Contents

1 Building Data - A Chatbot for the Data Science Study Programme at FHNW

1.1 Introduction

This documentation will go over how we built Data, the Chatbot for the Data Science Study Programme at FHNW. It will cover the following topics:

- Concept
- Methodology
  - Step 1: Classification
  - Step 2: Question Answering
  - Step 3: Concern Path
  - Step 4: Assemblying the Chatbot
- Future Work
- Conclusion
- References

We refer to the specific implementations of the individual components for more details, which are liked here. Each repository has its own README, which explains the implementation in detail, and further references relevant notebooks and python scripts if you want to dive deeper into the code.

1.2 Concept

Our project started based on the definition of the Chatbot Concept and the Chatbot Architecture. The concept, outlining planned features and the architecture, can be read here.

1.2.1 Ziel

Ziel dieser Challenge ist die Entwicklung des Chatbots namens Data. Er soll den Studenten vom Studiengang Data Science zur Verfügung stehen, und ihnen Fragen rund um den Inhalt der Spaces zu den Modulen beanworten können, wobei er eine Persönlichkeit besitzen und vorgegebenen ethischen Leitlinien folgen soll. Data soll auf die Inhalte der Modul-Spaces zugreifen und Standardanfragen mit Hilfe einer Wissensbasis beantworten können.

Der Bot soll auch Probleme des Benutzers erkennen und darauf moralisch adäquat reagieren, zum Beispiel mit aufmunternden Worten oder mit der Weitergabe an eine Ansprechperson. Er soll zudem zur Motivation der Studierenden beitragen.

Wir legen unseren Fokus in der Challenge darauf, eine Version des Bots zu bauen, die gut auf Deutsch funktioniert (Sprache der meisten Inhalte in der Wissensbasis). Dabei ist nicht ausgeschlossen, dass er auch auf Englisch funktioniert, aber wir werden uns nicht explizit darauf fokussieren.

1.2.2 Fähigkeiten des Bots

Folgende Informationen soll der Bot auf Anfrage bereitstellen können:

- Seine Fähigkeiten erläutern
- Details zum Aufbau des Studiengangs (Konzept, Handbuch, Curriculum, Reglement)
- Details zum Modul
  - Fachexperten
  - Sprache
  - ECTS
  - Typ
  - Level
- Inhalte des Tabs “Porträt” der Module
- *optional*:
  - Wissen aus den PDFs in den Lernmaterialien der Module bereitstellen
  - Lernmaterialien vorschlagen
  - Tab “Aufgaben”

In seinem Verhalten soll der Bot folgende Persönlichkeitsaspekte berücksichtigen:

- Die Benutzer duzen (wie im SG Data Science üblich)
- Ethisch und moralisch korrekt sprechen (gerecht, wertschätzend)
- Eine motivierende, humorvolle und empathische Persönlichkeit besitzen

- Einen Dialog mit dem Benutzer führen können und über seine eigene Geschichte Bescheid wissen
- Auf Probleme des Users adäquat reagieren, wenn er diese erkennt (z.B. Stress im Studium, Unzufriedenheit, depressive Phasen) und Kontaktinformationen von Ansprechpersonen bereitstellen

1.2.2.1 Was der Bot NICHT können soll

- Mehrsprachigkeit explizit unterstützen
- Auf Inhalte der Lernmaterialien zugreifen können (z.B. PDFs oder externe Links)
- Wir werden im abgesteckten Rahmen der Challenge keinen direkten Aufwand in die Verhinderung von Prompt Injection u.ä. investieren

1.2.3 Wissensbasis

Die Wissensbasis des Bots soll auf mehreren Quellen aufbauen. Dies sind die zur Bereitstellung seiner Fähigkeiten notwendigen Informationen, einerseits aus dem Spaces DB Dump, andererseits aus den PDFs zum Studiengang (Konzept, Handbuch, Curriculum und Reglement).

Dabei soll der Bot bei der Beantwortung der Fragen den mitgeliefierten Kontext priorisieren. Wird eine Frage gestellt, die der Bot nicht auf Basis vom vorhandenen Kontext aus der Wissensbasis beantworten kann, deklariert er dies und greift zur Beantwortung der Anfrage entweder auf das Wissen im LLM zurück (z.B. bei "Was ist eine lineare Regression?"), oder lehnt die Beantwortung der Anfrage ab.

Die Inhalte können in Deutsch wie auch in Englisch vorhanden sein, was wir bei der Entwicklung berücksichtigen.

1.2.4 Design

Wir gestalten einen Avatar für "Data", der im Chat Interface angezeigt wird.

