Biography Generator

# Aadishree Borkar

*dept. Applied Data Science and Analytics*

*SRH HOCHSCHULE HEIDELBERG*

Heidelberg, Germany

# Muskaan Chhabra

# dept. Applied Data Science and Analytics

*SRH HOCHSCHULE HEIDELBERG*

Heidelberg, Germany

# Ashwith Anand Poojary

# dept. Applied Data Science and Analytics

*SRH HOCHSCHULE HEIDELBERG*

Heidelberg, Germany

# Rakesh Hadne Sreenath

# dept. Applied Data Science and Analytics

*SRH HOCHSCHULE HEIDELBERG*

Heidelberg, Germany

# Lakshith Reddy Baddula

# dept. Applied Data Science and Analytics

*SRH HOCHSCHULE HEIDELBERG*

Heidelberg, Germany

***Abstract*— Biographical interviews are essential resources for humanities research which gives researchers a comprehensive view about individual live yet converting lengthy and complex transcripts into concise summaries is a time-consuming and labor-intensive task. To address this, we propose an automated system which is aimed at generating brief biographies from German-language interview transcripts. By employing advanced natural language processing techniques, which are specifically oriented towards the functions of text summarization and information extraction and utilizing powerful language models like Meta-Llama-3.1-70 B-Instruct, our approach aims to speed up creating biographical summaries while adhering to data privacy regulations such as GDPR. Our goal, through this automated system, is to enable more convenient and straightforward access to biographical pieces. As well as using NLP techniques we not only achieve the accessibility of biographical data but also, we enhance the quality of summarization by capturing life events of an individual. Using language models the biography task is achieved, also this research seeks to streamline the work of archivists and researchers by providing rapid access to critical biographical details.**

**Keywords: Natural Language Processing, Text Summarization, Biography, Information Extraction, Large Language Models.**

I. INTRODUCTION

Biographical interviews stand as invaluable repositories of personal and historical narratives, offering profound insights into the lives of individuals and the broader societal contexts in which they existed. The humanities have long recognized their significance, employing them as foundational sources for research and storytelling. However, the process of transforming these rich, often extensive interviews into concise, accessible biographies is a labour-intensive endeavour, frequently resulting in delays and gaps in archival collections.

The advent of artificial intelligence, particularly advancements in Natural Language Processing (NLP), presents an opportunity to revolutionize this process. By harnessing the power of AI, it becomes feasible to automate the extraction of key biographical

information from interview transcripts, thereby streamlining the creation of biographies. This research delves into the application of AI to this challenge, focusing specifically on German language interviews.

The potential implications of this work are far-reaching. By automating the generation of biographies, we can significantly enhance the efficiency of archival processes, making biographical information more readily accessible to researchers and the public alike. Moreover, this project contributes to the broader discourse on the intersection of AI and the humanities, demonstrating the potential of technology to preserve and disseminate cultural heritage.

The subsequent sections of this report will outline the methodology employed, the specific AI techniques utilized, and the evaluation of the system's performance. Ultimately, this research aims to demonstrate the feasibility and efficacy of using AI to automate biography creation from German language interviews, paving the way for future developments in this field.

1. *Motivation and Problem Statement*

Generating coherent and accurate biographical texts from structured data such as transcripts remains a significant challenge in Natural Language Processing (NLP). Current manual methods of biographical writing require significant time, expertise, and effort, while existing automated approaches often struggle with maintaining narrative coherence, managing large volumes of data, and ensuring factual accuracy. Moreover, biases in input data and the limitations of language models to consistently generate relevant summaries exacerbate these issues.

The need arises for an automated solution that can effectively transform transcript data into well-structured biographies with minimal human intervention. This project aims to address these challenges by leveraging state-of-the-art language models like Llama 3.1 70 B Instruct, combined with prompt engineering techniques and Natural Language Processing (NLP) techniques such as chunking and transcript filtering. The goal is to improve the efficiency, coherence, and contextual relevance of automated biographical generation systems, enabling scalable and accurate narrative creation from raw data.

II. RELATED WORK

The coherent and informative generation of biographies from structured textual data, such as transcripts, is the most attractive factor in Natural Language Processing (NLP) development during the last few years. The recent successes in large language models (LLMs), text summarization, and data extraction techniques have enabled the creation of automated systems proficient in turning raw textual data into meaningful narratives. The following literature review covers the available studies and methodologies that could be relevant to automated biography generation and to the case at hand, with the application of state-of-the-art Large Language Models like Llama 3.1 70 B Instruct.

