SRH HOCHSCHULE HEIDELBERG

MASTERS THESIS

AI-Driven Medical Chatbot for Medical Students and Patients: An LLM and Embedding-Based Approach

Author:

NATARAJU TUMAKURU SHREESHYLESHA

Matriculation Number: 11027742

Supervisors:

PROF. DR. CHANDNA SWATI

A thesis submitted in fulfilment of the requirements for the degree of
Master Of Science In Applied Data Science and Analytics

in the

School of Information, Media and Design

24th February, 2025

Declaration of Authorship

I, **Nataraju Tumakuru Shreeshylesha**, declare that this thesis titled, *"AI-Driven Medical Chatbot for Medical Students and Patients: An LLM and Embedding-Based Approach"* and the work presented in it is my own. I confirm that this work submitted for assessment is expressed in my own words and is my own. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are appropriately acknowledged at any point of their use. A list of the references employed is included.

- This work was done wholly for a research degree at this University.
- It has been clearly stated in this work if any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution.
- Where I have consulted the published work of others, credit is always given to the concerned work.
- Where I have quoted from the work provided by others, the source is always given unless it is entirely my work.
- I have acknowledged all main sources that supported me and helped during this work.

Signed: ______________________________

Date: ______________________________

Acknowledgements

I am deeply grateful for the support and guidance of numerous individuals who have contributed to the completion of this thesis, and I would like to express my heartfelt appreciation to all of them.

First and foremost, I would like to extend my sincere gratitude to my supervisor, Prof.Dr.Chandna Swati, for her invaluable guidance, expertise, and unwavering support throughout this research. Her mentorship has played a crucial role in shaping my ideas and refining my work.

I am also grateful to the faculty and staff of the School of Information, Media and Design at SRH Hochschule Heidelberg for their continuous encouragement and for providing an enriching environment for research and learning.

Special thanks to my colleagues and friends for their motivation, insightful discussions, and unwavering support. Their camaraderie has made this journey more fulfilling and enjoyable.

I am profoundly indebted to my family for their unconditional love, patience, and encouragement throughout this journey. Their unwavering belief in me has been my greatest source of strength and perseverance.

Lastly, I extend my gratitude to everyone who directly or indirectly contributed to the successful completion of this thesis. Your support and encouragement mean the world to me.

Abstract

Artificial intelligence (AI) is rapidly transforming the landscape of medical education and patient support. This thesis proposes an AI-driven medical chatbot system designed to serve two main user roles—medical students and patients—via distinct modes of operation. For medical students, the system provides an interactive knowledge platform with detailed explanations of complex topics, whereas for patients, it offers symptom analysis, healthcare provider recommendations, and even autonomous appointment booking through Agentic AI capabilities.

The architecture comprises several key components. First, textual data (such as PDFs of medical articles or learning materials) undergoes text extraction and chunking before being transformed into embeddings by a dedicated model. These embeddings are then indexed in a vector database, enabling efficient similarity-based retrieval. When a user submits a query, the system identifies and retrieves the most relevant text segments and subsequently augments the prompt for a Large Language Model (LLM), which can be tailored to specific roles: a learning AI assistant (e.g., Llama 2, 13B) for medical students and an agentic diagnostic AI (e.g., Mistral 13B) for patients.

In student mode, the chatbot focuses on explaining medical concepts and providing in-depth, evidence-based knowledge. In patient mode, the chatbot employs symptom analysis to generate preliminary diagnostic suggestions, seamlessly directing users to the HealthCare Provider Recommendation System. Harnessing its Agentic AI feature, the system can autonomously schedule appointments if necessary, reducing administrative overhead and expediting patient care. The entire process is orchestrated through a secure login mechanism, ensuring that only authorized users can access specific functionalities suited to their roles.

Experimental evaluations draw upon real-world medical texts and feedback from both medical students and healthcare practitioners. Early results indicate that this approach effectively balances accurate medical guidance, educational depth, and practical utility in assisting patients. Future enhancements include expanding multi-language support, improving the LLM's medical domain expertise, and integrating real-time health data streams to strengthen diagnostic accuracy.

Keywords: AI-driven Medical Chatbot, Role-based LLM, Patient Symptom Analysis, Healthcare Provider Recommendation, Autonomous Appointment Booking, Embedding Model, Vector Database

Contents

List of Figures

List of Tables

Listings

Abbreviations

CTM	Correlated Topic Model
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
PLSA	Probabilistic Latent Semantic Analysis
NMF	Non-Negative Matrix Factorization
TF-IDF	Term Frequency-Inverse Document Frequency
SVD	Singular Value Decomposition
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
HAC	Hierarchical Agglomerative Clustering
BERT	Bidirectional Encoder Representations from Transformers
NLP	Natural Language Processing
BIRCH	Balanced Iterative Reducing and Clustering Hierarchies
BOW	Bag-of-words
GloVe	Global Vectors for Word Representation
ML	Machine Learning
SBERT	Sentence-Bidirectional Encoder Representations from Transformers
GBERT	German-Bidirectional Encoder Representations from Transformers
T5	Text-To-Text Transfer Transformer
RoBERTa	Robustly Optimized BERT
UMAP	Uniform Manifold Approximation and Projection
PCA	Principal Component Analysis
T-SNE	T-Distributed Stochastic Neighbor Embedding
LLM	Large Language Models
GPT	Generative Pre-trained Transformer
LLaMA	Large Language Model Meta AI
JSON	JavaScript Object Notation

Chapter 1. Introduction

1.1 Introduction

Artificial Intelligence (AI) has become increasingly integral to modern healthcare and medical education, offering the promise of both better patient outcomes and more effective training for medical students. Traditionally, patient-facing solutions have relied on either rule-based systems or static information portals, often lacking deeper contextual understanding, adaptability, and the ability to perform autonomous tasks. Simultaneously, educational tools in medicine have mainly taken the form of passive content repositories or limited interactive platforms, providing a less-than-optimal experience for learning complex medical concepts.

This thesis addresses these limitations by proposing a role-based **AI-driven medical chatbot** capable of serving two distinct user groups:

- **Medical Students:** An interactive learning environment where students can ask in-depth questions about medical topics and receive dynamic, evidence-based responses.
- **Patients:** A user-friendly interface for symptom assessment, healthcare provider recommendations, and automated appointment scheduling.

The system's architecture harmonizes several advanced technologies:

1. **Data Ingestion and Chunking:** Relevant medical texts (such as PDFs, articles, or curated training materials) are first broken into smaller, semantically coherent segments to facilitate granular retrieval and reduce extraneous information.
2. **Embedding and Vector Storage:** Each chunk is converted into a numerical embedding via a specialized model (e.g., a general LLM-based encoder). These embeddings are then indexed in a *vector database*, allowing high-speed similarity searches to quickly identify the text segments most relevant to a user's query.
3. **Retrieval-Augmented Generation (RAG):** When a user submits a query, the system retrieves the top-matching chunks from the vector database. The large language model (LLM) then leverages these retrieved segments as context, enhancing the accuracy and trustworthiness of its generated responses.

4. **Role-based Large Language Models:** The chatbot differentiates between "student mode" (providing detailed medical knowledge and conceptual explanations) and "patient mode" (offering symptom-based advice and potential next steps). This role-specific approach ensures *in-depth* responses for students and *accessible* guidance for patients.
5. **Agentic AI for Appointment Booking:** The system integrates an *agentic* component that allows it to autonomously schedule or reschedule appointments. This feature reduces administrative overhead for providers and streamlines patient journeys, potentially leading to faster access to care.

The proposed chatbot thus spans a wide range of capabilities—from deep academic support for medical students to practical, real-world patient solutions. By uniting *embedding-based retrieval*, *role-aware LLMs*, and *autonomous scheduling*, the architecture tackles existing gaps in both user groups' experiences: medical students gain a dynamic tutor that can recall and contextualize complex material, while patients receive an accessible, on-demand assistant that not only provides preliminary information but also handles logistical tasks like finding the right specialist and booking an appointment.

Furthermore, ensuring a robust, user-centric design requires addressing core challenges. These include:

- **Data Quality and Scope:** The medical knowledge base must be reliable, up-to-date, and appropriately chunked to avoid misinformation.
- **Model Accuracy and Safety:** Large language models can exhibit "hallucination" or produce erroneous conclusions; thus, retrieval of verifiable source chunks is critical.
- **Ethical and Regulatory Considerations:** Patient data privacy, informed consent, and adherence to healthcare regulations remain paramount, particularly in managing personal health information and scheduling details.
- **User Experience Across Roles:** The interface must distinguish clearly between student and patient usage, providing advanced conceptual detail to students while simplifying medical jargon for patients.

Through systematic experimentation, including real-time feedback from both students and healthcare providers, this thesis evaluates the effectiveness of the proposed chatbot architecture in meeting these goals. The subsequent chapters detail the underlying technologies, design methodologies, system evaluation metrics, and future enhancements that together shape an advanced, *role-based* AI solution for bridging medical education and patient-centric healthcare.

1.2 Motivation

Modern healthcare simultaneously faces escalating patient demands and a persistent shortage of medical professionals. On one hand, patients often require immediate attention and convenient scheduling mechanisms to mitigate anxiety, enhance treatment adherence, and reduce missed diagnoses. On the other hand, medical students and early-career clinicians must navigate the complexities of specialized knowledge, hands-on practice, and ever-evolving guidelines to stay current. Traditional educational methods—such as static textbooks and passive lectures—can be inadequate for preparing learners to handle complex, real-time scenarios.

In parallel, conventional patient-facing tools, such as basic symptom checkers or rule-based triage systems, rarely provide sufficient context or personalized guidance. They also lack direct integration with hospital resources for scheduling appointments. Consequently, patients may feel uncertain about the seriousness of their symptoms or spend valuable time consulting multiple, unverified online sources.

These challenges point toward a critical need for a comprehensive solution that offers:

- **On-demand access to credible information:** For patients, this means efficient triage and clarity regarding symptoms. For students, it entails interactive learning tools that simulate realistic, complex queries.
- **Reduced administrative barriers:** Automating tasks like appointment booking frees up health professionals' time and ensures that patients receive timely care.
- **Seamless retrieval of validated data:** Whether it is up-to-date medical guidelines or localized healthcare provider information, the system must unify relevant data and present it contextually.
- **Adaptability for different user roles:** Advanced question-answering for students differs substantially from straightforward symptom analysis and scheduling for patients. A single chatbot must fluidly handle these distinct modes.

