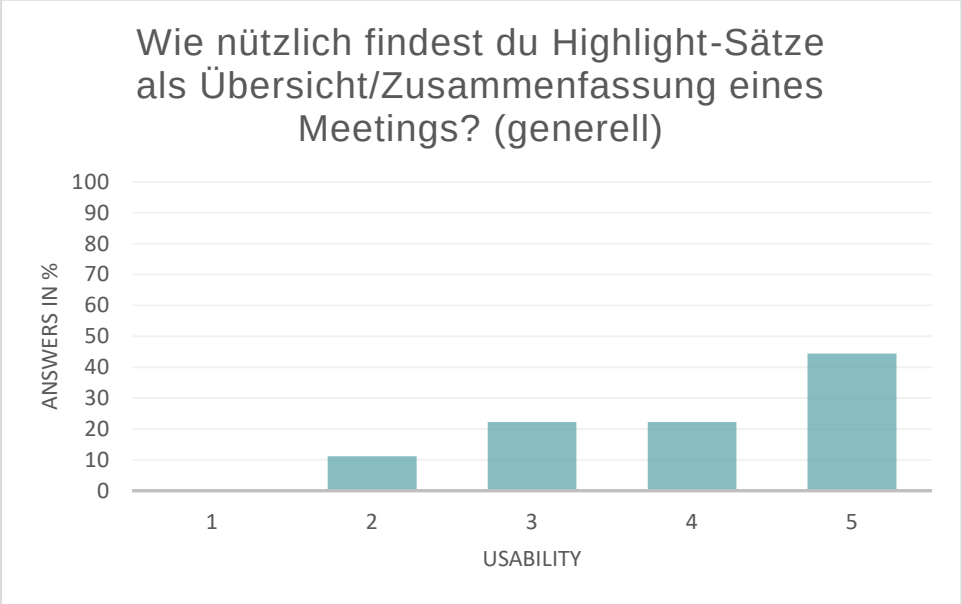
KeyBERT with n-grams

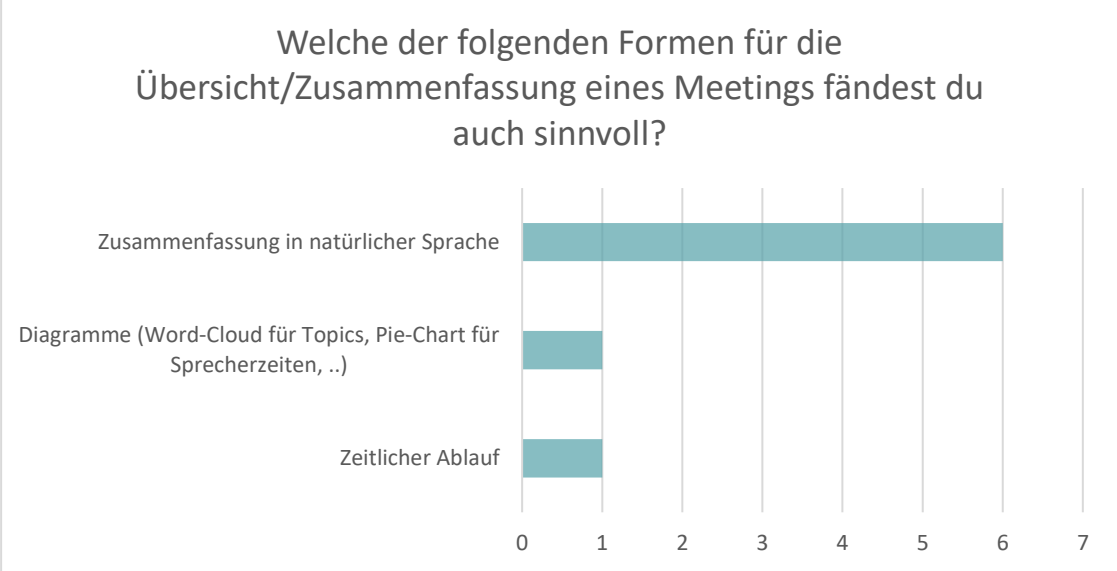
help preparing interview	preparing interview
professional level interviewing	coming interview
interviewing	