- Avatar Bild
- *optional*: Synthetische Stimme

1.2.5 Architektur & Tech Stack

Der nachfolgenden Skizze kann die geplante Architektur entnommen werden. Die einzelnen Komponenten werden im Folgenden kurz erläutert, wobei Änderungen an der Architektur oder der eingesetzten Technologien im Verlauf der Challenge nicht ausgeschlossen sind.

1.2.5.1 Chat Interface Der Bot steht dem Nutzer in einem simplen Web Chat Interface zur Verfügung. Dieses wird mit Streamlit umgesetzt. Die Logik

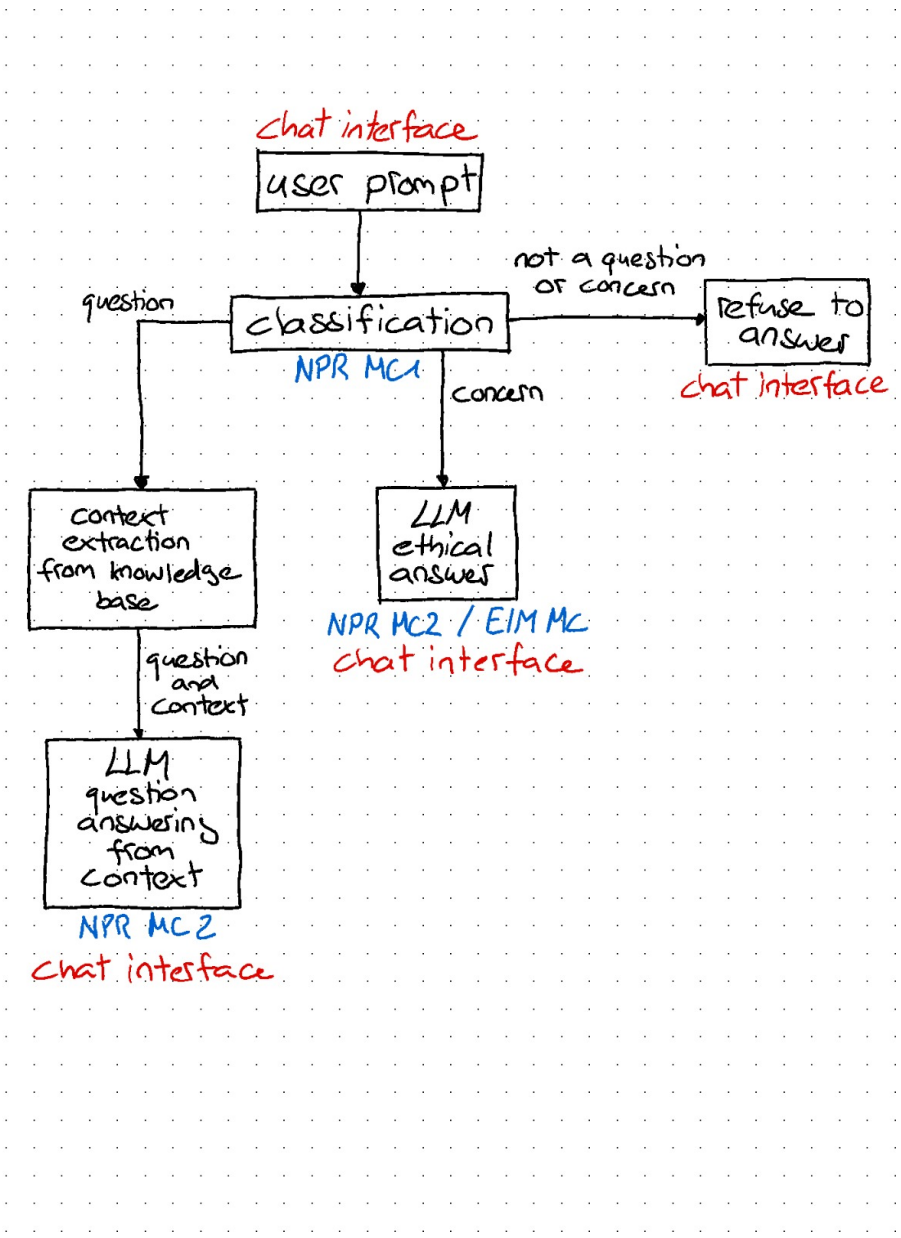

Figure 1: Architecture Sketch

zur Verarbeitung von Anfragen durch den Bot bauen wir mit Python, wobei wir LangChain verwenden um mit den verschiedenen LLMs zu kommunizieren.