By harnessing *large language models (LLMs)*, advanced embedding-based *retrieval systems*, and *agentic* AI capable of autonomously interacting with scheduling APIs, it becomes possible to unify medical education and patient-centric services in a single platform. This unified approach promises immediate benefits:

1. **Enhanced Learning:** Students can query the system at any complexity level, receiving deep, evidence-based explanations and case scenarios, thus bridging the gap between theoretical knowledge and real-world clinical encounters.
2. **Better Patient Outcomes:** Prompt appointment booking and symptom clarity can reduce treatment delays, lowering the risk of complications and alleviating patient stress.
3. **Administrative Efficiency:** Automated booking workflows diminish the overhead for healthcare facilities, reducing staff workload and the chance of scheduling errors or gaps.

Ultimately, the motivation for developing this dual-mode medical chatbot is to transcend the limitations of single-purpose educational platforms or simplistic symptom checkers. Instead, an **integrated AI-driven framework** is poised to deliver tailored user experiences, ensure seamless data flow, and cultivate both improved healthcare accessibility for patients and a transformative learning ecosystem for medical students.

1.3 Problem Statement and Challenges

Two problems define modern healthcare: patients require consistent counsel when faced with conflicting symptoms; medical students need basic access to current, accurate material to improve their core knowledge and prepare ready for clinical practice. Often missing the timeliness, complexity, and contextual clarity needed for complicated medical situations are existing solutions such simple symptom checklists or static web-based services. Furthermore, these stand-alone technologies seldom provide for timely scheduling or referral, thus leaving patients wondering about their next line of action on their medical route.

With regard to medical students especially, the difficulty is in Five Short Notes Names Especially when doing so in real time or when referencing other disciplines (pharmacy and pathology, for example), negotiating enormous numbers of research articles, medical textbooks, and clinical case reports can be challenging. Conventional e-learning systems barely combine theoretical knowledge with practical, scenario-based learning and lack efficient Q&A systems that fit to user inquiries.

Timeliness and Accuracy: Medical technology is evolving so rapidly that conventional resources could soon become extinct. Students run the danger of picking out outdated methods without fast, current sources.

About the patients, the challenges are equally important: Deciding whether their symptoms call for frequent visits or urgent care can be challenging for people, which could lead to worry and maybe postpone treatment.

Obstacles blocking healthcare access: Usually involving many phone calls or internet searches, making an appointment with a specialist or follow-up visit delays patients from receiving proper treatment—especially those unclear of the sort of expert to see.

Unverified or contradicting information: Patients may be pushed to self-diagnose via questionable online sources as basic symptom-checking websites may be inadequate or non-personalized.

Thus, the main challenge is designing a single, synthetic intelligence-powered system that satisfies both sets of needs:

1. A comprehensive teaching tool for medical students responding in-depth, contextually sensitively to specific questions derived from properly chosen clinical and scholarly sources. Allays any concerns, rapidly links patients to pertinent medical experts, and offers initial direction based on a patient-facing symptom analysis and triage mechanism.
2. Agentic appointment management helps to reduce administrative load and fill the information gap between pragmatic healthcare measures by enabling the system autonomously organize or postpone patient appointments.

These underlines the requirement of including Large Language Models (LLMs), embedding-based data retrieval, and user role-aware interactions. One such strategy is: Abbreviations 6 covers the academic demands needed for medical school as well as the pragmatic, immediate concerns patients have when presented with health issues. The next parts will address how this integrated chatbot idea may provide medical students and patients accurate, ethically acceptable, contextually relevant advice.

1.4 Research Questions

To answer the problem statements, following research questions needs to be answered. Below-mentioned research questions are obtained from the above-mentioned problem statements.

Research question 1: Does topic modelling aid in the extraction of topic process from the dataset of past interviews?

The best topic modelling techniques for extracting topics from the dataset must be analyzed to answer this research issue, which calls for a review of the literature.

Research question 2: How can we use various clustering techniques to extract information and cluster them from the German dataset?

Different types of clustering algorithms must be understood by reading through the literature to choose which algorithm is most suited for completing the work at hand. This question will provide to understand and chose the right algorithm.

Research question 3: How can language models be applied for topic modelling of German datasets?

Finding language models that perform well in German frequently requires a careful reading through many research articles that cover multilingual models or models created especially for German text.

1.5 Thesis Structure

This section will provide an overview and also explain the structure about the chapters which will be covered as part of the thesis:

Chapter 2 gives a broad overview of state-of-the-art methodologies which will help to understand the various techniques which are used in past to overcome the issue and the technique used for implementation of the thesis. Each of these methodologies is meticulously elucidated, offering a thorough understanding of their application and significance.

Chapter 3 explains about the thesis implementation pipeline, providing a comprehensive overview of each step included within the thesis framework. In this chapter, short explanations for every component of the thesis implementation process are also discussed.

Chapter 4 focuses on the practical implementation phase, were essential libraries are mentioned and gain a comprehensive understanding of the models employed to accomplish our task. This chapter provides a thorough breakdown of the libraries utilized throughout the code, specifying their roles in different sections of the code.

Chapter 5 is about the evaluation of the methodologies used to evaluate the cluster formation and topic extraction. It explains about the performance of the models and talks about the best performing model.

Chapter 6 is the final chapter and will provides the thesis conclusion and mention some of the challenges faced during the implementation of thesis along with the ideas on how the model can be improved and which other LLM models can be used as a part of the future work.

Chapter 2. State-of-the-Art

2.1 Introduction

The confluence of advanced artificial intelligence, massive data availability, and evolving clinical needs has transformed healthcare and medical education. In recent years, large language models (LLMs) and sophisticated embedding techniques have emerged as transformative tools that enable dynamic patient engagement and interactive educational support. These technologies now underpin conversational systems capable of handling complex, unstructured clinical data and generating contextually nuanced responses. This chapter provides an exhaustive review of the state-of-the-art approaches in the field, covering historical developments, technical innovations, decentralized data architectures, and evaluation methodologies. By integrating insights from two pivotal studies—Paper 1 and Paper 2—we lay the foundation for our dual-mode architecture designed to support both medical students and patients.

Our review is organized into several sections. Section 2.2 surveys the literature and traces the evolution of LLMs in healthcare; Section 2.3 delves into the methodologies, including retrieval-augmented generation, advanced embedding, dimensionality reduction, and clustering techniques; Section 2.4 addresses decentralized architectures and privacy-enhancing technologies; Section 2.5 outlines evaluation strategies and benchmarks; and Section 2.6 discusses the system architectures that support these functionalities. We conclude with Section 2.7, which explores the challenges and future research directions.

2.2 Literature Review

2.2.1 Evolving Role of LLMs in Healthcare and Medical Education

Historically, early healthcare information systems relied on rule-based expert systems and static databases with limited adaptability. With the advent of statistical language models and later deep learning approaches, the focus shifted toward data-driven methods. The introduction of transformer architectures in 2017 revolutionized natural language processing (NLP) by enabling models such as BERT and GPT to capture long-range dependencies and contextual nuances. This transformation paved the way for models that could generate highly fluent and context-sensitive responses.

Recent breakthroughs—exemplified by GPT-4 and open-source models like Meta's LLaMa series—have unlocked new possibilities for both clinical decision support and medical education. Paper 2 demonstrates that LLMs can now generate patient-tailored dialogue and educational content with remarkable fluency. However, both papers note that such systems require careful domain-specific adaptation and robust retrieval mechanisms to ensure factual accuracy.

2.2.2 Decentralized Architectures for Data Privacy

Data privacy is a critical concern in healthcare. Traditional centralized systems are vulnerable to data breaches and unauthorized access, posing significant regulatory and ethical challenges. Paper 1 proposes a decentralized architecture using Personal Data Stores (PDS) coupled with blockchain-based access controls. This approach empowers patients to maintain control over their own data while granting secure, auditable access to healthcare providers. Such architectures are designed to comply with stringent regulations (e.g., GDPR, HIPAA) and mitigate the risks inherent in centralized data storage.

Decentralization not only enhances data security but also builds trust among users. By filtering and de-identifying sensitive information at the edge before processing by LLMs, these architectures ensure that clinical interactions remain both private and effective.

2.2.3 Conversational AI for Patient Engagement and Education

Recent research has demonstrated the transformative potential of conversational AI systems in healthcare. Paper 2 outlines a multi-layered conversational framework where patient inputs are first processed by specialized modules—ranging from ad-hoc parsers for vital signs to intent recognition systems (e.g., Wit.AI)—before being passed to an LLM for free-form dialogue. This tiered approach is designed to protect sensitive clinical data while enabling rich, empathetic interactions.

For medical students, similar systems provide interactive tutoring that offers in-depth, evidence-based explanations and simulates realistic clinical scenarios. Tailoring responses based on user roles significantly enhances the system's utility, meeting the divergent needs of patients and students.

2.2.4 Methodological Innovations in Topic Modelling and Information Retrieval

Traditional topic modelling methods such as Latent Dirichlet Allocation (LDA), Non-negative Matrix Factorization (NMF), and Probabilistic Latent Semantic Analysis (PLSA) rely on bag-of-words representations, which often fail to capture the context-dependent meaning of medical texts. Recent innovations integrate deep learning-based embeddings and neural topic models (e.g., BERTopic) to extract more semantically rich topics.

Both Paper 1 and Paper 2 highlight the use of retrieval-augmented generation (RAG) as a means to ground the generative process in verifiable sources. By indexing trusted medical texts in a vector database and retrieving contextually relevant passages, these approaches improve both accuracy and reliability in clinical applications.

2.3 Methodologies

2.3.1 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) combines robust retrieval systems with powerful generative models to produce responses that are both fluent and factually accurate. In this approach, a vector database is populated with embeddings derived from a corpus of trusted medical literature (e.g., clinical guidelines and peer-reviewed articles). When a query is received, the retrieval system identifies the most relevant document fragments, which are then provided as context to the LLM.

This integration helps to mitigate hallucination by anchoring responses in verified data while enhancing the contextual richness of the generated responses. Both Paper 1 and Paper 2 provide evidence that RAG frameworks significantly improve the reliability of chatbot systems in healthcare.