15.6 User Survey

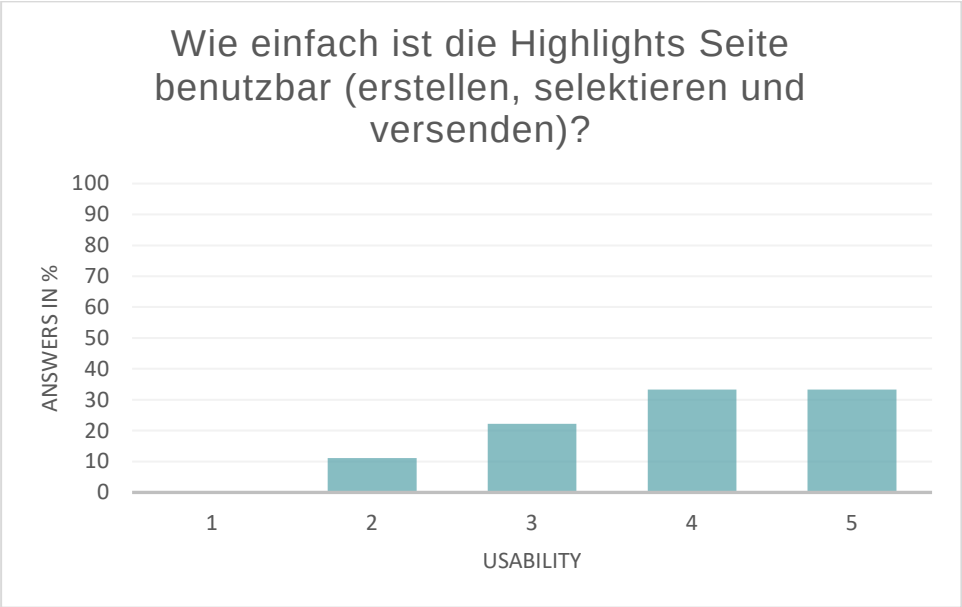
The following charts represent the answers to the user survey we conducted for evaluating the user design.

15.6.1 Usability of Highlight Sentences





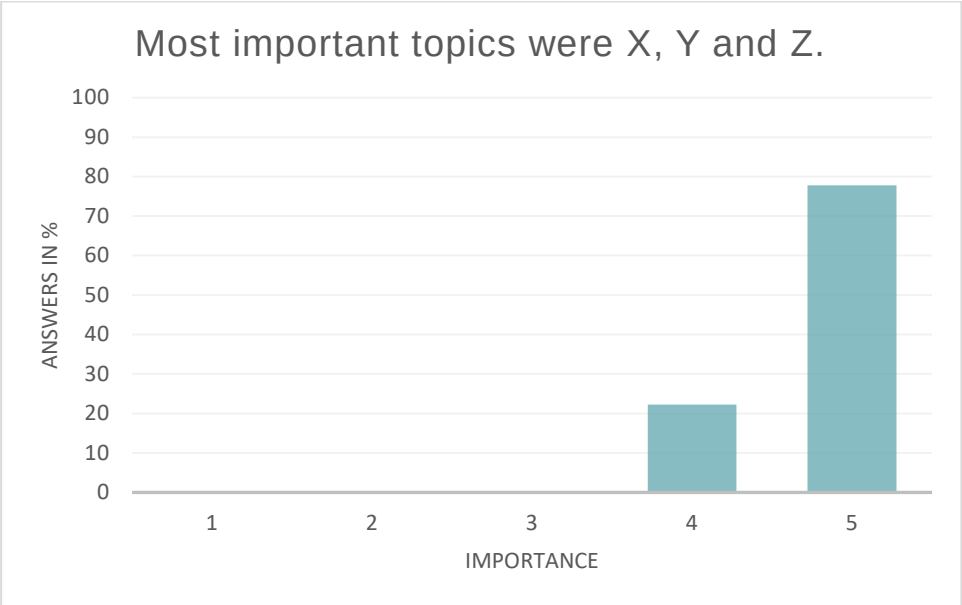
15.6.2 Usability of the Highlight Feature