Tech Stack:

- Chat Interface mit Streamlit (alternativ Gradio von HuggungFace)
- *voraussichtlich*: LangChain (Für Embedding und Kommunikation mit LLMs)

1.2.5.2 Prompt Classification (auch: npr MC1) Der Bot unterscheidet an erster Stelle zwischen 3 Arten von Anfragen:

- question
- concern
- not a question or concern

Mögliches Vorgehen:

- In einem Experiment evaluieren, wie viele Trainingsdaten benötigt werden, bis das Modell eine gute Performance in der Klassifikation erreicht
- Fine-tunig eines BERT-Modells zur Klassifikation der User Prompt

Tech Stack:

- HuggingFace Transformers library
- LLM: BERT Base Multilingual or similar

1.2.5.3 LLM für "concern" (auch: npr MC2 und eim MC) Der Bot geht auf die Anliegen des Users ein, wenn er diese erkennt. Er soll dabei empathisch und motivierend reagieren, und dem User bei Bedarf Kontaktinformationen von Ansprechpersonen bereitstellen.

Mögliches Vorgehen:

- Fine-tuning eines LLAMA2-Modells auf die ethischen Leitlinien für die Beratung und Unterstützung des Users im Bezug auf sein Anliegen.

Tech Stack:

- HuggingFace Transformers library
- LLM: LLAMA2-13B-Chat or similar

1.2.5.4 LLM für "question" (auch: npr MC2) Der Bot beantwortet die Frage des Users, wenn er sie versteht und die Antwort in der Wissensbasis vorhanden ist. Ansonsten lehnt er die Beantwortung der Frage ab, oder weist den User darauf hin, dass in der Wissensbasis dazu nichts vorhanden ist, und versucht mit internem LLM-Wissen weiter zu helfen.

Mögliches Vorgehen:

- Chunking und Embedding des Kontexts in der Wissensbasis.

- Fine-tuning/Instruction-tuning eines LLAMA2-Modells auf die Beantwortung der Fragen aus gegebenem Kontext.

Tech Stack:

- Embeddings (BERT/Open AI)
- Vector Storage PGVector
- HuggingFace Transformers library
- LLM: LLAMA2-13B-Chat or similar

1.2.6 Zielgruppe

Die Zielgruppe des Bots sind die Studierenden des Studiengangs Data Science. Wir haben zwei Personas definiert, die die Zielgruppe repräsentieren.

1.2.6.1 Persona 1: Anna, die eifrige Studentin

1.2.6.1.1 Demografische Daten

- Alter: 23
- Geschlecht: Weiblich
- Beruflicher Hintergrund: Lehre als Informatikerin

1.2.6.1.2 Persönlichkeit

- Ehrgeizig und fokussiert
- Detailorientiert
- Liebt es, frühzeitig zu planen

1.2.6.1.3 Bedürfnisse und Ziele

- Will den besten Überblick über ihre Module haben
- Sucht immer nach zusätzlichen Ressourcen für bessere Lernerfolge
- Möchte auf dem Laufenden bleiben, was Änderungen im Curriculum betrifft

1.2.6.1.4 Nutzungsszenarien

- Fragt den Bot nach den Leistungsnachweisen in spezifischen Modulen
- Will wissen, welche Fachexperten für ein Modul zuständig sind
- Plant das nächste Semester und fragt den Bot nach Modulen im Curriculum
- Wird vom Bot für ihre gute Arbeit gelobt und fühlt sich motiviert

1.2.6.2 Persona 2: Markus, der berufstätige Student

1.2.6.2.1 Demografische Daten

- Alter: 29
- Geschlecht: Männlich
- Beruflicher Hintergrund: Arbeitet Teilzeit im Rechnungswesen

1.2.6.2.2 Persönlichkeit

- Pragmatisch und zielorientiert
- Legt Wert auf Work-Life-Study-Balance
- Etwas stressanfällig aufgrund der vielen Verpflichtungen

1.2.6.2.3 Bedürfnisse und Ziele

- Sucht nach einem effizienten Weg, die Studieninformationen zu konsultieren
- Will möglichst wenig Zeit mit der Suche nach grundlegenden Informationen verbringen
- Sucht nach einer schnellen Möglichkeit, seine Fragen zu klären, um sich auf seine Arbeit und das Studium zu konzentrieren

1.2.6.2.4 Nutzungsszenarien

- Will schnell wissen, wie viele ECTS ein Modul hat
- Möchte erfahren, was ihn in einem spezifischen Modul erwartet
- Nutzt die motivierenden und empathischen Funktionen des Bots, um Stress abzubauen

Diese Personas können als Grundlage für die Entwicklung des Chatbots "Data" dienen. Sie repräsentieren die Bedürfnisse und Ziele der Zielgruppe und können dazu beitragen, die Funktionalität und das Verhalten des Bots optimal auszurichten.