2.3.2 Advanced Embedding Techniques

Embeddings convert text into dense numerical vectors that capture semantic meaning. Early methods such as Word2Vec and GloVe laid the groundwork, but transformer-based approaches like Sentence-BERT now dominate due to their ability to incorporate contextual cues. For multilingual and domain-specific applications—especially for German clinical texts—models such as GBERT, Cross English & German RoBERTa, and Jina embedding-v2-base-de are employed.

These embeddings are further refined using dimensionality reduction techniques such as UMAP, which projects high-dimensional vectors into lower-dimensional spaces while preserving intrinsic semantic relationships.

2.3.3 Dimensionality Reduction and Clustering

Dimensionality reduction is essential to manage high-dimensional embeddings. Techniques such as PCA, t-SNE, and UMAP reduce computational complexity and facilitate visualization. UMAP is particularly favored for its ability to preserve local and global data structures.

After reduction, clustering algorithms like HDBSCAN are applied to group semantically similar texts. HDBSCAN is preferred over methods like K-Means due to its robustness against noise and its ability to automatically determine the number of clusters. This multi-stage pipeline—embedding, dimensionality reduction, clustering, and topic extraction via LLMs—forms the backbone of modern topic modelling systems in digital health.

2.3.4 Topic Extraction via LLMs

Once clusters are formed, each group of semantically similar texts is processed to generate a concise topic label. An LLM (e.g., Llama 3.1) is used to synthesize information from each cluster and generate a topic name that captures the core theme. This process, driven by prompt engineering, enables the extraction of meaningful, context-aware topics, outperforming traditional bag-of-words methods.

2.3.5 Integrated System Architecture

Our proposed system architecture is highly modular and layered. It consists of:

- **Data Ingestion and Preprocessing:** Raw data from clinical texts, patient inputs, and educational materials are cleaned, segmented, and tokenized.
- **Embedding Generation and Storage:** Advanced embedding models convert the preprocessed text into dense vectors, which are stored in a vector database.

- **Retrieval Module:** Upon receiving a query, the system retrieves contextually relevant document fragments using similarity metrics.
- **Generative Module:** A fine-tuned LLM processes the query and retrieved context to generate a response.
- **Role-Aware Adaptation:** The system dynamically adjusts its response style based on whether the interaction is with a patient or a medical student.
- **Decentralized Data Management:** Sensitive patient data are managed via Personal Data Stores (PDS) and blockchain-based access controls.
- **Agentic Functionality:** Additional modules enable automated tasks such as appointment scheduling and real-time reminders.

Figure **??** illustrates the overall system architecture. This diagram, adapted from the architectures presented in Paper 1 and Paper 2, visually summarizes the flow from data ingestion to LLM-based response generation.

2.4 Challenges

2.4.1 Factual Accuracy and Mitigating Hallucination

LLMs are known to occasionally generate plausible yet factually incorrect responses (hallucinations). This risk is especially critical in healthcare, where inaccuracies can lead to severe consequences. While RAG helps ground responses in verified data, ensuring that the retrieval module consistently provides comprehensive and current context remains challenging. Future work must explore dynamic knowledge updating and domain-specific fine-tuning to further mitigate these issues.

2.4.2 Data Privacy, Security, and Ethical Considerations

Handling sensitive medical data demands stringent privacy and security measures. Decentralized architectures and data de-identification techniques, as outlined in Paper 1, are promising but require careful balancing to maintain data utility. Ethical concerns, including transparency, bias mitigation, and informed consent, are also paramount. Developing robust data governance and auditing frameworks is essential for building trust and ensuring regulatory compliance.

2.4.3 Computational and Resource Constraints

The computational demands of LLMs and high-quality embedding models are substantial. Processing large volumes of clinical data in real time requires significant GPU resources, memory, and processing power. Optimization strategies—such as model compression, quantization, and efficient batch processing—are necessary to reduce resource consumption and enable scalable deployments.

2.4.4 Complexity in Evaluation and Benchmarking

Evaluating AI systems in healthcare is inherently complex due to the need to balance automated quantitative metrics with qualitative human judgment. While metrics such as semantic similarity, precision@k, and clustering indices provide useful insights, they cannot fully capture aspects such as empathy, clarity, and clinical relevance. Standardizing evaluation protocols that integrate both automated and human-centric assessments remains an ongoing research challenge.

2.5 Evaluation Strategies

2.5.1 Quantitative Metrics

A comprehensive evaluation employs multiple quantitative metrics:

- **Semantic Similarity Scores:** Cosine similarity measures the closeness between generated responses and gold-standard references.
- **Retrieval Metrics:** Metrics like Precision@k and Mean Reciprocal Rank (MRR) assess the effectiveness of the retrieval module.
- **Clustering Validity Indices:** Silhouette Score, Davies-Bouldin Index, and Calinski-Harabasz Index evaluate the coherence and separation of clusters.
- **Pairwise Comparison:** Advanced methods utilize expert LLMs (e.g., GPT-4) to compare outputs across dimensions such as relevance, coherence, and fluency.

2.5.2 Human-Centric Evaluations

In addition to quantitative metrics, human evaluations are crucial:

- **User Studies and Surveys:** Collect feedback from clinicians, patients, and medical students regarding clarity, empathy, and clinical relevance.

- **Expert Panels:** Engage healthcare professionals to assess the accuracy and utility of the chatbot's outputs.
- **Task-Based Evaluations:** Design real-world clinical scenarios and educational simulations to evaluate system performance.
- **Longitudinal Studies:** Monitor user satisfaction and system performance over time to evaluate long-term impact.

2.5.3 Comparative Benchmarking

Comparative studies are essential to contextualize performance:

- **Traditional vs. Modern Approaches:** Benchmark the LLM-based approach against conventional methods (e.g., LDA, NMF).
- **Cross-Model Evaluations:** Compare outputs from different LLMs (e.g., GPT-3.5, Llama2-70B, Mistral-7B) under identical conditions.

2.6 System Architectures

2.6.1 Decentralized Data Architectures and Personal Data Stores (PDS)

Paper 1 proposes a decentralized data architecture that employs Personal Data Stores (PDS) in conjunction with blockchain-based smart contracts. This design empowers patients by ensuring that sensitive data remain under their control and only accessible by authorized healthcare providers. Such an approach minimizes the risks of centralized data breaches and complies with regulatory frameworks like GDPR and HIPAA. The decentralized model further supports traceability and transparency, essential for ethical data governance.

2.6.2 Modular and Layered Conversational Frameworks

Effective LLM-based chatbot systems are designed using a modular, layered approach:

- **Input Processing:** Raw data are preprocessed, cleaned, and segmented to remove noise.
- **Embedding and Retrieval:** Advanced embedding models transform text into dense vectors that are stored in a vector database for efficient retrieval.
- **Generative Response:** An LLM processes the query together with the retrieved context to generate a role-specific response.

- **Decentralized Management:** Sensitive data are managed via decentralized architectures to ensure privacy.
- **Agentic Functionality:** Automated modules facilitate tasks such as appointment scheduling.

2.6.3 Integration of RAG within a Secure, Decentralized Framework

The integration of Retrieval-Augmented Generation (RAG) within a decentralized framework is a key innovation. By retrieving context from verified sources and ensuring that only de-identified data are processed by the LLM, the system achieves both factual accuracy and data security. This integration is critical for applications in healthcare, where both accuracy and privacy are paramount.

2.7 Future Directions

2.7.1 Enhancing Model Reliability and Factual Accuracy

Future research must address the challenges of reducing hallucination and ensuring factual accuracy by:

- Implementing domain-specific fine-tuning.
- Integrating hybrid verification systems.
- Developing dynamic knowledge updating mechanisms.

2.7.2 Expanding Privacy, Security, and Ethical Frameworks

Advancements in privacy-enhancing technologies such as federated learning, differential privacy, and secure multi-party computation are essential. Establishing transparent data governance and bias mitigation strategies will further enhance ethical compliance.

2.7.3 Overcoming Computational and Scalability Constraints

To manage increasing computational demands, future systems should focus on:

- Resource-efficient architectures through model compression and quantization.
- Distributed computing strategies using scalable cloud and edge infrastructures.
- Optimized batch processing to reduce runtime.

2.7.4 Integrating Multimodal and Cross-Domain Capabilities

The next generation of healthcare chatbots will benefit from:

- Multimodal fusion of text, images, sensor data, and structured clinical records.
- Cross-domain adaptation to seamlessly transition between clinical decision support and educational tutoring.
- Cultural and linguistic adaptation to serve diverse populations effectively.

2.7.5 Standardizing Evaluation Protocols

There is a pressing need to develop standardized evaluation frameworks that integrate both quantitative metrics and human-centric assessments. Future work should focus on:

- Creating integrated benchmarking protocols.
- Conducting longitudinal studies to evaluate long-term impact.
- Fostering interdisciplinary collaboration to refine evaluation standards.

2.8 Summary

This chapter has provided an exhaustive review of the state-of-the-art in LLM-driven systems for healthcare and medical education. We traced the evolution of LLMs from early transformer models to modern systems that offer advanced fluency and contextual understanding. Detailed discussions on retrieval-augmented generation, advanced embedding techniques, dimensionality reduction, and robust clustering methodologies were presented, along with an in-depth analysis of decentralized architectures and privacy-enhancing techniques as outlined in Paper 1. Comprehensive evaluation strategies that blend automated metrics with human assessments were also discussed, highlighting the strengths and limitations of current systems.

The multi-layered system architecture—combining data ingestion, embedding generation, retrieval, and LLM-driven response generation—forms the foundation for our proposed dual-mode system. By addressing critical challenges such as factual accuracy, data privacy, computational efficiency, and evaluation complexity, our thesis aims to bridge the gap between clinical practice and medical education.

Future research directions include enhancing model reliability, integrating multimodal data sources, expanding decentralized frameworks, and standardizing evaluation protocols. These efforts will be essential for developing next-generation AI-driven medical chatbots that improve patient engagement and educational outcomes.