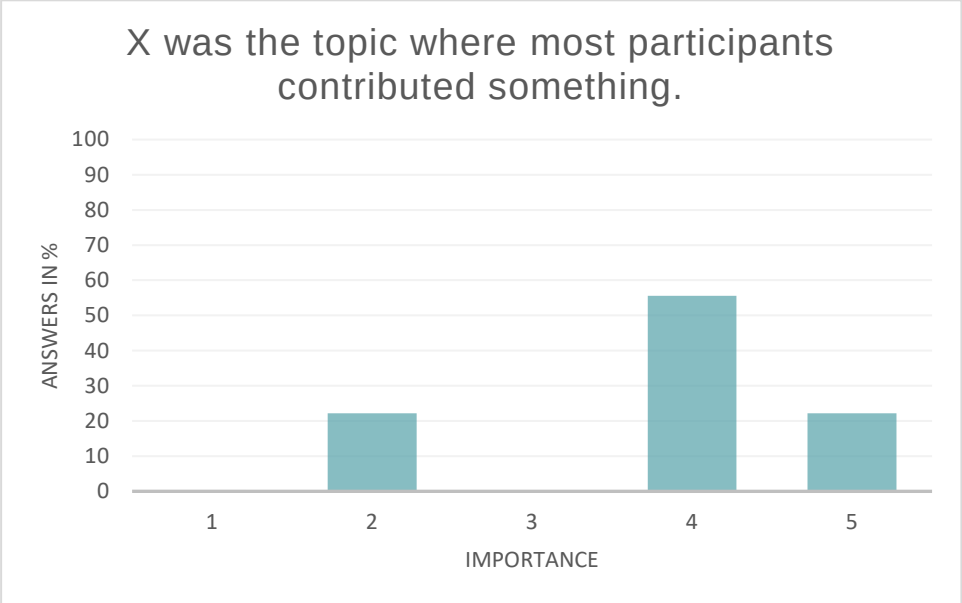


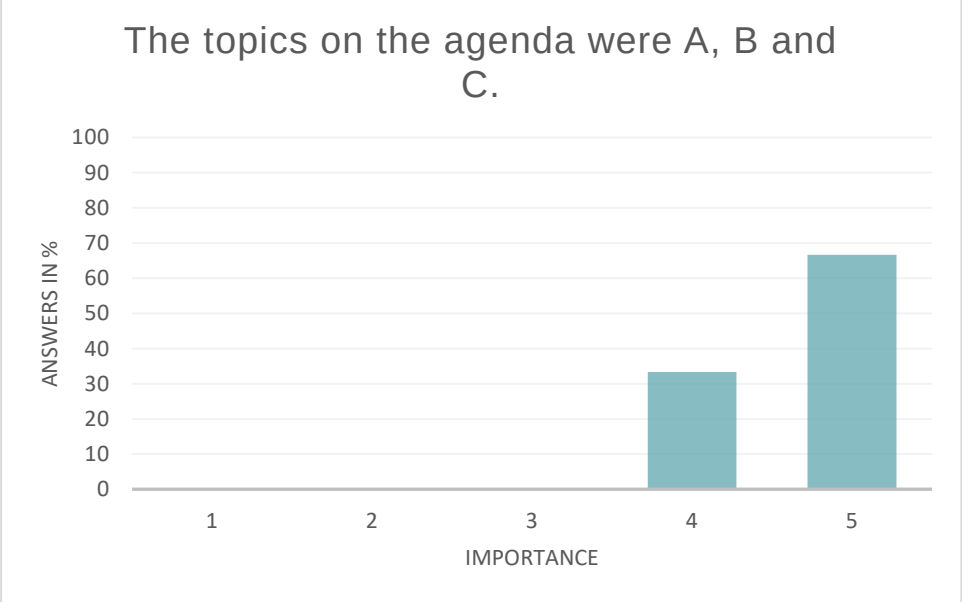
15.6.3 Which parts were hard to use? Do you have any recommendations to the design or user flow?

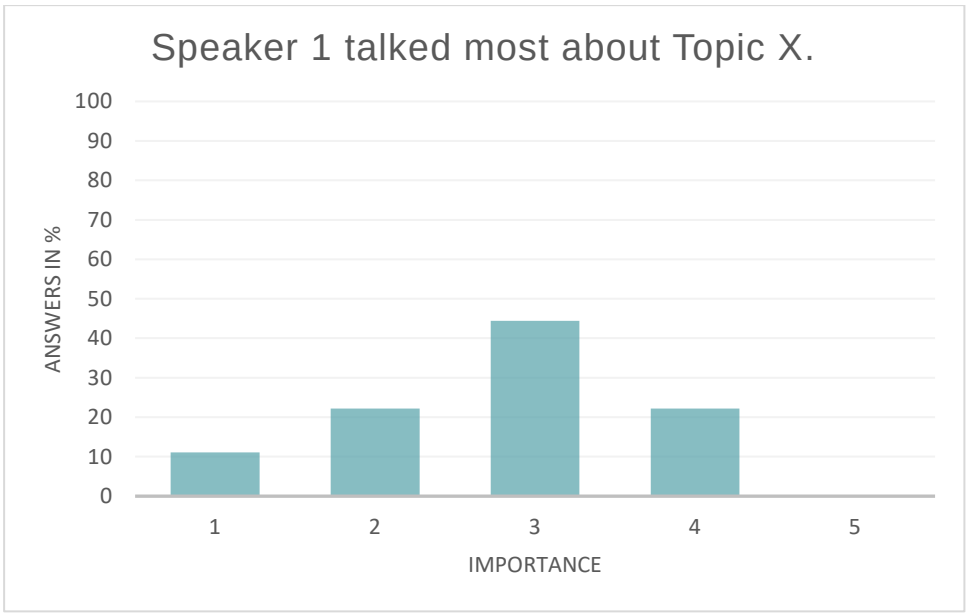
- Text der symbole sollte immer ersichtlich sein
- Auf einem mac (highsierra) Add sentence --> Formatierung nicht korrekt dargestellt z.b. bei längeren Sätzen über 2 Zeilen wird nich alles dargestellt + save-button überdeckt einige "Häkchen-Kästchen" Audio kann nich angehört werden --> Fehler Highlights, die automatisch als top5 angezeigt werden, wirken oft nicht relevant. Es bräuchte zwingend die Abmachungen, die Anwesenden, und die besprochenen Themen. Bei den zusätzlichen "add sentence" hat es dann aber einige praktische Sätze dabei. Trankript ist nich immer korrekt...
- "Add Sentences" war nicht übersichtlich. Teils war nicht der ganze Satz sichtbar.
- Verstehe Frage nicht: Die Highlights bestehen doch bereits. Ich muss also weder erstellen, selektieren noch versenden!