Biographical text generation represents a coherent narrative of summarizing the key facts about a person's life and achievements. Conventionally, it is a creative task performed by humans, through experience and resource-intensive processes. However, with the advent of Natural Language Processing (NLP), automated systems have been developed based on large-scale language models trained on a huge dataset to generate human-like summaries. Models such as GPT-3, GPT-4, and Llama have revolutionized this field by leveraging deep learning techniques to understand and generate contextually relevant text [1],[2].

The Llama 3.1 70 B Instruct model is specifically fitted with the tasks of instruction execution and represents a quantum leap in this domain. It is a powerful autoregressive model that outshines others for complicated prompts and generates high-quality, coherent text outputs which are contextually appropriate. Llama 3.1, with its 70 billion parameters, is particularly well-suited for summarization tasks due to its enhanced capabilities in handling large chunks of data, maintaining narrative coherence, and adapting its output style to specific requirements [2]. This project leverages the strong points of Llama 3.1 in structured data processing and narrative generation to produce factually correct biographical summaries from transcript data.

1. *Techniques of Transcript Analyses and Data Extraction*

In the creation of these biographies, the quality of the input data determines a lot, in this case, analysis and filtering of transcripts. Techniques such as named entity recognition (NER) and keyword filtering are employed to extract portions of the transcript data that are relevant, which then feed a summarization model**.** Named Entity Recognition **(**NER) helps in identifying important entities such as names, dates, and locations, which are critical to biographical narratives [3]. The filtered selection of transcript rows at specific prefixes of the 'Sprecher' field makes it exclusive to the most relevant content, hence increasing the overall quality of the generated biographies.

Most of these phases are important for optimizing the performance of the model, including cleaning, normalizing, and segmenting data into manageable chunks. Segmenting text is helpful in controlling the size of the input and keeping the summarization process coherent in the whole biography. The Llama 3.1 model's ability to handle chunked input effectively allows it to maintain thematic continuity, a common challenge in biographical text generation [4].

1. *Enhancing Text Coherence in Generated Biographies*

Coherence in generated texts is important to make biographies sound as natural as possible and not computer-generated. One notable study by Horacek and Buchberger [6] highlights methods to enhance text coherence specifically in the context of generating German texts. What the authors have been able to show is that through paraphrasing and establishing contextual links, cohesion between sentences is maintained. They go into a number of key techniques explored, including paraphrasing to remove ambiguity, and handling complex structures through contextual links and semantic network transformations.

The insights from this study underscore the importance of addressing coherence challenges when generating biographical texts, which are directly applicable to the goals of this project. By incorporating principles from Horacek and Buchberger’s work, the project aims to ensure that the generated biographies are not only accurate but also narratively consistent and engaging [6].

1. *Automated Biography Generation with Llama 3.1 70 B Instruct*

The use of Llama 3.1 70 B Instruct distinguishes this project by integrating a state-of-the-art model specifically optimized for instruction-based tasks and summarization. Unlike earlier models that may struggle with maintaining coherence over long narratives, Llama 3.1’s architecture allows it to generate text that remains contextually consistent even when processing large, segmented transcripts. This capability is particularly valuable for biography generation, where maintaining a logical and chronological flow of information is essential.

Studies have shown that models like Llama 3.1 outperform other Large Language Models (LLMs) in specific tasks such as summarization, question answering, and instructional prompts, making them ideal for projects that require precise and context-aware text generation [2]. By leveraging Llama 3.1, this project benefits from improved output quality, reduced manual editing, and enhanced narrative coherence compared to traditional or less advanced automated systems.

1. *Literature Gaps and Contribution of Llama 3.1*

While a lot of progress has been made, several challenges still lie ahead of automated biography generation with respect to mitigating biases in training data, preserving factuality, and overcoming multilingual limitations. Llama 3.1 addresses these gaps by integrating instruction tuning, allowing it to adhere more closely to prompts that define factual constraints and style requirements and, subsequently, decrease errors present in the biographies generated [5].

With other research focusing more on GPT-3 and BERT, the deployment of Llama 3.1 in the current project underlines further its scalability and adaptability across a variety of transcript formats.