Chapter 3. Design and Methodology

This chapter is about the blueprint and the Methodologies used in the thesis involves a systematic progression of the steps to ensure a careful exploration of the topic modelling techniques. The process contains various steps and the implementation will start from the point of data collection and then analysing the data leading to the interpretation of results and their alignment with the initial research objective. The collected data is analysed then transformed into a format that is conductive for the effective processing which involves few steps like removal of unwanted data, combining the data, cleaning the data, and so on then, the data is used to feed into the model to obtain the results. Finally, the thesis concludes with a discussion of findings, implications, and potential avenues for future research, presenting a well-rounded culmination of the research endeavour. Figure 3.1 shows the implementation steps. These steps will be explained further below,

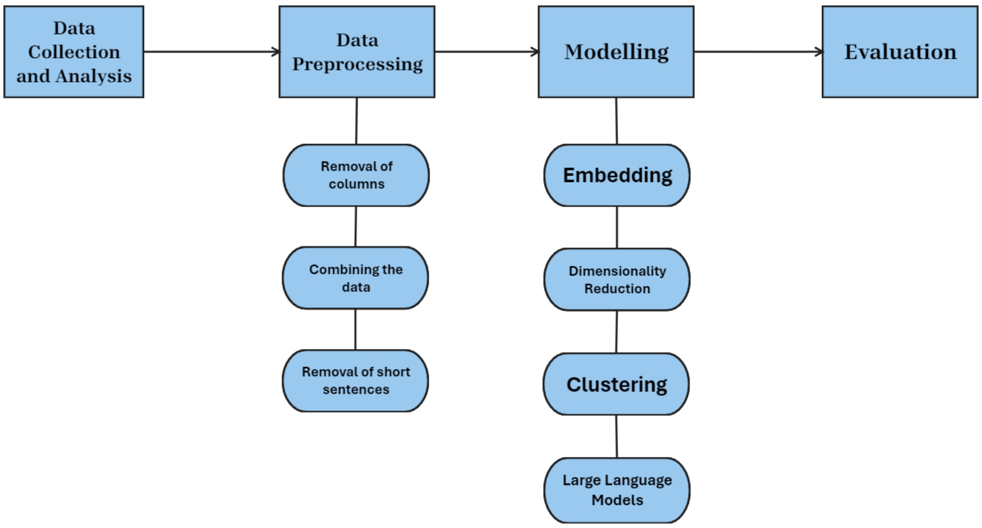

FIGURE 3.1: Thesis implementation phases

3.1 Data Collection and Analysis

The data for this thesis is taken from the Fern Universität in Hagen, Germany. From the early 1980s to the present, Fern University has conducted numerous interviews as part of its contemporary history research projects. These include biographical interviews were individuals interviewees narrate their own stories and also gives their point of view on the events that occurred during that time. Apart from these biographical interviews, autobiographies, written texts and various letters are also collected from many people with different culture background but have a connection to social events in Germany. The video interviews are also present which has wide information about various topics like work councils, trade unionists, migrants, concentration camp survivors, and so on. The interviews contain some information about the Germany history. There are about 350 Video interviews as part of the research which are convert into transcripts. The transcript of this dataset and contains three columns:

1. **Timecode:** As the video interview is convert into transcript, the timecode is used to identify the start and end of the interview. It is in the format of $hh : mm : ss$.
2. **Sprecher:** The name of the speaker in the interview.
3. **Transcript:** The interview transcript the conversation between the speakers.

3.2 Data Preprocessing

Once the data is analysed it needs be Pre-processed in order to use it as the data might have mistaken, redundancies, missing values, or inconsistencies due to all these reasons it needs to be cleaned so that, we will be able to use it for further analysis. For example, during the training of a machine learning model it is important to clean the data in order to avoid overfitting or underfitting or wrong predictions. Hence, before using the dataset to any tasks it is better do some data cleaning if needed. There is not much Preprocessing is used in the thesis, but it is important to clean the data before using it. Few steps are followed in this thesis is mentioned below,

1. **Removal of Unwanted columns**

   As the dataset has three columns of data, the timecode, the Sprecher and the transcript. In the Timecode column except the timestamp of the video we don't have any information which can be used for analysis. Hence, it is removed. Sprecher column is removed as it is not required for the analysis. Since, it consists of the who speak that particular sentence in the conversation and after analysis for the few transcripts it was not making much sense. Hence, it is removed. Transcript column is the key column which contains the main information of the conversation which is required for the analysis. Here, the rows of the interviewer is not removed as may contains the information which might be essential for the thesis.

2. **Combining the data**
   As the dataset was in the excel format and each conversation transcript was in the form of a new line in the column Transcript, it was necessary to combine the data into one single corpus. Hence, it is necessary to combine the data into one a paragraph so that contexts of the conversation can be maintained, and which will help in the future phases. On the other hand, the video conversation of each individual was available in the separate sheets in the excel file. Hence, it is necessary to merge the data into a single text document. As the looping through each page and combining them into a paragraph will take more time than the combination of the whole data. So, combining the data in one document will help to reduce the time taken for the analysis.
3. **Removing the short sentences**
   As the transcript column was combined into a paragraph and there we have lot of sentences which are smaller than 3 words. This Sentence will act as nosie as there are repeated words, Stopwords, punctuations, etc. have formed the sentence which is doesn't add value to the analysis as it is act like a noise in the data. Hence, it is necessary to remove all the sentences that are smaller than 3 words. As it doesn't make sense to have a sentence with less than 3 words. So, filtered the short sentences out of the dataset was done as it was like a noise in the data.

There are few other steps that can be done to clean the data like Stopwords, punctuations, lemmatization, etc., which I haven't considered as it is not required. Since, these preprocessing is only required when we are doing word embedding hence, I have not considered it in the thesis. As I'll will doing sentence embedding which will help to capture the meaning of the sentences in the paragraph and get better results.

3.3 Modelling

In this stage, the approach of achieve topic modelling will be eaxplained. As the Preprocessing of the dataset is done. The dataset is ready for topic modelling where it will undergoes stages like embedding, dimensional reduction, clustering and generating the topic name using the language model. The various stages and mentioned modelling techniques are explained by,

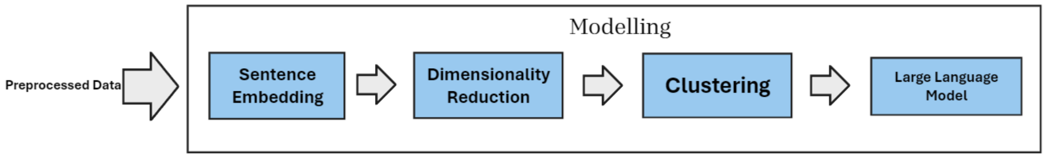

FIGURE 3.2: Modelling Workflow

3.3.1 Sentence Embedding

After the Preprocessing of the dataset is done. The dataset is ready for embedding and as Sentence Embedding approach will be used to get the numerical representation of the text data which will help the machine learning model to understand the text data. As the dataset is in German language choosing the good sentence embedding model is important to get better result as most of the embedding model are mostly trained on the English language Primary and then there are few models which are trained on multiple languages. So, choosing the good model is important to get better results. So, after careful examine some of the model for German language and few of the model is mentioned below,

1. GBERT Derived Embedding Models
2. Sentence-T5 derived Models
3. Paraphrase-multilingual
4. Cross English & German Roberta
5. Jina embedding-v2-base-de
6. German Sematic_STS_v2 and etc.,

After comparing the models embedding and evaluating them, Jina embedding-v2-base-de and Cross English & German Roberta model which work well in my scenario and by considering all the other factors like computational power, runtime, etc., I have chosen Jina embedding-v2-base-de model. As this model have a capabilities to handle both English and German text data along with it has been to work well in monolingual and cross-lingual applications without bias.

3.3.2 Dimensionality Reduction

The output of the embedding will be in very high dimensional space, and this cannot be used for further tasks. Therefore, in order to transform a high-dimensional vector space into a low-dimensional vector space, dimensionality reduction is necessary. There are lot of techniques that can be used for dimensionality reduction like PCA, t-SNE, UMAP, etc... For this thesis I have chosen UMAP reduction technique as it is a powerful and versatile dimensionality reduction technique which has some of the advantages when compared to PCA (Principal Component Analysis), t-SNE (t-distributed Stochastic Neighbour Embedding) and other techniques. Here is some of the reason why I have chosen UMAP reduction technique,

1. It preserves the local and global structure of the data as it maintains the relationships between the data points at multiple scales, ensuring small groups and the global structure of the data is well represented at the reduced dimensional space.
2. It is highly scalable and can be used for large datasets and its computational complexity is suitable for both small and large sentence embeddings.
3. One important aspect of sentence embedding is that it provides freedom in selecting the distance metric that is used to determine the degree of similarity between the data points.
4. It is faster and produces high quality embedding especially for large datasets when compared other technique.
5. It will help to increase better cluster separation.
6. This technique is quite versatile as it can be used for both dimensionality reduction and visualization.

As UMAP, have lot of benefits like it is very scalable, efficient, flexible and can be used for large datasets and it doesn't consume much computational power when compared to other techniques. This helps for producing meaningful results.

3.3.3 Clustering

After the Dimensionality reduction is carried out for the embedding, the data is now ready for the clustering. In this phase, there are two cluster technique based on the scenario which can be used i.e., K-Means and HDBSCAN clustering. By comparing the K-means and HDBSCAN clustering, I have chosen HDBSCAN as it is works well and has ability to cluster is well when compared to K-Means. Here are some of the reasons for choosing HDBSCAN over K-Means,

1. For K-Means Clustering, it is necessary to specify in advance how many clusters there are.
2. It struggle we it tries to deal with the handling outliers.
3. It tries to form uniform clusters of similar size as it assumes that all the clusters will be of same size.

After looking into these aspects and choosing the HDBSCAN clustering, the technique works effectively for identifying the clusters of varying densities and robust to the noise which is suitable for our dataset. As HDBSCAN approach doesn't have a control on the creation of the clusters but, it can be configured with the help of parameters like minimum cluster size, minimum samples and dew other parameters which will help to form the clusters by considering the core point which ensures that the clustering process efficiently irrespective of data. By applying the HDBSCAN and UMAP technique, we will be able to uncover distinct clusters of semantically similar sentence to group them together and by filtering the noise and outliers. The formed clusters will be stored in the json format to feed the formed clusters to the Llama 3.1 model.

3.3.4 Language Model

This is the final phase which will help to generate the topics for the given text. the clusters formed in the previous stage will be feed to Large Language model Llama 3.1. By providing the prompt along with the cluster, which will help the Language model to understand the sentences in the cluster and generate a generalized topic name for the particular clusters. This process involves leveraging the model's ability to process and synthesize information from multiple sentences, producing a coherent and meaningful topic label for each cluster. The use of Llama 3.1 model ensures that the generated topic names are contextually appropriate and semantically meaningful, enhancing the interpretability and utility of the cluster results. In this stage Llama 3.1 version model is used for the generation of topic names as it is the latest version open source model available and one of the powerful model which works efficiently for tasks like labelling, summarization, information retrieval, text generation and so on.