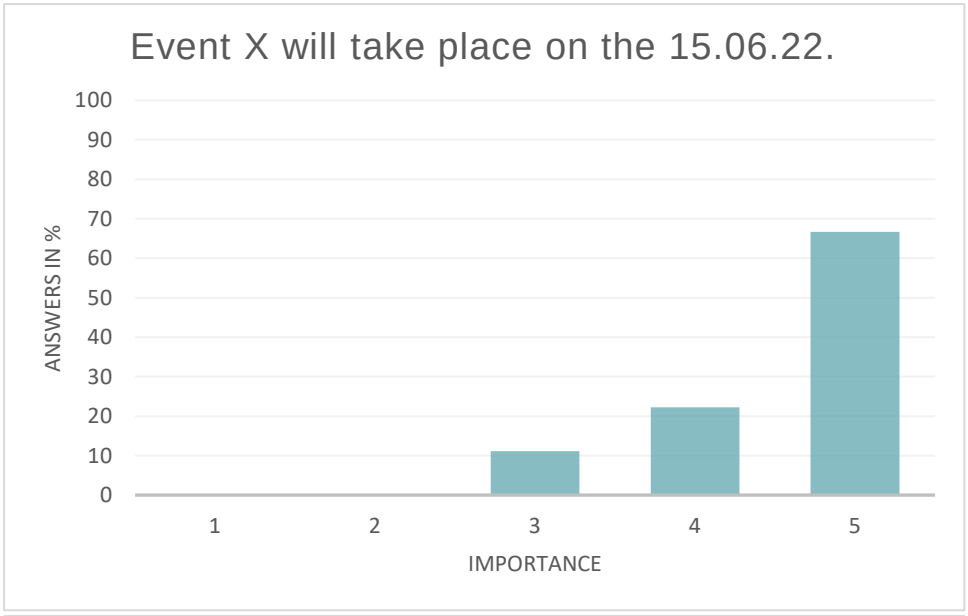
15.6.4 Sentence Relevance Assessment

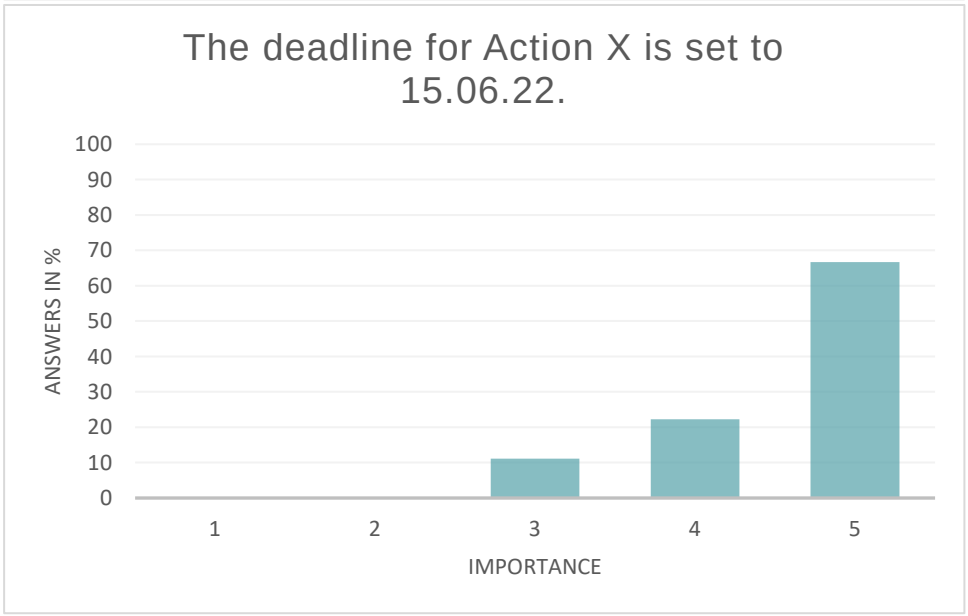


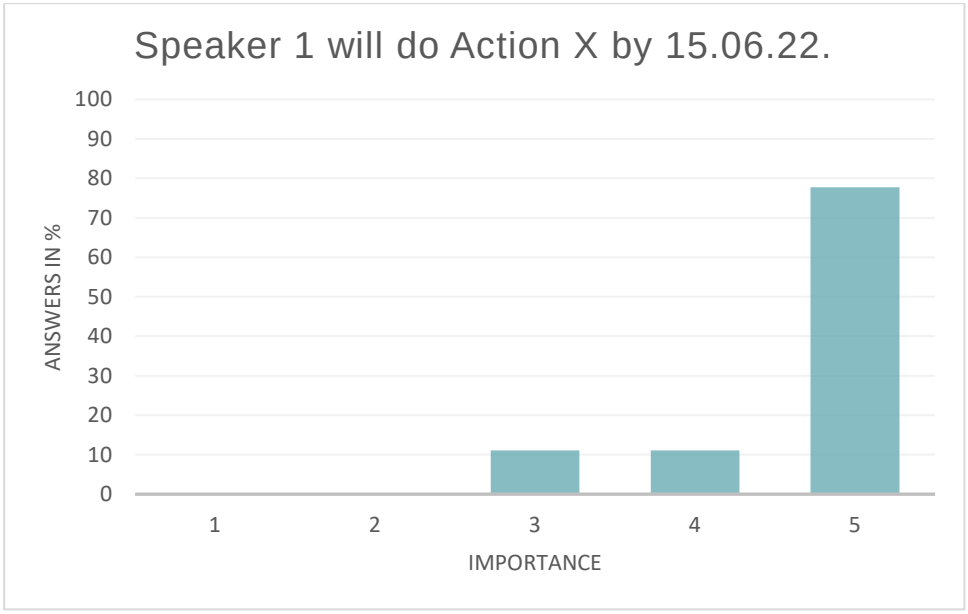