III. METHODOLOGY

1. *Overview*

In this project, we developed an automated system to generate concise biographical summaries from interview transcripts using large language models (LLMs). The core of the methodology involves breaking down raw transcript data into manageable chunks, applying filtering techniques to extract relevant biographical details, and using LLMs like Llama 3.1 70B Instruct to generate coherent biographies.

The workflow begins with file handling and text extraction, where transcripts are segmented based on speaker roles (interviewee and interviewer) to isolate meaningful content. This is followed by chunking the transcript data into smaller sections, which are processed sequentially through the LLMs. Early experiments with models such as Mistral 7B Instruct v0.2 and Llama 2 Chat 7B revealed challenges with verbosity and information hallucination, leading us to refine the process and adjust chunk sizes.

In subsequent iterations, we integrated Llama 3.1 70B Instruct, which improved the narrative coherence and output quality. Key steps in the methodology also include prompt engineering to guide the model’s focus and implementing techniques like sentence chunking to align the text structure with the language flow, ensuring more accurate and cohesive biographies. The following section outlines the specific steps, techniques, and models used in the biographical generation process.

1. *Read the document*

The first step in the process involves the user uploading a Microsoft Word (docs), Portable document format (pdf) or Comma-Separated values (csv) files through the web page, after which the system reads the file and extracts the text from it. Various python functions are used to handle the reading process. BytesIO method is used to ensure that the file stream is handled correctly.



Fig. 1. Code snippet of reading

In order to check for exceptions during the read process we processes the content into a pandas DataFrame. The read\_csv function handles reading the uploaded CSV file, and an error message is displayed in case the file is not able to be read, allowed\_file function is used to validate the CSV file extensions. The statement file.stream.seek(0) ensures that the reading process begins from the start of the file. In this segment we check for file validity, manage the file stream, and handle errors that may arise during the file reading process. The design makes it a robust and reliable solution for reading structured data from external files.

*C. Process the data*

Since usually the interview transcripts are large, we split the content into two components, the interviewee and the interviewer. Doing this will make the transcript simpler to understand and process for the Large Language Model (LLM). First, we filter rows based on speaker prefix. The extract\_rows\_with\_sprecher function focuses on extracting rows of the speaker's name which is in the "Sprecher" column of the transcript begins with a specified prefix, such as "IP\_" or "INT\_". This enables the selective extraction of speaker-specific transcripts based on these identifiers, making the data retrieval process more targeted. To eliminate any rows where the "Sprecher" field is empty or null we use df.dropna(), this is done so no irrelevant or incomplete data gets passed as input thus ensuring that only meaningful rows are processed. Once the relevant rows are filtered, the transcripts are extracted, concatenated, and returned as a combined string. This is achieved through the transkript\_to\_string() function, which compiles all the selected transcripts into a single coherent output and ensures that only the most relevant speaker data is returned for further analysis.

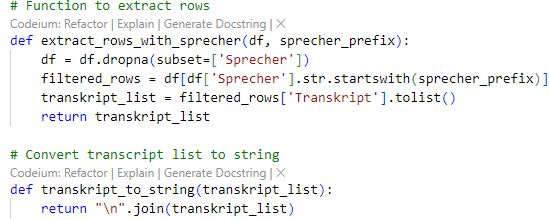


Fig. 2. Code snippet of preprocessing

D*. Chunking the document*

Once we read and process the document, the next key step is to chunk the text. By breaking the text into manageable sections, it becomes easier to work with the large volume of content, allowing the Large Language Models to produce more accurate analysis. In our initial approach, we employed a word chunking strategy, starting with relatively small chunks of 700 to 1,000 words. Using this method, processing a sample document consisting of approximately 7,000 rows took around 5 to 6 minutes. Though this approach was a manageable way to handle data, the processing time required was notably long. To resolve this, we increased the chunk size to 2,000 to 4,000 words. This adjustment resulted in reducing the processing time by half, thereby improving efficiency significantly. Since the chunk sizes were relatively small there were more chunks being produced, this led to more processing time. We noticed that as the number