3.4 Evaluation

The quality of the generated topics can be evaluated by comparing them with the human generated labels and by feeding the clusters to other open source large language model and models like LDA, NMF and BERTopic will give us a clarity of the quality of the generated topics. For evaluation of clustering, I have used Silhouette coefficient, davies bouldin score and calinski harabasz score. which will give the clear understanding of the quality of the clusters formed and also based on this we can identify how good the embedding model is. The metric helps to choose the right embedding model for the given dataset as embedding of text data is very important. If the embedding model is not well suited for the dataset, it will not be able to generate good clusters. So, with the help of Silhouette coefficient, davies bouldin score and calinski harabasz score we can understanding of the quality of the clusters formed and how good the embedding has been done for the dataset. Along with this the topic generated by the Llama 3 model is compared with the human estimation so that, we can understand the quality of the topics generated and by feeding the clusters to other open source large language model which will give us a clarity of the quality of the generated topics. With the help of these metrics, I have compared the clusters formed and the generated topics of the model with human estimation.

3.5 Summary

This chapter was about the explanation of step-by-step process which was followed to accomplish this thesis. Starting from data collection where data was understood and analysed to know the structure of the data then, preprocessing of the data were unwanted columns and noise were eliminated and combined into a single paragraph was done. After that, the data was converted into numerical representation using the embedding technique followed by dimensionality reduction and clustering.

Chapter 4. Implementation

In this chapter, a comprehensive overview of the libraries utilized throughout and the functions applied for achieving topic modelling, specifying their roles in different sections of the code is given. Through deeper understanding of the inner workings od modelling and explaining the implementation of the algorithms, the thesis is concluded.

For implementation of the thesis, I have used platform like VS Code(Visual Studio Code) and Jupyter Notebook(Jupyter Notebook). VS code is integrated developement environment and Jupyter Notebook is a web based IDE, where both this IDE will support various languages and frameworks for the model implementation and developement.

4.1 Libraries

In this section, we will focus on certain aspects of libraries and their uses in the model implementation process.

Pandas:

Pandas is a library that is commonly used for data analysis and manipulation. It provides different functions and methods that are compatible with structured data such as data frames, CSV files, Excel sheets, etc. This library is very helpful in projects related to data science, machine learning and data analytics because it facilitates tasks such as: data cleaning; pre-processing; manipulation; visualization among others. Python Documentation [2023].

NumPy:

NumPy is a fundamental package for scientific computing in the python which provides a multidimensional array object, derived object (like masked array and matrices) and an assortment of routines for fast operations on arrays which includes mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic algebra, statistical operations, and much more. NumPy have been used in few places in this thesis, like converting the embedded data into an array so that it can be used for dimensional reduction and to find the total number of clusters formed Python Documentation [2023].

Pickle:

Pickle is one of the most used libraries in python for data serialization and deserialization as it converts objects into bytes streams for storage and restore them into original form. As pickle can handles objects of python includes list, dictionaries and so on, using different protocols optimized for efficiency. Common use cases include saving objects states, caching, and inter-process communication. The pickle library is used to store the embedded data formed as the data set is huge and running embedding every time takes lot of time Python Documentation [2023].

Json:

The python json module will offers a means of encoding and decoding data in the lightweight, text-based JSON (Javascript Object Notation) format which is frequently used for data transmission. The module allows you to serialize python objects, such as dictionaries, lists into json strings and deserialize json strings into python objects. Json is a human readable and language independent which can be used for web APIs and configuration files. Json format is used for storing the clustered data as this format is human readable and can be used for APIs as the clustered data will be feeded into the large language model Python Documentation [2023].

umap-learn:

umap-learn is an open-source python package will helps to implement the Uniform Manifold Approximation and Projection algorithm for dimensionality reduction. It does this by first learning a high-dimensional graph representation of the data, and thereafter, optimizes a low-dimensional layout which captures the local structure. The umap-learn package offers an easy-to-use interface, in addition to native interfaces to other well-known data science libraries like Pandas, Scikit-learn, and NumPy. It can also deal with supervised and unsupervised learning, which turns out to be versatile for many applications. Of these, a very notable one is dimensionality reduction of feature spaces, hence improving efficiency to downstream tasks like data visualization, classification or clustering Python Documentation [2023].

hdbscan:

hdbscan is a Python package that implements the HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) algorithm. This package is often used in tasks which involves complex data like biology, image processing, grouping the text etc., and this can be easily integrated with the other libraries like Pandas and scikit-learn. This algorithm is used for discovering the patterns in large dataset where number of clusters is unknown Python Documentation [2023].

transformers:

The transformers package is by Hugging Face is a powerful library that allows to access the wide range if pretrained transformers models for many NLP tasks. It supports both PyTorch and TensorFlow and it is easy to use where APIs allows the user to load the models. tokenize data and fine tune them. The package also includes a convenient pipeline API for applying models to various tasks with minimal code. With extensive documentation and a large community, transformers has become a go-to tool for state-of-the-art NLP Python Documentation [2023].

torch:

Troch is a deep learning library that is used for machine learning, and it is core library of PyTorch, a popular open source framework developed by Facebook's AI research lab. PyTorch is widely used for building and training deep learning models due to its speed, flexibility, and ease of use. This package provides a range of functionalities, including tensor operations (Similar to Numpy arrays with GPU acceleration), automatic differentiation, and building neural networks. It also supports GPU acceleration using CUDA, making it suitable for high-performance computing Python Documentation [2023].

Flask:

Flask, being an adaptable and simple web framework helps to create web apps with python. the server-side processing is handled via this tool while it also influences the web request management thanks to its essential utility for forming a web server and directing the requests between the front-end and back-end section of the application Python Documentation [2023].

HTML/CSS:

The application's front-end was developed with HTML for structure and CSS for styling. This combination ensures the user interface is functional and attractive. Hence, it is able to carry out its task without much ado and in a commendable way as well. User input is collected using HTML forms while CSS serves the purpose of improving on webpage aesthetics.

4.2 Code Implementation

This section will explain how the code is implemented in the thesis to achieve topic modelling. This section consist of subsection as follows:

1. Preprocessing
2. Embedding
3. Dimensionality reduction
4. Clustering
5. Large Language Model

4.2.1 Preprocessing

Once the initial data analysis has been done, the data will undergoes preprocessing steps as mentioned below,

LISTING 4.1: Initialization of the list of noise

```
# List of common filler words
filler_words = {"ja", "hm", "mhm", "ach", "gut", "und", "eben", "ne", "ok", "aha", "ach so", "
    nicht", "ja", "nein", "wieder", "schon", "naja", "wieso", "wieso nicht", "wieso nicht"}
```

As there the data is the transcript of the video Interviews, it consists of lot noise in the data. There are lot of shot sentence, noise, repeated small sentences etc., hence, it is necessary to remove them. The below code has list of noise which can be removed from the data. and this is the initialization of the list.

LISTING 4.2: Filtering of noise from the data

```
  def has_repeated_filler_patterns(sentence, filler_words):
    # Check if any filler word appears repeatedly either consecutively or separated by commas
    or spaces in the sentence.
      pattern = r'\b(' + '|'.join(re.escape(word) for word in filler_words) + r')\b(?:[\s,]+)
    +\1\b'
```

```
        if re.search(pattern, sentence):
            return True
    return False

```

The function `has_repeated_filler_patterns` is useful when talking about text because it helps by filtering out certain sentences from it while breaking the text down into different sentences through regular expressions that look for punctuation marks showing end of sentence. This function checks a variety of factors including minimum word count as well as unique word count. If all these conditions are met, then that particular sentence will go into a filtered sentence list. In addition this could also call has repeated filler patterns to eliminate noise.

LISTING 4.3: Filter sentences from the text

```
def filter_short_and_filler_sentences(text, filler_words, min_words, min_unique_words,
    min_characters):
    # Filter sentences from the text based on length, unique words, minimum characters, and
    absence of repeated filler patterns.
    sentences = text.split('. ')
    filtered_sentences = []

    for sentence in sentences:
        words = sentence.split()

        if len(words) >= min_words and len(set(words)) >= min_unique_words and len(sentence)
    >= min_characters:
            if not has_repeated_filler_patterns(sentence, filler_words):
                filtered_sentences.append(sentence)

    return '. '.join(filtered_sentences)
```

The `filter_short_and_filler_sentences` function will process the text to filter out sentences and it splits the text into individual sentences using the regular expression to check the various punctuation marks that signify the end of a sentence. This function check several conditions like minimum number of words number of unique words and exceed the length in characters once these conditions are satisfied, the sentence will be appended to the list of filtered sentences. Additionally, it will call `has_repeated_filler_patterns` function to identify and remove noise.

LISTING 4.4: Removing columns and combining the sentences to paragraphs

```

    excel_data = pd.read_excel(file_path, sheet_name=None)

    # Combining the sentences to paragraphs and applying the filtering
    combined_paragraphs = {}

    for sheet_name, df in excel_data.items():
        # Drop the 'Timecode' and 'Sprecher' columns
        df = df.drop(columns=['Timecode', 'Sprecher'])

        # Convert the 'Transkript' column to a single string
        df['Transkript'] = df['Transkript'].astype(str)
        paragraph = ' '.join(df['Transkript'].tolist())

        # Filter short sentences and apply filler pattern filtering
        filtered_paragraph = filter_short_and_filler_sentences(paragraph, filler_words,
    min_words, min_unique_words, min_characters)

        # Save the filtered paragraph by sheet name in the dictionary
        combined_paragraphs[sheet_name] = filtered_paragraph

    # The `combined_paragraphs` dictionary now contains the filtered paragraphs for each sheet
     in memory.
    # You can now use `combined_paragraphs` as needed in your code.

    print("Filtered paragraphs have been processed and stored in memory.")
```

The above code will help to read the excel file, remove the columns and combine the sentences to paragraphs, where it reads the data from each sheet in the excel file and removes the columns Timecode and Sprecher. It also combines the data into one single paragraph by adding the sheet name to the start of the paragraph. This code will call the `filter_short_and_filler_sentences` function to filter the sentences and adds the sheet name at the begin of the paragraph and save the combined data to a text file and pickle file for the future use.