1.2.7 Privatshpäre

Wir nutzen unsere eigenen fine-tuned Modelle, dementsprechend verlassen keinerlei sensible Daten unser System. Der Chatbot ist nur für Studenten des Studiengangs Data Science zugänglich, und damit sind auch nur Informationen verfügbar, auf die die Studenten ohnehin Zugriff haben.

1.2.8 Evaluierung

Ein Chatbot, der falsche Antworten gibt, oder nicht auf die Fragen des Benutzers eingeht, ist nicht hilfreich. Er soll deshalb auf verschiedene Arten evaluiert werden.

1.2.8.1 Prompt Classification

Quantitativ:

- F1-Score
- Accuracy

Qualitativ:

- Indiduelle Beispiele testen auf mehreren Modellen für Vergleich

1.2.8.2 LLM für "concern"

Quantitativ:

- Nicht geplant

Qualitativ:

- Individuelle Beispiele, um zu testen, ob der Bot den ethischen Leitlinien folgt

1.2.8.3 LLM für "question"

Quantitativ:

- Retrieval (Werden die relevanten Chunks identifiziert)
- BLEU Score (Wie gut ist die Antwort)

Qualitativ:

- Individuelle Beispiele, um die Brauchbarkeit der Antworten zu testen

1.3 Methodology

1.3.1 Step 1: Classification

Find the implementation and all details on the Classification here: Text-Classification

1.3.1.1 Objective

The primary objective was to enable our chatbot, named Data, to classify user prompts into one of three categories: questions, harm, or concerns. This classification is vital for the chatbot to respond appropriately to user interactions. For details on why this is vital to our architecture, see Concept.

1.3.1.2 Data Selection and Preparation

We selected three datasets for creating a comprehensive training dataset for our classifier. These included:

- **GermanQUAD for Questions**: This dataset provided a robust set of question-answer pairs in German, ideal for training our model to recognize user queries.
- **GermEval for Harm**: This dataset was chosen to help the classifier identify harmful or inappropriate content.
- **Stress-Annotated Dataset (SAD) for Concerns**: To enable the chatbot to recognize and appropriately address user concerns, especially those of a sensitive or personal nature.

A qualitative assessment using a BERT classifier suggested the feasibility of our approach. We then proceeded with a train-test split to prepare the datasets for model training.

1.3.1.3 Model Selection and Training

We chose three model architectures:

1. **LinearSVC with TF-IDF**: Selected as a baseline model to compare against more advanced deep learning approaches. LinearSVC was chosen based on its superior performance among seven tested machine learning classifiers.
2. **LSTM-CNN**: This model was considered due to the LSTM's ability to handle sequential data, potentially learning relationships between word semantics. The CNN was employed for initial feature extraction before feeding data into the LSTM. Pre-trained embeddings (bert-base-german-cased) were utilized, anticipating a reduction in overfitting risk and less need for extensive training data.
3. **BERT Fine-Tuning as Classifier (bert-base-multilingual-cased)**: This model was selected for its support of both English and German sentence structures, coupled with the expectation that its pre-training would require less training data.

1.3.1.4 Evaluation and Improvement All models performed well on the test dataset, with BERT achieving the highest accuracy (98%). However, when tested on a benchmark dataset comprised of examples created by our team, all models showed a significant drop in performance (Accuracy: SVC 0.60, LSTM-CNN 0.62, BERT 0.67). Based on these results, we decided to continue with BERT as our primary classifier model.

To enhance model performance, we created a synthetic dataset using GPT-4 with tailored prompting. This dataset, comprising queries from all three classification categories, led to a marked improvement in BERT's performance on our benchmark dataset.

We also recognized some misclassifications in basic queries, such as "Hallo was kannst du" being incorrectly categorized as harm. To address this, we extended our synthetic dataset with a "human feedback" component, that can be continually updated based on flagged results from production use, to further refine accuracy.

1.3.2 Step 2: Question Answering

TODO: add retrieval information somewhere!

Find the implementation and all details on the Question Answering here: Implementation

1.3.2.1 Objective The aim was to develop a large language model (LLM) capable of providing context-informed, abstractive answers to user queries.