of chunks increased so did the size of the output and we were also getting multiple biographies. This led us to increase the chunk size to 20000-25000 words which brought the processing time to about 1 minute. However, a limitation emerged: the token limit restricted the size of the chunks that could be processed effectively. We also noted that the initial chunks produced more detailed outputs compared to subsequent ones, regardless of the model being used. Additionally, the output contained superfluous information, including extra dates that were irrelevant to the biographies, as well as a higher volume of unnecessary data. As a result of these issues, we opted to switch from word chunking to sentence chunking. The divide\_into\_chunks function uses sent\_tokenize() to divide the transcript into individual sentences. We set the chunk limit parameter as 70000 characters as it is an ideal length for the chunk to be input into the Large Language Model, the limitation of the current Large Language Model used is around 85000 characters. This also aligns the text with the flow of the language and ensures it is segmented naturally. Next, word\_tokenize() is employed to calculate the word count for each sentence, helping to determine how many sentences can be combined within a chunk. This is done so each chunk adheres to the defined word limit for ease of processing. The function tracks the cumulative word count as sentences are grouped. When the cumulative word count surpasses the set threshold, the current group is marked as a chunk and appended to a list. This process is repeated until the entire transcript is broken down into manageable chunks.

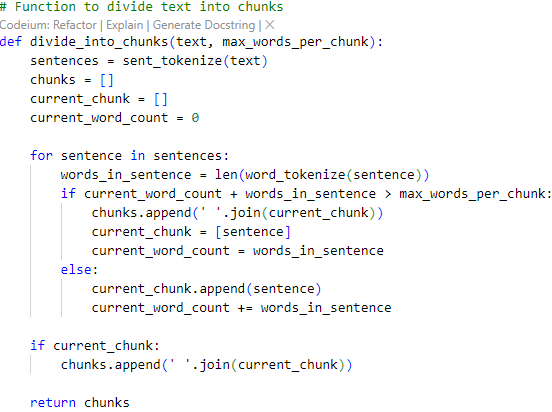


Fig. 3. Code snippet of chunking

E*. Biography*

Next, the chunks are passed sequentially into the Large Language Model. We use Llama 3.1 70 B, it is the most recent and capable model released to date. We are accessing the model through an API call using TogetherAI’s large language model server to generate summaries. TogetherAI offers an API service that allows us to utilize their server for running any available large language models. To get started, we first need to create a TogetherAI account and generate an API key, which can be done on the profile/settings/api keys page. With the API key in hand, we can begin using their services. For our tasks, we use the “invoke” function from the “Together” library to request the desired model on the TogetherAI server via an API call. During this process, we prompt the model to create a biography of the input text.

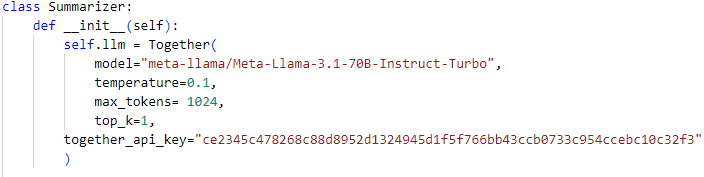


Fig. 4. Code snippet of using TogetherAI library to use Llama 3.1 70 B

We prompt the model to generate biography for the intended person in the input text that is Interviewee. The generate\_biography method is the core functionality, responsible for sending chunks of transcript data to the LLM along with a prompt aimed at generating a biography. We specify parameters such as temperature, max\_tokens, and top\_k, which influence the LLM's response and output quality to integrate with Together API. To guide the LLM in structuring the biography appropriately we use an English-Language prompt that gives output in German. The prompt instructs the Large Language Model to give information related to formative or significant life events with years mentioned in the interviewee's childhood, adolescence and adult life, date and location of birth, parents and siblings, their names, backgrounds, relationships and any relevant details and biography is coherent, chronological, detailed, and presents a well-rounded view of the interviewee's life journey. The invoke\_with\_retry() function is employed to handle multiple attempts to invoke the LLM. It uses exponential backoff to manage retries, ensuring robust handling of temporary issues or failures during the LLM invocation process. As each chunk is created it is passed into the model with the prompt, in the output if there are any abruptly ending sentences they are cut out and the output is saved in a file.