4.2.2 Embedding

This the code will load the pre-trained Jina embeddings model and tokenizer the pre-processed text data. The model will be used convert the text data into embeddings.

LISTING 4.5: Load Jina AI embedding model and tokenizer

```

# Load the pre-trained Jina embeddings model and tokenizer
model_name = 'jinaai/jina-embeddings-v2-base-de'
model = AutoModel.from_pretrained(model_name, trust_remote_code=True)
```

```
tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)

# Assigning the combined_paragraphs to the text variable
text = combined_paragraphs
# Split the text into sentences based on '. ' delimiter
sentences = text.split('. ')
# Define the batch size for processing
batch_size = 32 # can be adjusted based on the computational power
embeddings = []

# Process the sentences in batches
for i in range(0, len(sentences), batch_size):
    batch_sentences = sentences[i:i+batch_size]

    # Tokenize the batch of sentences
    inputs = tokenizer(batch_sentences, return_tensors='pt', padding=True, truncation=True,
    max_length=512)
    # Compute embeddings without tracking gradients
    with torch.no_grad():
        outputs = model(**inputs).last_hidden_state

        # Compute the mean embedding for each sentence in the batch
        batch_embeddings = outputs.mean(dim=1).cpu().numpy()
        # Append the batch embeddings to the list
        embeddings.extend(batch_embeddings)

# Convert the list of embeddings to a NumPy array
embeddings = np.array(embeddings)
```

Load Model and Tokenizer: It loads a pre-trained Jina embeddings model and its corresponding tokenizer. The `trust_remote_code=True` argument ensures that remote code is trusted when loading the model.

Split Text: The data is split into individual sentences using '. ' as the delimiter. As the text dataset has been combined into a paragraph, it is necessary to split the text into individual sentences. As the model can handle sequences length of the embeddings is 8192 and I have reduced to sequence length to 512 as in the future phase as the maximum sequence length need more computational power and memory.

Batch Processing: Sentences are processed in batches of size 32 to manage memory and computational efficiency. This will also reduces the runtime of the embedding process. The batch size can be alter based on the computational power.

Tokenization and Embedding Computation: Each batch of sentences is tokenized using the model's tokenizer. The embeddings are computed without tracking gradients for efficiency. For every sentence, a fixed size embedding is obtained by calculating the mean of the last concealed state across tokens.

Store Embeddings: The embeddings for each batch are collected and converted into a NumPy array for further use.

4.2.3 Dimensionality Reduction

As the embedding data will be having high dimensional space and it cannot be used for the visualization and clustering directly hence, the UMAP reduction technique is used which will helps to convert the high dimensional space into a low dimensional space.

LISTING 4.6: Dimensionality Reduction

```
import umap.umap_ as umap
# Create a UMAP model instance with specified hyperparameters
umap_model = umap.UMAP(n_neighbors=20, n_components=10, min_dist=0.1, metric='cosine',
    low_memory=True)

# Fit the UMAP model to the filtered embeddings and transform the data
umap_embeddings = umap_model.fit_transform(filtered_embeddings)
```

Initialize UMAP Model: The UMAP model is initialized with the following hyperparameters:

- `n_neighbors`: Number of neighboring points used in local approximations.
- `n_components`: The number of dimensions for the reduced embedding.
- `min_dist`: Minimum distance between points in the low-dimensional space.
- `metric`: Distance metric used to calculate distances between points.
- `low_memory`: Optimizes memory usage by storing fewer intermediate results.

These hyperparameters will help to balance capturing both local and global structure in the data while reducing the dimension of the data. Setting the `n_neighbors` will allows the model to consider neighborhood size preserving both local relationships and broder patterns. The `n_components` which will helps to retains the important information for downstream tasks with any data loss. The `min_dist` will relatively control the distance between groups and tightens the clusters making it easier for grouping the distinct groups. The use of `low_memory=True` will help to reduce the memory usage of the model whenever it is dealing with larger dataset.

Fit and Transform Data: The UMAP model is trained on the `filtered_embeddings` data to reduce its dimensionality from high-dimensional space to a 10-dimensional space. The `fit_transform()` method performs both fitting the model and transforming the data into the new lower-dimensional space. The transformed data is stored in `umap_embeddings`.

4.2.4 Clustering

HDBSCAN is used to cluster the embeddings obtained from UMAP. The clustering results are organized into a JSON file to facilitate easy access and analysis of the clusters and their associated sentences.

LISTING 4.7: HDBSCAN Clustering

```

# Create a HDBSCAN Clustering instance with specified hyperparameters
clusterer = hdbscan.HDBSCAN(min_cluster_size=40, cluster_selection_epsilon=0.4, metric='euclidean', cluster_selection_method='eom')
clusters = clusterer.fit_predict(umap_embeddings)

# Print the number of clusters
num_clusters = len(set(clusters)) - (1 if -1 in clusters else 0)
print(f"Number of clusters: {num_clusters}")
```

Initialize and Apply HDBSCAN: The HDBSCAN clusterer is configured with:

- `min_cluster_size`: Minimum size of clusters.
- `cluster_selection_epsilon`: The parameter to regulate the clusters granularity.
- `metric=`: The distance metric used for clustering.
- `cluster_selection_method`: Method for selecting clusters (excess of mass).

The `fit_predict()` method applies HDBSCAN to the `umap_embeddings` and returns cluster labels.

Count and Print Number of Clusters: The number of clusters is calculated by counting unique cluster labels and adjusting for the label `-1`, which represents noise. The result is printed.

LISTING 4.8: Count and Print Number of Clusters

```
# Create a list to hold clusters with their sentences
clusters_list = []

for cluster in np.unique(clusters):
    # Convert the cluster ID to a regular Python int
    cluster_id = int(cluster)
    # Collect sentences for each cluster
    sentences_in_cluster = [sentence for i, sentence in enumerate(sentences) if clusters[i] ==
     cluster]

    # Create a dictionary for the cluster
    cluster_dict = {
        "cluster_id": cluster_id,
        "sentences": sentences_in_cluster
        }
    # Add the cluster dictionary to the list
    clusters_list.append(cluster_dict)

# Save the clusters to a JSON file
output_json_file = 'clusters.json'
with open(output_json_file, 'w', encoding='utf-8') as f:
    json.dump(clusters_list, f, ensure_ascii=False, indent=4)

print(f"Clusters have been saved to '{output_json_file}'")
```

Organize and Store Clusters: The code creates a list of dictionaries where each dictionary represents a cluster. Each dictionary contains:

- `cluster_id`: The ID of the cluster.
- `sentences`: A list of sentences belonging to that cluster.

Save Clusters to JSON: The list of clusters is saved to a JSON file named `clusters.json`. The `json.dump()` function serializes the list and writes it to the file. A confirmation message is printed indicating successful saving.

4.2.5 Large Language Model

The cluster data which is stored in the JSON file is used to generate the topic names for each cluster. This process involves leveraging the model's ability to process and synthesize information from multiple sentences, producing a coherent and meaningful topic label for each cluster.

Loading the Model and Tokenizer: Accessing the pre-trained language model and tokenizer through Hugging Face model hub with a script that has a name Meta-Llama-version, it is based on an access token (this model is being used because of my compute resources constraints although other models may be possible too).

LISTING 4.9: Loading Llama 3 model

```python
import json
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer

# Debugging function to help track where issues might be occurring
def debug_message(message):
    print(f"[DEBUG] {message}")
# Load the model and tokenizer once at the beginning
try:
    debug_message("Loading model and tokenizer...")
    model = AutoModelForCausalLM.from_pretrained("meta-llama/Meta-Llama-3-8B", token="HuggingFace access token")
    tokenizer = AutoTokenizer.from_pretrained("meta-llama/Meta-Llama-3-8B", token="HuggingFace access token")
    debug_message("Model and tokenizer loaded successfully.")
except Exception as e:
    debug_message(f"Error loading model or tokenizer: {str(e)}")
    raise
```

Generating Topics: A function named `generate_topic` is used to generate a topic name for a set of sentences within a cluster. The sentences are combined into a corpus and fed to the language model using a prompt, which returns a concise and specific topic of up to 5 words. The topic generation process is wrapped in error handling to catch any issues that may arise.

LISTING 4.10: Generate Topic

```python
def generate_topic(sentences):
    try:
        # Combine sentences into a single string
        combined_sentences = " ".join(sentences)
        # Create a prompt for the model
        prompt = f"Analysiere die folgenden Saetze und generiere einen praezisen, aussagekraeftigen Themennamen von maximal 5 Worten. Der Themenname sollte den Kerninhalt erfassen und spezifisch sein. Saetze: {combined_sentences} Thema:"
        # Tokenize input and generate the topic
        inputs = tokenizer(prompt, return_tensors="pt")
        outputs = model.generate(**inputs, max_new_tokens=10, num_return_sequences=1)
        # Decode the generated topic
        return tokenizer.decode(outputs[0], skip_special_tokens=True).strip()

    except Exception as e:
```

```
        debug_message(f"Error during topic generation: {str(e)}")
    return "Topic generation failed"
```

Loading Clusters: The clusters of sentences are loaded from a JSON file (`clusters.json`), which contains a list of dictionaries. Each dictionary represents a cluster and includes a `cluster_id` and the corresponding `sentences`.

LISTING 4.11: Load Clusters from JSON

```
# Load the clusters from the JSON file
input_json_file = 'clusters.json'
try:
    debug_message(f"Loading clusters from {input_json_file}...")
    with open(input_json_file, 'r', encoding='utf-8') as f:
        clusters_list = json.load(f)
    debug_message("Clusters loaded successfully.")
except Exception as e:
    debug_message(f"Error loading clusters: {str(e)}")
    raise

# List to hold the results with generated topics
clusters_with_topics = []
```

Processing Clusters Sequentially: The function `process_clusters_sequentially` iterates through each cluster, skipping any noise clusters (identified by `cluster_id` = -1). For each valid cluster, the function generates a topic, appends the topic to the cluster data, and adds the updated cluster to a list (`clusters_with_topics`). After processing each cluster, the script clears the CUDA memory (if using a GPU) to optimize memory usage.

