Hochschule
für Technik
Stuttgart

Betreutes Praktisches Studienprojekt

Praktikumsbericht

im Rahmen des Studiengangs
Wirtschaftsinformatik
der Hochschule für Technik Stuttgart

vorgelegt von:

Julian Raubald

(Matrikelnummer 1003812)

Praktikumsbetrieb:

iteratec

Betreut durch:

Prof. Dr. Stefan Knauth

Stuttgart, Tag. Februar. 2024

Inhaltsverzeichnis

1 Einleitung

In diesem Bericht schildere ich die Aufgaben und Projekte, welche ich in meinem Praxissemester im Wintersemester 23/24 bei dem Unternehmen iteratec GmbH durchgeführt und begleitet habe. Dabei gehe ich auf die eingesetzten Technologien ein und teile meine gesammelten Erfahrungen und ordne diese inhaltlich im Rahmen meines Studiums der Wirtschaftsinformatik ein.

1.1 Unternehmen iteratec GmbH

Iteratec ist ein Unternehmen, das sich auf digitale Produktinnovation, Software- und Architekturentwicklung sowie digitale Infrastrukturen spezialisiert hat. Als Partner für die digitale Transformation unterstützt es Unternehmen und öffentliche Organisationen bei der Digitalisierung ihrer Prozesse, Produkte und Dienstleistungen. Dies umfasst die Entwicklung von Innovationen, die technische Umsetzung individueller Softwarelösungen, deren Betrieb und Weiterentwicklung sowie Schulungen in agilen Methoden. Das Unternehmen entwickelt mobile Applikationen, Cloud-Architekturen und bietet Application Performance Management an. Es nutzt auch Technologien wie Blockchain, AI und IoT für die Entwicklung neuer Geschäftsmodelle. Zu den Kunden gehören mittelständische Unternehmen, DAX-Konzerne und Organisationen aus verschiedenen Branchen. 1996 in München gegründet, hat iteratec Standorte in Deutschland, Österreich und Polen und beschäftigt rund 500 Mitarbeiter und 120 Studierende. Seit 2019 sind viele Mitarbeiter über eine Genossenschaft am Unternehmen beteiligt.

1.2 Name und Geschichte

Das Unternehmen wurde 1996 gegründet und existiert bereits seit über 25 Jahren. Während zu dieser Zeit das Wasserfallmodell in der Softwareentwicklung weit verbreitet war hat sich iteratec schon früh auf agile Methoden und Werte spezialisiert. Erkennbar ist das insbesondere an dem Namen, welcher sich aus den Begriffen iterativ und Technologie zusammensetzt. Heute hat iteratec bereits über 1000 Softwareprojekte erfolgreich abgeschlossenen.

1.3 Unternehmensstruktur

Standortübergreifend hat iteratec 4 Hauptgeschäftsführer. An jedem Standort gibt es jeweils einen Geschäftsstellenleiter. Jeder Mitarbeiter hat eine persönliche Führungskraft (pFK), welche für die persönliche Entwicklung vor allem im, aber auch außerhalb des Unternehmens zuständig ist. Jedes Projekt hat einen Product Owner (PO), welcher die Interessen des Kunden im Projekt vertritt, und einen Projektleiter, welcher dafür zuständig ist die Fristen und eingesetzten Ressourcen zu managen. Alle Mitarbeiter begegnen sich Standort- und Positionsübergreifend auf Augenhöhe, was sich unter anderem dadurch zeigt, dass sich alle duzen. Die bisherigen Alleingesellschafter der iteratec GmbH haben im Jahr 2018 entschieden, das Unternehmen weder innerhalb ihrer Familie zu vererben noch dieses an externe zu verkaufen. Daraus entstand die iteratec nurdemtean eG, eine Genossenschaft bei der sich die Mitarbeiter am Unternehmen beteiligen, mit dem Ziel, die Unternehmenskultur zu halten.

1.4 Studierende bei iteratec (SLAB)

Die Abteilung für studierende, das SLAB (Studentenlabor), ist Standort übergreifend und schließt sowohl alle Studierenden als auch deren Betreuer mit ein. Halb-jährlich werden neue SLAB Leiter gewählt, deren Aufgabe es ist, alle anfallenden Aufgaben an studierende zu delegieren und im engen Austausch mit dem SLAB-Verantwortlichen zu stehen. Eine Aufgabe ist das bereitstellen einer Startbegleitung für neue Studierende.