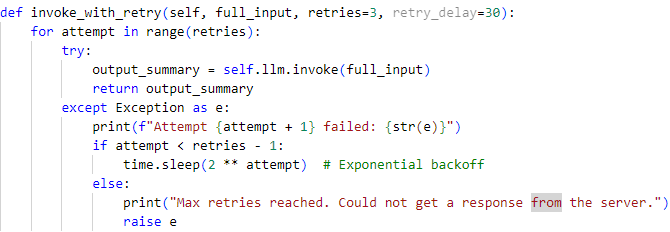


Fig. 5. Code snippet of invoking retry function

The output of the next chunk goes through the same process and gets appended to the previous output in the file. This process is repeated continuously until all the chunks are passed. The result in the file is the biography. Though we have also tried multiple other models namely Mistral-7B-Instruct-v0.2, Llama 2 70 B, Llama 3 8 B yet Llama 3.1 70 B model proved to be the best for the task at hand.

1. F. *Alternative models tested*

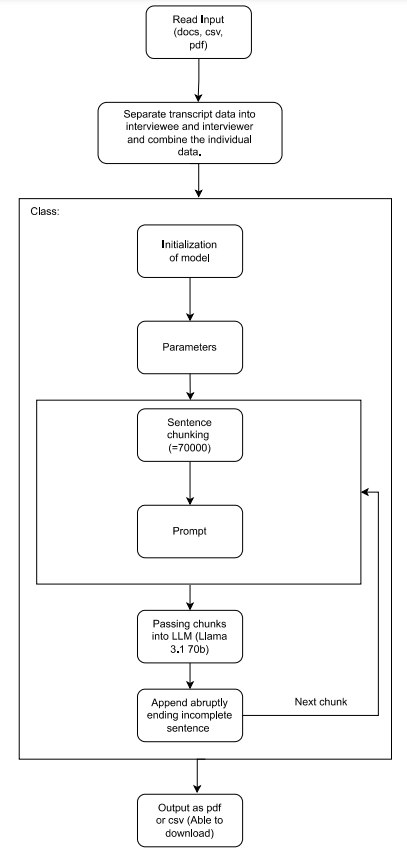
Initially, we integrated two different Large Language Models (LLMs) into our biography generation pipeline: Mistral 7B Instruct v0.2 and Llama 2 Chat 7B. The overarching goal was to automate the generation of biographical summaries from transcripts, while maintaining accuracy, coherence, and 

Fig. 6. Flowchart of architecture

conciseness. Our process followed a two-step model: first, segmenting the input document into chunks of raw text, then processing these chunks through the models to summarize and generate biographies.

In the first iteration of our model, the transcripts were divided into large chunks of words without any prior preprocessing or filtering. These chunks were then passed onto the Mistral 7B Instruct v0.2 model. The prompt given to Mistral 7B was designed to capture all significant events and details of the interviewee, condensing this information into a coherent summary. However, several issues arose during this phase:

1. Redundancy: Mistral 7B often produced summaries that were verbose and repetitive, leading to unnecessarily lengthy texts.

2. Lack of abstraction: The summaries generated by Mistral 7B were often too detailed and included trivial information that diluted the focus on key biographical events.

3. Scalability issues: As the input document size increased, the summary produced by Mistral grew disproportionately large, making it difficult to manage in later stages of the process.

Following this step, the summary generated by Mistral 7B was passed to Llama 2 Chat 7B, whose task was to transform the summary into a structured biography, constrained to approximately 500 words. However, Llama 2 Chat 7B encountered several challenges:

1. Hallucination of information: The model often generated additional data not present in the input summary or transcript. This led to inaccuracies, as biographies included false or speculative details.

2. Limited focus: The model occasionally struggled to maintain a coherent structure, leading to biographies that were either too brief or overly focused on irrelevant aspects.

The results from this combined pipeline were inconsistent, with summaries and biographies that did not reliably reflect the content and structure of the source transcripts. The gold standard we set—producing concise, structured, and accurate biographies of around 500 words—was not met by either model in this configuration.

In a subsequent attempt to address these issues, we modified our methodology by switching from word chunking to sentence chunking. This adjustment allowed the model to process smaller, more manageable pieces of text, which we hypothesized would reduce redundancy and improve coherence. We also refined the prompts to guide the model more effectively in extracting only the most relevant biographical information.

While this approach marginally improved the output from both Mistral 7B and Llama 2 Chat 7B, the core problems persisted. Mistral 7B still produced lengthy summaries, and Llama 2 Chat 7B continued to generate unreliable or fabricated information. The outputs, though slightly improved, were still far from the gold standard of coherent, accurate biographies.