LISTING 4.12: Process Clusters Sequentially

```
def process_clusters_sequentially():
    for cluster in clusters_list:
        try:
            # Extract cluster information
            cluster_id = cluster["cluster_id"]
            # Skip the cluster with ID -1 (noise)
            if cluster_id == -1:
                debug_message(f"Skipping noise cluster {cluster_id}")
                continue

            sentences = cluster["sentences"]
            # Debug: Show the cluster info
            debug_message(f"Processing cluster {cluster_id} with {len(sentences)} sentences.")

            # Generate a topic for the cluster
```

```
            topic = generate_topic(sentences)
            # Add the generated topic to the cluster data
            cluster["topic"] = topic
            # Append the updated cluster to the results list
            clusters_with_topics.append(cluster)
            # Clear memory after processing each cluster
            torch.cuda.empty_cache()  # If you're using a GPU, clear the cache
            debug_message(f"Processed cluster {cluster_id}, Topic: {topic}")
        except Exception as e:
            debug_message(f"An error occurred while processing cluster {cluster_id}: {str(e)}")

# Process all clusters sequentially
try:
    process_clusters_sequentially()
except Exception as e:
    debug_message(f"Error during cluster processing: {str(e)}")
    raise
```

Saving Results: Once all clusters have been processed, the updated clusters with generated topics are saved to a new JSON file (`clusters_with_topics.json`) for further use.

LISTING 4.13: Save Clusters with Topics to JSON

```
# Save the clusters with topics to a new JSON file
output_json_file = 'clusters_with_topics.json'
try:
    debug_message(f"Saving clusters with topics to {output_json_file}...")
    with open(output_json_file, 'w', encoding='utf-8') as f:
        json.dump(clusters_with_topics, f, ensure_ascii=False, indent=4)
    debug_message("Clusters with topics saved successfully.")
except Exception as e:
    debug_message(f"Error saving clusters: {str(e)}")
```

Throughout this code, a debugging function (`debug_message`) is used to print out messages for tracking the progress and identifying potential issues.

4.3 Web Application for Smaller Text Corpus

In this section, the development of the a web application designed to process the smaller text data with the above implementation. The primary goal of this application is to provide a user-friendly interface for analysis of the smaller text corpus. This application will also the same workflow but, the data will be directly feed into the embedding model which is followed by clustering then to the LLM model (Llama).

4.3.1 Methodlogy and Technologies Used

To implement this web application the following Technologies are employed, I have used the function which i have used in the previous section along with the flask and HTML/CSS. Here the flask is used to run the web application where is manage the POST request and Get request. The HTML/CSS is used to design the user interface (UI). The data is processed in the same way as in the previous section. Over here instead of loading the files the user can directly paste the text data which we want to do topic modelling. Once the data is loaded then the user can change the minimum size of the cluster and click on process button to process the data. The data will be undergo the same process which is mwntion in thw previous section like embedding, clustering and then to language model to generating the topic.

Below are few snapshots of the web application,

This is the UI of the web application where the user can change the minimum size of the cluster and process the data.

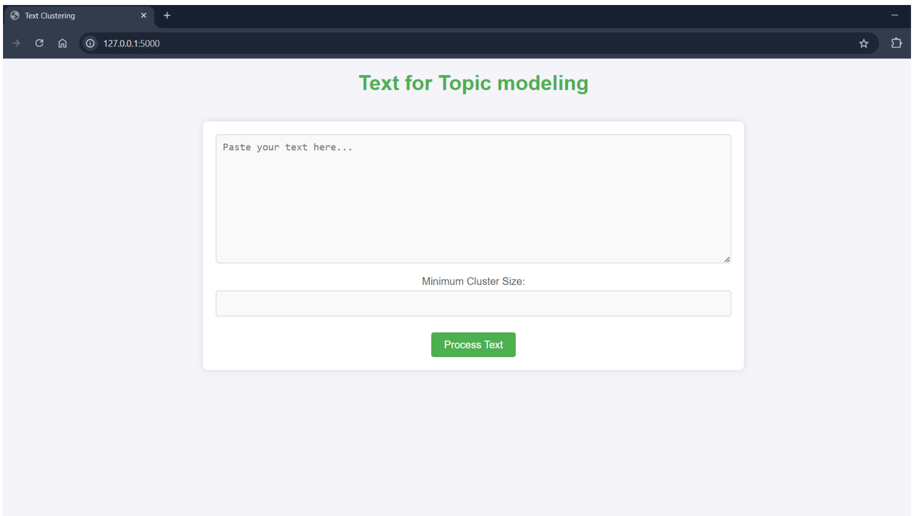

FIGURE 4.1: UI for the web application

The output of the web application looks as attached below,

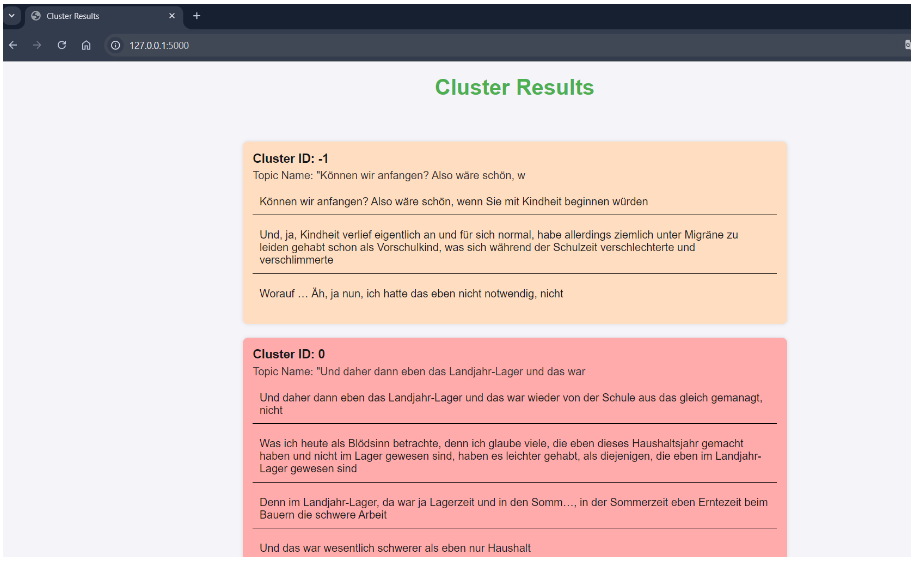

FIGURE 4.2: UI for the web application

This web application is used to process the smaller text data and generate the topic in the same way as the previous section. This will help the user to process the data faster and can be used for both German and English language.

4.4 Summary

This chapter gives the detailed explaination about the various libraries used for the implementation of the topic modelling and along with the web application for smaller text corpus. The process of achieve the topic modelling on the historical biographical interview data using the embedding, clustering and large language model is explained in the previous section. The web application is used to process the smaller text data and generate the topic in the same way as the previous section. With the help of this implementation approach for achieving topic modelling describe how LLMs can be used for topic modelling when the dealing with the larger text documents. With this chapter implementation of the thesis is completed and next chapter will be about checking the performance of the models implemented.

Chapter 5. Evaluation

In this chapter, the evaluation of the model is discussed with various techniques used to analyze how good the clustering has been done and generation of the topics with other LLM models along with the human evalaution.

5.1 Evaluation Design

There are various techniques that can be used to evaluating the model performance and one of the most effective way is through human evaluation. Along with this I have tried to evaluate the model performance by evaluating the clusters formed by the model amd comparing the topic generated by the LLM model with other open-source LLM models. For the human evaluation following of the questions are asked to

1. **How well do you think notebooks effectively explain concepts or content to you?**
   This question was asked to check if the information provided to explain the Python codes for implemented models in the VS code Jupyter notebooks were easy to understand.
2. **How would you rate the cluster formation of the model?**
   The purpose of this question is to determine whether the sentences inside the clusters that the model created were similar to one another or not.
3. **How would you rate the topic generation of the model?**
   This question was asked in order to know if the topics generated by the model were good or not for the cluster sentences so that, the model was able to understand the content of the cluster and generate the proper topic name for the cluster.
4. **How would you rate the LLM model approach when compared to the traditional approach?**
   This question will help us to understand if the LLM model approach is good or not compared to the traditional approach.

5.2 Cluster Evaluation

Cluster evaluation is used to evaluate the quality of the clustering done by the model. There are various techniques that can be used to evaluate and I have considered the below technique. This cluster evaluation is done using the Silhouette Score, Davies-Bouldin Index and Calinski-Harabasz Index which will suitable for evaluating the clusters formed by the clustering algorithm. This will help to understand the quality of the clusters and embedding models. As the data is in the German language, finding the embedding model is very important and the selected embedding model need to cover the entire data so that, the clustering can be done properly. As if chosing a inefficient embedding model, will result in inproper cluster formation and which leds to the poor results. So, by conducting theis evaluation, we can understand the quality of the embedding model along with the cluster formation. The below metrics will give an idea how the evalaution score is calculated is explained below,

1. **Silhouette Score:** Measure of how closely an object is to one's cluster relative to other clusters. The score has a range from -1 to 1, with higher score indicating better clusters. These also give insights into the cluster cohesion and separation.
2. **Davies-Bouldin Index:** ThThis indexes the average similarity between each cluster and its most similar one. Lower Davies-Bouldin values determine a better partition. The compactness of the separation of the clusters will be measured.
3. **Calinski-Harabasz Index:** It measures the ratio of between-cluster scatter to within-cluster scatter. Bigger values are indicative of a better-defined cluster. Index works best with clusters that are convex in shape GeeksforGeeks [2024].

The metrics for each embedding model are shown below,

Embedding Model	Silhouette Score	Davies-Bouldin Index	Calinski-Harabasz Index
Jina AI de	(0.3 - 0.6)	0.95	3459
Cross RoBERTa en-de	(0.1 - 0.4)	1.1	456
BERTopic	(-0.3 - 0.1)	1.5	236

TABLE 5.1: Cluster Evaluation Metrics for HDBSCAN

Based on the above metrics, the Jina AI de model performed well with the hdbscan clustering for the german interview transcript dataset. This metric made me to choose the Jina AI de model for embedding of the interview transcripts.

5.3 Topic Generation with other LLMs

Evaluation includes my using the same LLM model for topic generation by feeding the clusters to the LLM model and prompting the model for generating topics based on the cluster. Since I am using the Llama 3 - 8B model for topic generation, I want to check with other LLM models. The topics generated for each cluster were manually evaluated for whether they are similar or not. Random proverb clusters chosen to prompt models like the GPT 3.5 model by Open AI ChatGPT, Google Gemini model, and a few other LLM models, have been presented here.