Andere Aufgaben sind z.B. das organisieren von Workshops und Events.

1.5 Meetings und Veranstaltungen

Bei iteratec gibt es abseits der regulären Projektmeetings viele weitere Veranstaltungen an denen man teilnehmen kann. Im folgenden Kapitel erläutere ich die wichtigsten.

1.5.1 Barcamp

https://wiki.iteratec.io/display/CREATE/Barcamp

1.5.2 SLAB-Meeting

Das SLAB-Meeting findet zwei mal pro Jahr statt und beschränkt sich auf den Standort. Hier wird die SLAB-Leitung gewählt, welche immer aus zwei Personen besteht. Es wird retrospektiv geschaut und bewertet wie gut die Prozesse rund um die Studentenbetreuung funktionieren. Es werden Verbesserungsvorschläge gesammelt und nach Nutzen und Durchführbarkeit bewertet. Anschließend werden entsprechende Maßnahmen an verantwortliche verteilt die sicherstellen, dass diese durchgeführt werden.

1.5.3 GS-Meeting

Das Geschäftsstellen-Meeting für die Geschäftsstelle Stuttgart, an dem alle Mitarbeiter des Standorts eingeladen sind, findet etwa ein Mal pro Monat statt. Hier werden aktuelle Themen besprochen und Mitarbeiter können Ihre Projekte vorstellen. Die Geschäftsführung verkündet während des Meetings wichtige Neuigkeiten, anstehende Events und präsentiert die aktuellen Geschäftszahlen. Neue Mitarbeiter und studierende werden begrüßt und ausscheidende Mitarbeiter verabschiedet. Zum

Ende können Kollegen für die Wall of Fame in einer der drei Leitwerte von iteratec (Permanent Exzellent, Konstruktiv unabhängig und Inspirierend mutig) nominieren und so besondere Leistungen und gute zusammenarbeit würdigen.

1.5.4 BUILD23

Bei der BUILD handelt es sich um eine Messe, welche ausschließlich von iteratec Mitarbeitern besucht werden darf. Sie geht über zwei Tage und fand dieses Jahr bei der Geschäftsstelle in München statt. Im Vorfeld kann jeder Mitarbeiter, Student oder jedes Team sich ein Thema überlegen und als Vorschlag einreichen. Nachdem die Themen freigegeben wurden plant man einen entsprechenden Messestand über drei bis sechs Zeitslots, in denen man sein Thema, Projekt oder Workshop präsentieren kann. Hierzu gibt es an jedem Stand Ausrüstung für eine Silent-Disco. Ziel der Build ist es das erlangte Wissen Projekt- und Standortübergreifend auszutauschen und sich mit Kollegen zu vernetzen oder Interessensgemeinschaften zu bilden. Im Anschluss an die Build findet die Weihnachtsfeier statt.

2 Tätigkeitsbereich und Aufgabenstellung

Im folgenden Kapitel werde ich genauer auf meinen Arbeitsbereich und meine Aufgaben eingehen.

2.1 Arbeitsbereich und Ausstattung

Sämtliche Arbeitsplätze sind sogenannte Flexarbeitsplätze. Das heißt, dass man sich morgens einen freien Schreibtisch sucht und dort seine Arbeit beginnt. Es gibt vereinzelt Bereiche wie das SLAB, wo sich bestimmte Gruppen zum arbeiten treffen aber auch dort hat niemand einen festen Arbeitsplatz. Jeder dieser Arbeitsplätze verfügt über einen höhenverstellbaren Schreibtisch und 2 Monitore inklusive Dockingstation für den Laptop. Jedem Mitarbeiter ist es gestattet seine Arbeit von Zuhause aus durchzuführen, solange man mindestens einen Tag in der Woche vor Ort im Büro arbeitet. Ein Großteil der Kommunikation mit Kollegen erfolgt daher über Videokonferenzen oder Chats mithilfe der Software Microsoft Teams. Für Meetings stehen zahlreiche Meeting räume zur Verfügung, welche man sich jederzeit reservieren kann. Für die Arbeit erhält man als Student einen Dell Laptop mit Windows-Betriebssystem.