In a further iteration, we replaced Mistral 7B Instruct v0.2 with Mixtral-8x7B-v0.1, another LLM with a similar architecture but optimized for different use cases. However, the results from Mixtral mirrored those of Mistral 7B. The model continued to produce verbose and somewhat redundant outputs, with summaries that were overly detailed and less focused on the key events necessary for a biography. While there were marginal gains in processing speed, the overall quality of the output remained suboptimal, with no significant improvements over the previous iteration.

In the final stage of the project, we integrated the more powerful Llama 3.1 70B Instruct model. This model, designed for instruction-based tasks, was expected to handle the complexities of summarization and biography generation more effectively. The model generated summaries and biographies that were more coherent and fluid than those produced by earlier models. Llama 3.1 better identified and extracted the key biographical events from the transcripts, avoiding trivial or repetitive details.

However, despite these improvements, some limitations remained. The token limit of Llama 3.1 70B was relatively small, which sometimes resulted in the loss of precise details, such as specific dates and minor but important biographical information. While the model performed better than its predecessors, there were still occasional inaccuracies in the generated biographies, particularly when dealing with complex or lengthy transcripts. We explored the possibility of running Llama 3.1 70B locally, but due to the extensive computational power required, local deployment was not viable. The installation and execution demand exceeded the available resources, making cloud-based deployment the only feasible option.

Throughout these iterations, the output of each model was compared against a predefined gold standard, which consisted of manually written biographies of approximately 500 words, focusing on key events, dates, and accomplishments of the interviewee. The gold standard was used to assess the coherence, accuracy, and conciseness of the generated biographies.

1. Mistral 7B Instruct v0.2 produced overly detailed and verbose summaries, deviating significantly from the gold standard. The resulting biographies were incoherent and included irrelevant information.

2. Llama 2 Chat 7B improved on the structure, but often introduced fabricated details, causing significant deviation from the factual accuracy required by the gold standard.

3. Mixtral-8x7B-v0.1 yielded similar results to Mistral, with only slight gains in summarization speed but no significant improvements in content quality.

4. Llama 3.1 70B Instruct came closest to meeting the gold standard, generating biographies that were more coherent and relevant, though still occasionally lacking in detail and accuracy due to token size constraints.

In conclusion, while Llama 3.1 70B Instruct provided the best performance in terms of narrative coherence and adherence to the target word count, it still fell short of consistently meeting the gold standard. The findings highlight the ongoing challenge of balancing model complexity with computational efficiency, especially in tasks that require high accuracy and factual consistency. Further improvements could be achieved by refining preprocessing methods, addressing token limitations, and exploring more scalable deployment options.

G. *Frontend*

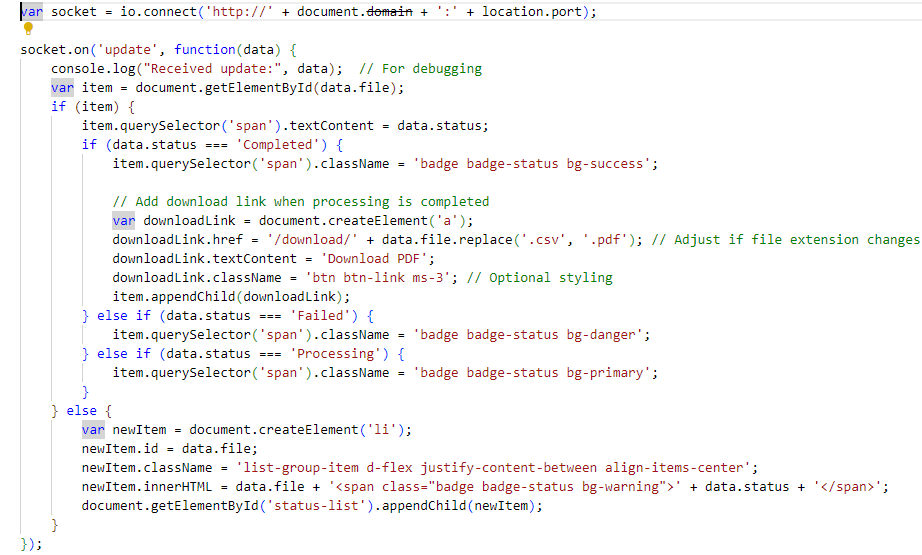
The design implementation for user interface was done using Cascading Style Sheets (CSS) and Hypertext Markup Language (HTML). On the user interface, the user is given a choice to either upload a file (csv or pdf) or a folder. The interaction with input documentation takes place using JavaScript code. A socket connection is established using io.connect to the server's origin. The JavaScript code leverages a socket connection to update the user interface dynamically as processing statuses change for specific files. When the server sends an update, an event listener is triggered. The status appearance in the user interface changes as per the trigger, it shows either 'Processing', 'Completed' or 'Failed'. If the status of the file is marked as 'Completed', the script creates a downloadable link to the PDF version of the file. To integrate frontend and backend we utilize Flask Web application that handles file uploads and processes them asynchronously. 