The evaluation was performed manually on the generated topics of each cluster to verify if it is similarly trained as the state-of-the-art models. which will give an overview of how good the clustering has been done and what topics are generated.

The output of the complete text based on the sentences belong to the particular cluster will looks as shown in the below attached imaged,

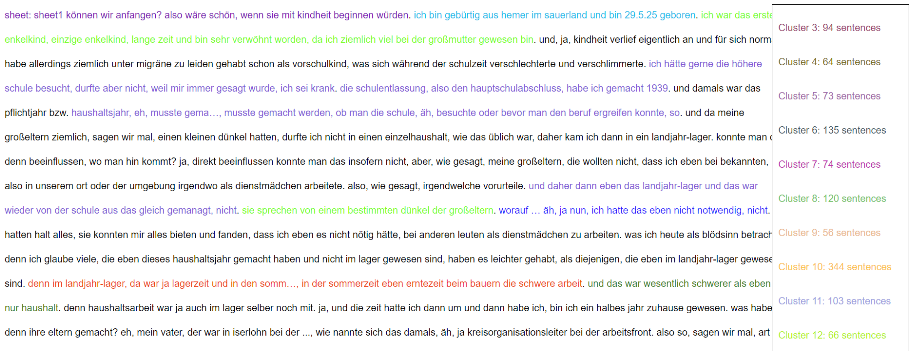

FIGURE 5.1: Clusters sentence distribution w.r.t Original text corpus

This will helps us to get to know how the sentences for each cluster are present and how the topics are distributed accross the documents.

Along with these metrics, I have also compared the clusters formed and the generated topics of the model with human estimation. By taking feedback from humans, on the questions framed in the evalaution design section to understand the model performance.

Human Feedback

On Q1: As jupyter notebooks consist of python code with comment the people with the technical knowledge were able to understand the code without much explanation but, the people with non-technical knowledge were understanding the notebook code based on the comments added. So, the people with non-technical knowledge suggested to having a user interface would be better so that everyone can understand.

On Q2: As the data is huge and there are lot of sentences under each cluster, a random cluster was selected to people to get to know the whether the sentences in the cluster is similar or not and got positive feedback that, the sentence in the cluster is similar to each other and there are lot of sentences in the cluster which has same meaning but expressed in different ways.

On Q3: As a this question is a follow up of Q2, the topic generated by the model for the specific cluster is almost matching and the topic name generated is representing as the title of the cluster which will help to understand related to what domain the cluster belongs to.

On Q4: This question was asked to know if the LLM approach is good or not compared to the traditional approach where I tried to show output of the conventional approach like LDA, NMF etc.., and ask to compare with this thesis approach based on this there was some feedback like the traditional approach has the word representation and this model has a sentence representation. The traditional approach is has grouped the words which are like each other together and the LLM approach has a sentence representation. There was mixed feedback on this question as few people felt traditional approach is better and few people felt the LLM approach is better.

Based on the above feedback, for Question 1, 2 & 3, decide to build a web application which can handle the same text data.

5.4 Discussion

After reciving the feedback from the humans on the evalaution of the models, it was clear that the Jina AI de model was performing well with the HDBSCAN clustering algorithm for the interview transcripts dataset. As the feedback was positive, when it comes to the clustering of the sentences and generation topic name of the cluster made it clear that the technique of achieving the topic modelling using embedding, cluster and LLM models was good. Eventhrough, the feedback on the comparsiom of the traditional approach and LLM approach was mixed. Based on these feedbacks the idea of building an web application came up and the web aplication was created which can handle the same text data (mentioned in the previous section).

The cluster evaluation plays an important role as it help to understand the quality of the clusters and embedding models. The meteric scores of silhouette score, davies-bouldin score and calinski-harabasz score gave an idea of how good the clustering is done. The evaluation of the topic generation with other LLMs is a manual evaluation which also helped in understanding how good the Llama model has generated the topics and along with the human feedback on the cluster formation and the topic generation help to understand the performance of the model. With the completion of this chapter, the evaluation of the models has been completed.The next chapter will provide a summary of the thesis and also explain the challenges encountered during the research.

Chapter 6. Conclusion and Future Work

This final chapter of the thesis serves to conclude the research, by presenting a comprehensive reflection on the various aspects explored and identifies potential areas for the future research. This chapter will give an quick overview of the thesis by providing the key findings, conclusion and future work. As the main objective of this thesis is to identify an unique approach which can handle the complex relationships among sentences in a text and to achieve the topic modelingusing the latest techniques.

6.1 Conclusion

This thesis proposes a new method of topic modeling, unlike the earlier methods in the forms of LDA, NMF, and LSA. The conventional approaches have great capabilities in topic extraction, but they still strongly depend on word co-occurrence patterns. Besides, they cannot extract the deeper contextual understanding of text data. Meanwhile, BERTopic is a language model-based topic modeling that covers an alternative by making use of embeddings, dimensionality reduction, clustering, and the representation of topics using c-TF-IDF and count vectorizer techniques. Despite such merits, at the same time, BERTopic fails in completely capturing such complex semantic relationships among words in a text. Moreover, performance can be poor for smaller datasets, and results may slightly vary on runs with the same data.

Considering these limitations, a new approach to topic modeling in an attempt to solve some of these challenges. In particular, this thesis focuses on improving topic modeling by providing better semantic understanding of relationships among sentences and extracting more meaningful topics from text. While the novelty of this approach embeds, reduces the dimensionality, and clusters, it makes a further step by feeding the clustered sentences into large language models. Large language models are better equipped to grasp complex semantic relationships and therefore allow for more accurate and insightful topic generation. This, in turn, helps the researchers and analysts in identifying the underlying themes and patterns in text data more appropriately.

As the dataset is in German language, choosing the right embeddings model is a very important part of the process to get the best results. There are many embeddings models which can be used to tackle the problem but, I chose Jina embedding-v2-base-de model because of its capabilities to handle both English and German languages data and gave better clustering results than other models and it uses less computational power and time to process the data. For clustering the embedded data, I have used HDBSCAN algorithm over K-Means as it struggles to handle noise in the data and also need to specify the number of clusters in advance. As the language models are powerful in extracting the semantic relationships and most of the models are multilingual and handle German language feeding the clustered sentences to LLMs model will help to extract more meaningful topics for the text documents.

6.2 Answers to Research Questions

By answering the research questions that were formed earlier in this thesis, we shall be able to appreciate its possibilities and whether its objectives have been met.

1. **Q1: How does topic modelling aid in the extraction of topic process from the dataset of past interviews?**
   The topic modelling approach can be used to extract the topics from the historical interview dataset as this question was made to understand the topic which were already exsited in the past to understand how the topic modeling has been achieved in the past and how it can be improved. This question help to understand the potential of the approaches like LDA, NMF, LSA, BERTopic, prompt-topic-model etc., and how they can be improved in the past to achieve better results. With the help of this question, I was able to understand the workflow which has been used and about its advantages and disadvantages and how has been used in the past which various approaches to achieve better results. This question helped me to understand the potential of the approaches and how it can be used for the interview dataset. This question also helped me to come up the idea of approaching the problem of handling the huge text data by using embedding, clustering and use of large language models.

2. **Q2: How can we use various clustering techniques to extract information and cluster them from the German dataset?**
   This question will help me to understand the various clustering techniques and how they can be used to in this thesis to achieve topic modelling. There are

many clustering techniques like K-Means, DBSCAN, Hierarchical Clustering, HDBSCAN etc., which will help to cluster the data. This research support for choosing the clustering technique suitable for the dataset. As these techniques are effective in grouping the data which are similar to each other. After understnading the pros and cons of various clustering techniques, which helped to achieve better cluster formation for the text data in this thesis. As the number of topics present in the data is unknow and identifying the topics from the data is difficult this question helped to understand how clustering techniques can be used to achieve the topic modelling. After understanding the various aspects of clustering algorithms, this question helped me to choose the HDBSCAN algorithm to achieve better results. As this algorithm is effective in group the similar data point to each other, it can handle the noise in the data and it also doesn't require specifying the number of clusters in advance.

3. **Q3: How can language models be applied for topic modelling of German datasets?**
This question was asked in order to know how the language models can be applied to the German dataset. As well all know about the language models and how powerfull they are can be used used for multilingual and cross lingual applications. So, the language models can be used for the German language datasets. However, to achieve topic modelling through language models is not an easy task. As the LLM model can't be used to directly extract the topic from the data and this processing of large amount of data cannot be handled by LLM model. So, this question helped to understand the potential of the approaches and as mention in research question 1, about the thesis approach where the data will undergoes embedding and clustering then, the clustered data will be fed to LLM model through the prompt to generate the topic names. In this way the LLM model can be used to extract the topic from the huge dataset. There are many advantages of using the LLM model to extract the topic when compare to traditional approaches, as the traditional methods will extarct the topic based on the occurrence of the words in the data but, the LLM model can understand the context of the sentences in the cluster and generate the topic names. There are many open source LLm model available and can be used based on the computational power of the device.

6.3 Future Work and Challenges

The methods used in this thesis were effective for extracting the topics from the historical interview transcripts along with the corresponding challenges. Firstly, the noise present in the datast can be managed in much better way. The noise are not removed completely which is ending up in forming a seperate cluster which is full of noise. Secondly, with the limited available of the computational power is major challenge faced as the embedding model and language model needs high computational power to process the data. Latest version of LLMs were not used due to the high computational power. Thridly, handling of the noise in the data, as the removal stopwords and normalizing the data will have higher chance of losing the context of the information. Hence, the small amount of noise has been removed after looking into the cluster results but, the noise are not removed completely.

In Future, with the help of better computational resources the challenges of using bigger and better embedding models and language models can be utilized for even better results. The prompting to the language models can be improved which will result in better topic name generation. Hyperparameters can be fine-tuned to get even better results. The web application can also be improved by using the better UI design and better styling.

Appendix A. Source Code

The Source Code is available on GitHub.

GitHub Link: https://github.com/praveenbm1997/Mapping-Memories-Exploring-Topics-in-Historical-Biographical-Interviews

Bibliography

GeeksforGeeks (2024). Clustering metrics.

Python Documentation (2023). Python documentation - the python standard library.