2.2 Hauptaufgaben

Zu meinen Hauptaufgaben gehört insbesondere das entwickeln von Software, beziehungsweise das schreiben von Quellcode. Eigenständiges recherchieren und Problemlösung gehören dabei fest zum Arbeitsalltag. Entwickelt wird iterativ nach einer abgewandelten Kanban-Variation. So werden Tickets für einzelne Features oder gefundene Bugs in Jira im Backlog angelegt. Die Entwicklung erfolgt in mehreren Sprints, bei denen Aufgaben aus dem Backlog in Wellen entnommen und in die "NewSpalte geschoben werden. Sobald mit der Arbeit an einer Aufgabe begonnen wurde, schiebt man das Ticket in die În ProgressSpalte, bei erledigten Aufgaben kommen diese in die "ReviewSpalte, bei Problemen oder Unklarheit in die "Waiting-Spalte bis eine Entscheidung gefunden wurde wie man fortfahren will. Bei den ein- oder zwei-wöchentlichen Meetings werden die "NewTickets unter den Entwicklern aufgeteilt und mögliche Probleme oder Unklarheiten besprochen. Aufgaben, welche sich in "Review"befinden kommen in die "DoneSpalte falls alles wie gewünscht funktioniert und der Product Owner zufrieden ist. Da es keine festen WIP-Limits (Work in Progress) gibt kann nicht von reinem Kanban gesprochen werden, allerdings ist es eher unüblich dass ein Entwickler mehr als 2 Aufgaben parallel bearbeitet.

2.3 Verwendete Software und Grundlagen

In diesem Kapitel werden alle Software-Werkzeuge, Technologien und Begriffe beschrieben, welche während der gesamten Dauer meines Praktikums Projektübergreifend verwendet werden.

2.3.1 Technologien

2.3.2 Werkzeuge

Visual Studio Code

Visual Studio Code (VSCode) ist ein leistungsfähiger und anpassbarer Quelltext-Editor von Microsoft, der eine breite Palette von Programmiersprachen unterstützt und zahlreiche Erweiterungen für Entwickler bietet.

GitLab

GitLab ist eine auf Git basierende webbasierte DevOps-Plattform, welche für die Versionskontrolle und CI/CD bei der Softwareentwicklung verwendet wird. Für die Testdateien und Dokumentation verwenden wir zusätzlich LFS (Large File Storage) um größere Dateien im Repository abzulegen.

Microsoft Teams

Ein Kommunikationstool von Microsoft, welches genutzt wird um sich mit Kollegen auszutauschen, Dateien zu teilen und Besprechungen zu planen so wie durchzuführen.

Jira

Ein Projektmanagement-Werkzeug von Atlassian, das speziell für Softwareentwicklungsteams konzipiert ist und Funktionen für die Verfolgung von Aufgaben, Bugs und agile Prozesse bietet. Es kam insbesondere beim verwalten des Backlogs und beim bearbeiten von Tickets zum Einsatz.

Miro

Eine digitale, kollaborative Whiteboard-Plattform, die es unserem Team ermöglicht, visuell zu brainstormen, zu planen und Projekte in Echtzeit zu gestalten.

2.4 PERSPEKTIVWECHSEL!!!

3 TenderInsights

3.1 Kurze Projektbeschreibung

In dem Projekt geht es darum einen PoC - also einen Proof of Concept zu erstellen, der mithilfe von AI bzw. Large Language Models (LLM) große Dokumente nach gesuchten Inhalten durchsucht und aufbereitet. Im Fokus stehen dabei insbesondere öffentliche Ausschreibungen zur Akquise von neuen Projekten für das Unternehmen. Bisher hat ein Mitarbeiter aus PMO die Ausschreibungen händisch grob für passend oder nicht geeignet erklärt und dann an die Akquise weitergeleitet. Dort hat ein Mitarbeiter die Ausschreibung sehr genau analysiert und einen sogenannten One Pager erstellt, der alle Daten welche für eine Entscheidung, ob man sich bewerben möchte oder nicht, relevant sind. Aufgabe des PoC ist es nun diesen One Pager mithilfe von künstlicher Intelligenz und einem LLM aus einer oder mehreren bereitgestellten PDF-Datei zu generieren. Der Anwender soll dies über ein Userinterface im Stile einer Webanwendung bedienen können.

3.2 Verwendete Software und Grundlagen

In diesem Kapitel werden alle Technologien und Begriffe beschrieben, welche im Entwicklungsumfeld des TenderInsights verwendet werden oder für ein Verständnis zuträglich sind.

3.2.1 Technologien

Bei der Entwicklung des TenderInsights werden viele unterschiedliche Technologien eingesetzt. Während meines Praktikums habe ich in meiner Tätigkeit als Entwickler viele davon kennengelernt und verwendet. Nachfolgend erkläre ich die wichtigsten Technologien.