Fig. 7. Code snippet of JavaScript

For each uploaded file, a new thread is created. This allows the file processing to happen in the background, without blocking the web server or making users wait for the process to complete. After initiating the file processing, the server immediately returns a JSON response 'Files are being processed', that informs the user that their files are being handled in the background.

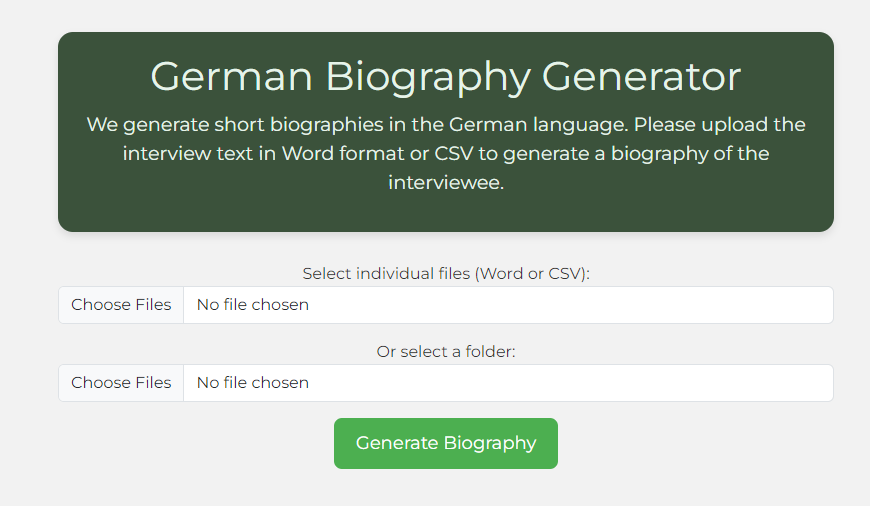


Fig. 8. Output of frontend

The @app.route('/download/<filename>') route allows users to download processed files. When a user accesses the /download/<filename> URL, the server locates the processed file in the specified directory and sends it to the user. We kept the user interface simple, direct, user-friendly and scalable as well.

H*. Output*

Once the biography is generated, it is saved as a PDF file using the save\_text\_to\_pdf function. We have the standard output in the text file from which we can compare to the output we received in pdf format. The main differences we have observed are-

1. Detail of output: The gold standard output focuses on information, and it is precise and concise while the output we have received focuses on narrative detail and includes the events that are important.
2. Observation- The gold standard output provides direct communication and facts while the output we received includes emotional tones.
3. Structure- The standard output structure includes a list of life events in an order like education, job or personal life while the output we receive provides the dates of life events and important events.

The function utilizes the FPDF library to create and format the PDF. The biography text is split into lines to ensure proper formatting, and then written into the PDF. After the PDF is created, it is saved in a designated directory and can be downloaded by users through the /download route. Key aspects of this process include initializing the PDF object with FPDF(), using pdf.multi\_cell() to make sure the text wraps correctly within the PDF, and saving the file in the output\_pdfs directory. The saved PDF file retains the same name as the original CSV file but with a .pdf extension. Thus, the biography can be downloaded as a pdf file.