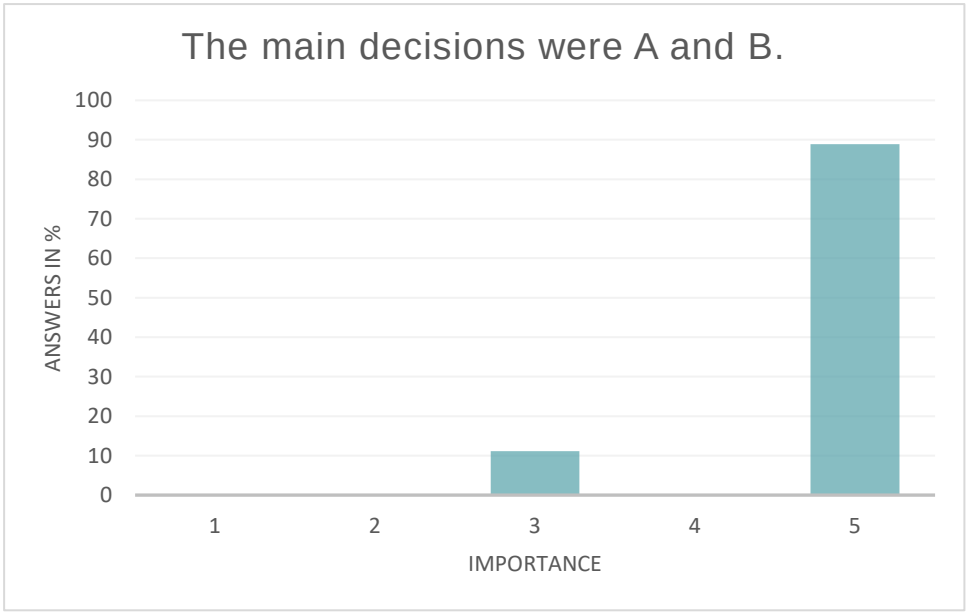


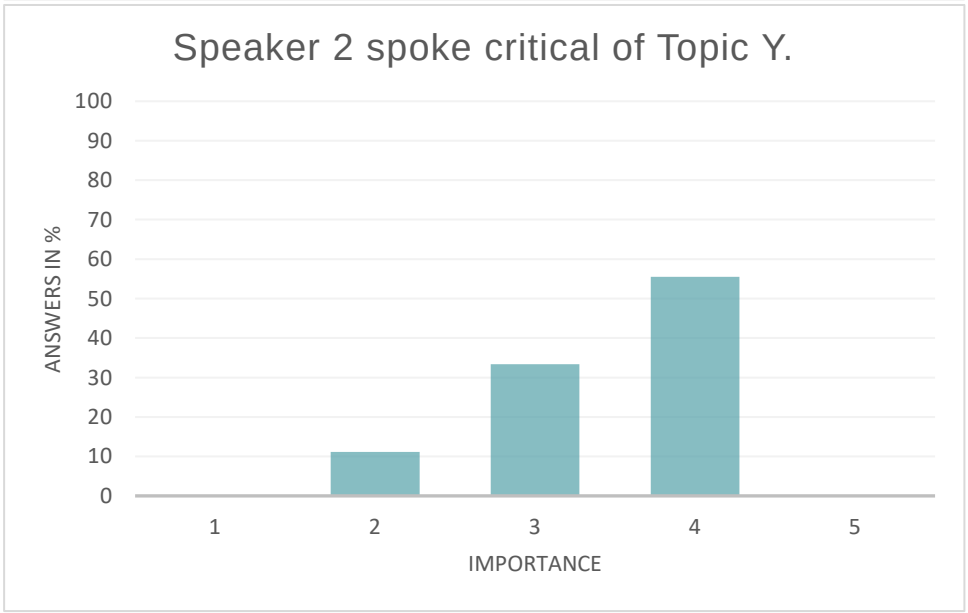

Speaker 1 talked most about Topic X.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE


Event X will take place on the 15.06.22.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE

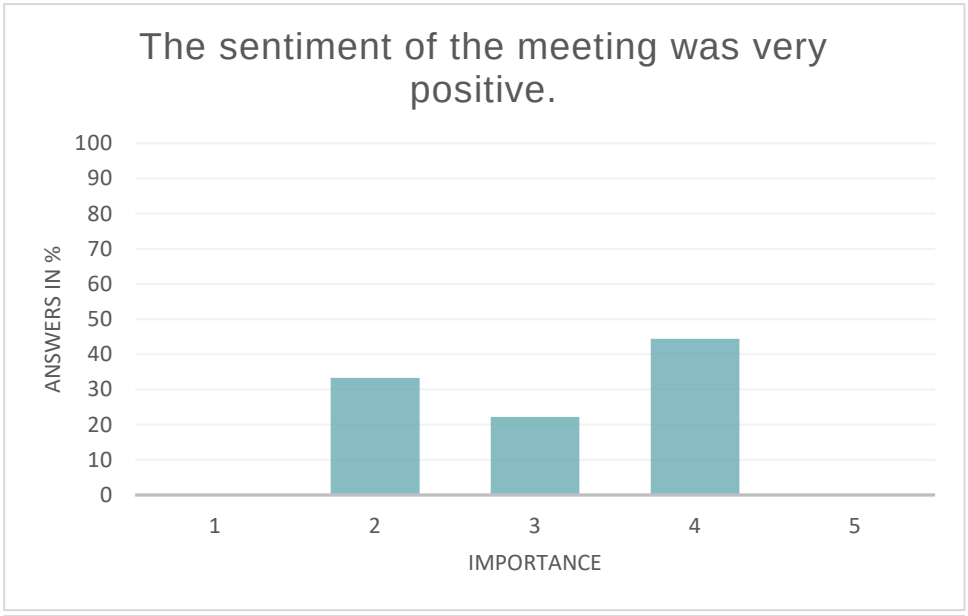

The deadline for Action X is set to 15.06.22.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE

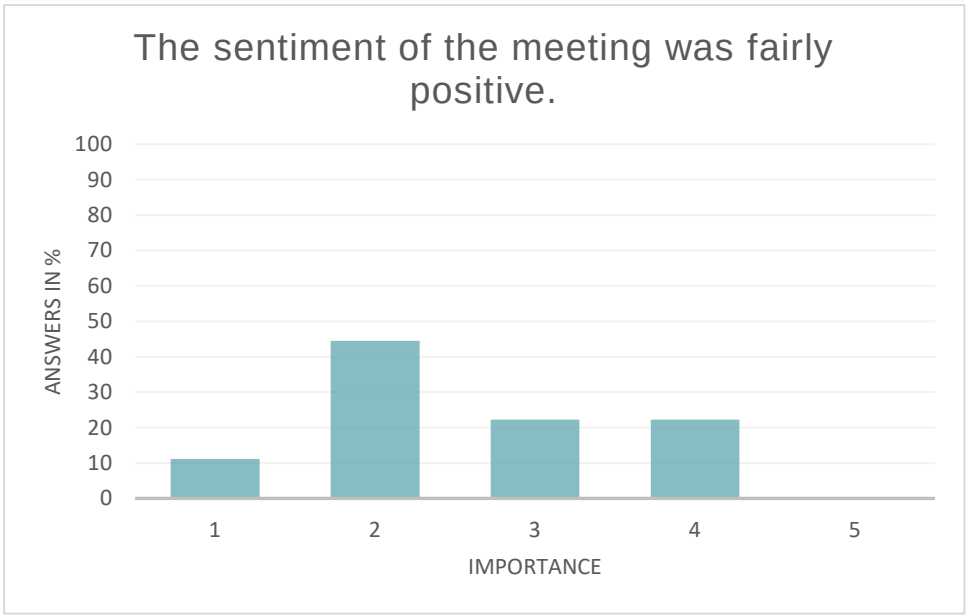

Speaker 1 will do Action X by 15.06.22.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE

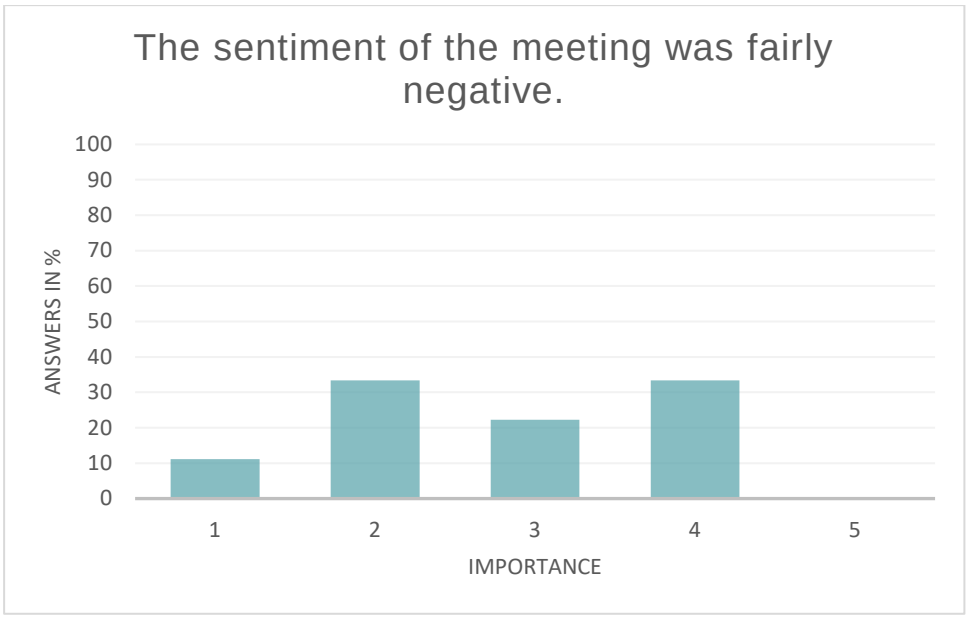

The main decisions were A and B.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE

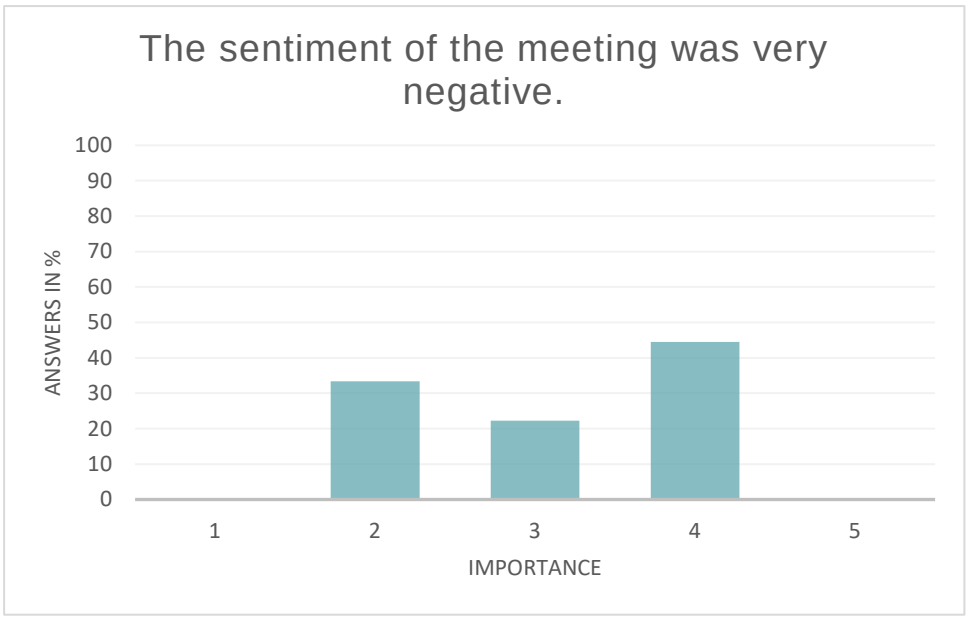

Speaker 2 spoke critical of Topic Y.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE

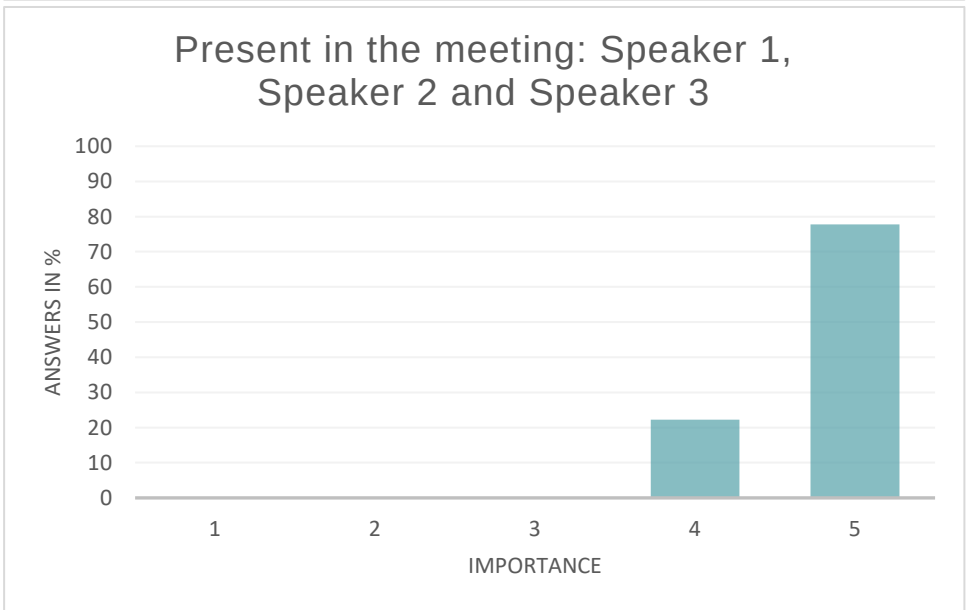

There was a fight between Speaker 1 and Speaker 2 for 3 Minutes.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE

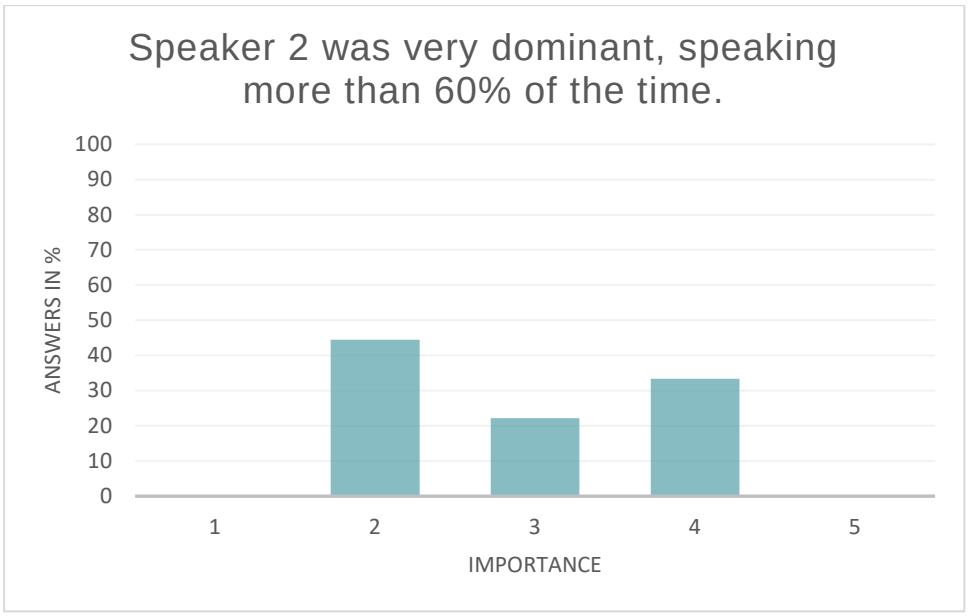

The sentiment of the meeting was very positive.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE

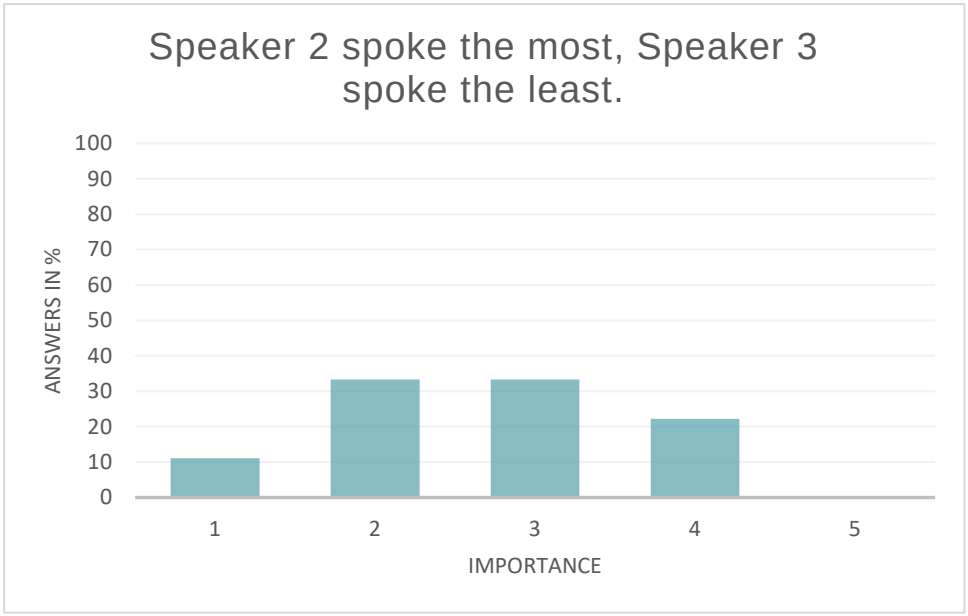

The sentiment of the meeting was fairly positive.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE


The sentiment of the meeting was fairly negative.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE


The sentiment of the meeting was very negative.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE


Present in the meeting: Speaker 1, Speaker 2 and Speaker 3
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE


Speaker 2 was very dominant, speaking more than 60% of the time.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE


Speaker 2 spoke the most, Speaker 3 spoke the least.
ANSWERS IN %
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
IMPORTANCE