Python

Python ist eine höhere Programmiersprache welche sich durch ihre klare und leicht verständliche Syntax auszeichnet. Sie ist bekannt für ihre zahlreichen Standardbibliotheken und ihrer reichen Auswahl an Modulen und Frameworks. Da OpenAI nur JavaScript und Python unterstützt, und Python aus zuvor genannten Gründen einsteigefreundlicher ist wurde sich für diese Sprache entschieden.

Langchain

Langchain ist ein Open-Source-Framework, das auf die Entwicklung und integration von Sprach-KI-Anwendungen spezialisiert ist. Es erleichtert das einbinden von Sprachmodellen wie GPT-3.5-turbo von openAI in Projekte und wird beim TenderInsights für Textgenerierung verwendet.

OpenAI - GPT

OpenAI ist ein US-Amerikanisches Unternehmen welches sich mit der Erforschung und Entwicklung von künstlicher Intelligenz beschäftigt. Eines der bekanntesten Tools bzw. Frameworks ist GPT - Generative Pretrained Transformer, ein Sprachverarbeitungsmodell zur Textgenerierung, Übersetzung, Zusammenfassung und mehr. GPT wird im TenderInsights verwendet um Information aus Texten zu extrahieren und um die gelieferten Prompts zu bewerten.

Azure

Microsoft Azure ist eine Cloud-Computing-Plattform, welche eine breite Palette von Diensten und Lösungen für Unternehmen und Entwickler bietet. Zu den Entwickler-Tools und -Diensten gehört unter anderem Azure openAI, welches eine openAI API zur Verfügung stellt. Im Gegensatz zu openAI befinden sich die Server in Europa, weshalb im späteren Verlauf des Projekts aus Gründen des Datenschutzes zu Azure gewechselt wurde.

ChromaDB

ChromaDB ist eine Open-Source-Datenbanklösung, welche speziell für die Arbeit mit Vektor-Einbettungen in KI-Anwendungen entwickelt wurde. Da der Kontext für Sprachmodelle stark unter der Größe von durchschnittlichen Dokumenten liegt wird eine Vektor-Datenbank genutzt um die relevanten Stellen in den Dokumenten zu finden und als Kontext mit dem Prompt an das Sprachmodell für die Textgenerierung zu übergeben.

JSON

JavaScript Object Notation, oder kurz JSON, ist ein Daten-Austauschformat welches häufig verendet wird um Daten zwischen einem Server und einer Webanwendung zu übertragen. Da es für Menschen gut lesbar und für Maschinen einfach zu parsen (Entspricht einem normalen Dictionary in Python) ist eignet es sich hervorragend für das Speichern der OnePager Informationen.

3.2.2 Wichtige Begriffe

Tabelle 3.1: Wichtige Begriffe

Name	**Beschreibung**
Prompt	Anleitung oder Frage die dem Sprachmodell vorgibt, was es tun soll. Auch Eingabeaufforderung genannt.
Prompt Engineering	Gestalten und Optimieren von Eingabeaufforderungen (Prompts)
Token	Grundlegender Baustein für die Verarbeitung und das Verständnis von Sprache in KI-Systemen
Embedding	Einbetten von Text in Vektoren
Vektordatenbank	Datenbank welche die aus dem Embedding generierten Vektoren enthält
Similarity Search	Suche von nahestehenden Vektoren innerhalb der Vektordatenbank um gesuchte Dokumenteninhalte zu finden
Large Language Model (LLM)	Fortgeschrittene KI, die umfangreiche Textdaten versteht und generiert, basierend auf umfassendem Training und komplexen Algorithmen

3.3 Projektstand bei Arbeitsbeginn

Zum Zeitpunkt meines Projekteintrittes gibt es bereits ein Frontend welches mithilfe von Streamlit implementiert wurde. PDF-Dateien können über einen File-Uploader im Frontend an die Anwendung übergeben werden. Da die meisten LLM-Modelle eine begrenzte Token-Anzahl für die Abfragen haben ist es nicht möglich das gesamte Dokument an das LLM zu übergeben. Daher wird das Dokument aufgesplittet und mittels einer Vektordatenbank auf Ähnlichkeit mit die gesuchte Information abgeglichen. Die k Ergebnisse mit der größten Ähnlichkeit werden zusammen mit dem Prompt als Abfrage an das LLM übermittelt. Die Abfragen werden mit der Langchain API durchgführt, das verwendete Modell ist "gpt-3.5 turbo". Es gibt 3 Ausschreibungsdokumente in verschiedenen Größen mit denen die Abfragen getestet

werden können. Es gibt 3 Prompts, welche bislang nur bedingt gute Ergebnisse liefern.