IV. RESULTS AND DISCUSSIONS

In the initial implementation, file handling and data processing were prone to errors, particularly with large interview transcripts and improperly formatted files. The system lacked selective filtering, resulting in inefficient data processing. Additionally, the word-based chunking method led to slow processing times (5–6 minutes) and redundant, fragmented outputs, often producing multiple biographies with irrelevant data. Several improvements were made through file handling that use using advanced Python functions that enhance error handling resulting in a more reliable system. Selective extraction of relevant interview data streamlined processing, focusing only on important sections. During data processing, a targeted approach was employed to extract relevant speaker-specific transcripts, focusing on the "Sprecher" field in interview documents. The selective extraction, filtering of incomplete rows, and concatenation of transcripts into a combined string for analysis helped streamline the data, ensuring the large language model (LLM) receives accurate and focused input. This methodology proved effective in structuring complex interview data. By switching from word-based chunking to sentence-based chunking with larger chunks we managed to cut processing times by half, with most documents now being processed in about a minute. This also meant fewer chunks, resulting in more cohesive and consistent outputs. By aligning the chunks with natural sentence breaks, the quality and coherence of the generated biographies improved significantly, reducing redundancy and enhancing accuracy. There was a significant improvement in both accuracy and output quality due to the usage of Llama 3.1 70 B from TogetherAI’s API. Apart from significantly improved processing speed, it generated more accurate biographies and produced far fewer redundant outputs as well. Despite some issues with token limits, the generated biographies are more cohesive, detailed, and relevant compared to the initial results. The model’s ability to capture important life events and structure them chronologically was markedly improved.

Generative data from Artificial Intelligence: While working with Llama 2 Chat 7 B model, we faced the issue of generated data in the output. This has the potential to be a huge problem as biographies are fundamentally based on factual data. AI models are trained on vast amounts of internet data. This data, while rich in information, contains both accurate and inaccurate content, as well as societal and cultural biases. Since these models mimic patterns in their training data without discerning truth, they can reproduce any falsehoods or biases present in that data. They are designed to predict the next word or sequence based on observed patterns. Their goal is to generate plausible content, not to verify its truth. As a result, they might produce content that sounds plausible but is inaccurate. Future innovations and developments tend to this bias and limitations.

V. CONCLUSION

In this project, we aimed to automate the generation of biographical texts from structured transcripts using advanced language models, particularly the Llama 3.1 70 B Instruct model. The focus was on transforming raw textual data into coherent and meaningful narratives, while ensuring the relevance and accuracy of the output. The Llama 3.1 70 B model demonstrated strong capabilities in handling instruction-based summarization tasks, delivering coherent and contextually accurate biographical narratives. By incorporating techniques such as transcript filtering and chunking for large data processing, the project was able to enhance the overall quality and coherence of the generated biographies. Despite the success of the approach, there were some limitations, particularly in managing biases within the input data and maintaining factual accuracy throughout the biographies. Additionally, challenges remain in supporting diverse languages and optimizing the model for even more complex biographical narratives. This project contributes to the field of automated biography generation by demonstrating the practical application of a state-of-the-art large language model, Llama 3.1 70 B, in structured data processing. Future work could explore expanding the system’s multilingual capabilities, refining the model's ability to fact-check in real-time, and further improving coherence in longer, more complex narratives. Additionally, integrating more advanced data filtering techniques or experimenting with alternative language models could yield even better results. In conclusion, the successful integration of the Llama 3.1 70 B model into this project has laid the foundation for scalable, automated biography generation, offering valuable insights into how large language models can be leveraged to enhance textual coherence and narrative quality.

VI. References

[1] T. Brown, B. Mann, N. Ryder, et al., “Language Models are Few-Shot Learners,” \*arXiv preprint\*, arXiv:2005.14165, 2020.

[2] H. Touvron, L. Martin, K. Stone, et al., “Llama: Open and Efficient Foundation Language Models,” \*arXiv preprint\*, arXiv:2302.13971, 2023.

[3] G. Lample, M. Ballesteros, S. Subramanian, et al., “Neural Architectures for Named Entity Recognition,” \*arXiv preprint\*, arXiv:1603.01360, 2016.

[4] C. Raffel, N. Shazeer, A. Roberts, et al., “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,” \*arXiv preprint\*, arXiv:1910.10683, 2020.

[5] J. Wei, X. Wang, D. Schuurmans, et al., “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,” \*arXiv preprint\*, arXiv:2201.11903, 2022.

[6] H. Horacek and E. Buchberger, “Achieving Text Coherence in a Generator for German Texts,” in \*Proceedings of the International Conference on Natural Language Processing\*, pp. 831-836, 1986. Available: 10.1007/978-94-009-4634-7\_107.

[7]https://mitsloanedtech.mit.edu/ai/basics/addressing-ai-hallucinations-and-bias/