1.3.2.2 Data Selection and Preparation We selected the GermanQUAD dataset, which includes questions, contexts, and corresponding answers in German. This dataset was used to train the LLM in extracting answers from the provided context.

1.3.2.3 Initial Approach and Realization Initially, we aimed for the model to learn answer extraction from context by fine-tuning it with examples from the GermanQUAD dataset. However, we observed that the answers generated were extractive, mirroring the dataset's nature. To achieve our goal of abstractive response generation, we modified our approach by using GPT-3.5 to create abstractive answers from questions, contexts, and the original answers.

1.3.2.4 Context Swapping and Source Referencing Context swapping was incorporated into our methodology, as we recognized the importance of the model learning to not provide answers when relevant context was unavailable. We also trained the model to refer to sources at the end of its abstractive responses by splitting the available context into multiple blocks and adding synthetically generated sources to these blocks.

1.3.2.5 Model Training and Selection We selected and trained three different LLMs:

- Baseline: meta-llama/Llama-2-13b-hf
- German-optimized Llama-2-13b: flozi00/Llama-2-13b-german-assistant-v7
- German-optimized Mistral-7b: VAGOsolutions/SauerkrautLM-7b-v1-mistral

We opted for base models over chat variants for the learning experience, despite potentially lower overall performance.

1.3.2.6 DVC Stages Using a DVC pipeline allowed us to automate the retrieval and fine-tuning stages of our methodology. This pipeline was designed to be run on a GPU-enabled machine.

1.3.2.6.1 Retrieval Stages The retrieval stages aimed to create chunks from relevant documents (Data Science study program documents, course materials, etc.) and save them in a ChromaDB vector store. This store was later queried for relevant documents in response to user queries.

1.3.2.6.2 Fine-tuning Stages The fine-tuning stages were designed to enhance the language model's question-answering abilities using our modified GermanQUAD dataset. The resulting fine-tuned models can then be used to generate abstractive answers to user queries.

1.3.2.7 Dataset, Models, and Experiments

1.3.2.7.1 Fine Tuning Dataset The dataset, derived from GermanQUAD, was modified to include both answerable and unanswerable questions, with abstractive answers generated using GPT-3.5. The dataset was split into training, validation, and test sets.

1.3.2.7.2 Fine Tuning Models We conducted experiments with three different language models:

- Baseline: meta-llama/Llama-2-13b-hf
- German-optimized Llama-2-13b: flozi00/Llama-2-13b-german-assistant-v7
- German-optimized Mistral-7b: VAGOsolutions/SauerkrautLM-7b-v1-mistral

1.3.2.8 Evaluation We performed quantitative and qualitative evaluations to ascertain the best-performing model in terms of generating context-informed, abstractive answers.

1.3.3 Step 3: Concern Path

For the concern path, we built a prompt upon ethics theory. Find the comprehensive report including evaluation here: Concern Path

The functionality is built directly into the Chatbot Dashboard, see dashboard.py.

1.3.4 Step 4: Assemblying the Chatbot

After the individual components were developed, we assembled the chatbot using the Hugging Face Gradio framework. It allowed us to create a simple, user-friendly interface for the chatbot, which can be accessed via web browser. It has a Chat Tab, a Retrieval Tab and a Classification Tab. This way, each component can be tested individually, which is immensely helpful for debugging and improving the chatbot.

The Chat Tab is the main interface for the user. It first uses the classifier to determine the type of the user input. If the input is a question, it is passed to the Question Answering module, which retrieves relevant contex and generates the answer. If the input is a concern, the Concern Path is triggered, which generates an appropriate messasge to continue the conversation. If the input is harm, the chatbot responds with a predefined message that denies the user's request.

The implementation can be found in dashboard.py.

For setup instructions, see README.md.

1.4 Future Work

- When we would move towards deploying "Data" in a production environment, our focus would be on continual retraining and improvement of the classifier. This will be achieved by leveraging new human feedback to refine the model's accuracy and adaptability to real-world interactions.
- Our next steps involve continuous improvement of the question-answering module based on real-world interactions and user feedback.
- Another important improvement would be building a history into the Question Answering module, so that the chatbot can remember previous

questions and answers and use them to generate more context-informed answers.
- Instead of opting for a solely classifier, we would include the help of an LLM in that task, and instead of having either the Question Answering model or the Concern model responding, we would opt for one model with both functionalities built into it. This could for example be achieved by the combined model using a response generated by a separate model focused on QA, and using that in it's response, or fine-tuning a model to have both capabilities.

1.5 Conclusion

TODO

1.6 References

TODO