3.4 Softwarearchitektur der Anwendung zum Zeitpunkt der BUILD23

Um nachfolgende Kapitel besser verstehen zu können erkläre ich in diesem Kapitel die Grobe Software-Architektur der Anwendung zum Zeitpunkt der BUILD23 anhand der Abbildung 3.1.

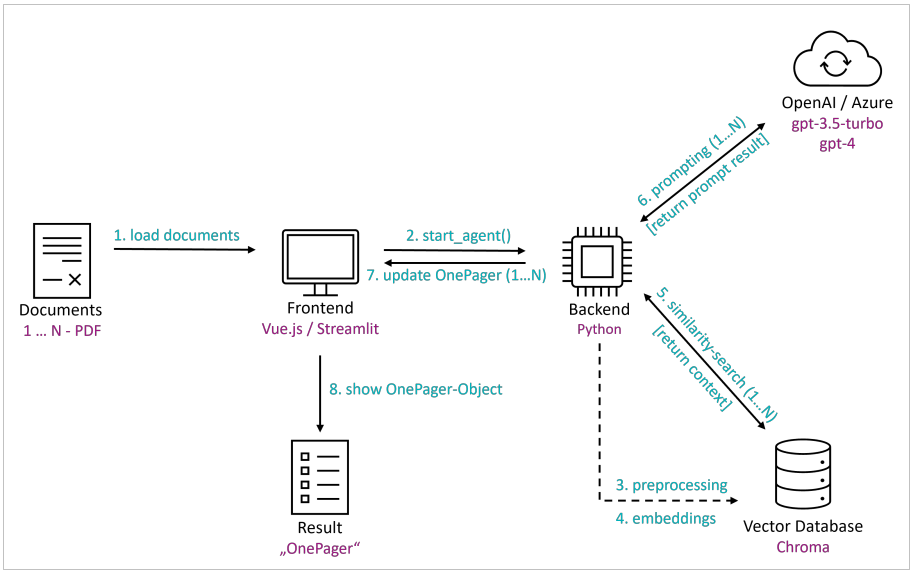

Abbildung 3.1: Schematische Darstellung der Software-Architektur

Zuerst werden die Ausschreibungsdokumente im Frontend hochgeladen (Siehe 1.) und der Agent wird mit den gewünschten Prompts gestartet (Siehe 2.). Der Agent beginnt nun im Backend mit dem PreProcessing, welches die Dokumente in fachlich logische Blöcke aufspaltet und personenbezogene Daten anonymisiert (Siehe 3.). Anschließend wird das Embedding, also eine Repräsentation der Daten als Vektoren, erstellt und als der Vektordatenbank angelegt (Siehe 4.). Nun wird für jeden Prompt eine Similarity-Search durchgeführt (Siehe 5.), bei der dem möglichst passende Stellen aus den Dokumenten aus der Vektordatenbank extrahiert werden, damit diese im Prompt als

Kontext eingebettet für die Textvervollständigung an das LLM übergeben werden (Siehe 6.). Das Ergebnis wird in ein JSON-Objekt übersetzt und der OnePager wird entsprechend aktualisiert (Siehe 7.). Währenddessen baut sich im Frontend Stück für Stück der OnePager mit allen gefundenen Informationen zusammen (Siehe 8.).

Anf: Datenschutz, große seiten, chatten mit Dokument Grundlagen: Token, Similarity Search, Prompts

3.5 Meine Rolle und Aufgaben im Projekt

Ich bin in der Rolle eines Softwareentwickler in das Projekt gekommen. Meine Aufgabe bestand Anfangs darin, die Prompts für die einzelnen Felder des One Pagers zu formulieren um die gelieferten Ergebnisse zu verbessern. Dies resultierte in den Aufbau einer Test- und Evaluierungsarchitektur in Python mithilfe derer die Prompts vollautomatisch bewertet werden und anhand ihrer Kennzahl ausgewählt und weiter optimiert werden. Durch Komplikationen mit dem Datenschutz wurde der Umstieg von Langchain auf Azure von Microsoft beschlossen, welchen ich durchgeführt habe.

3.5.1 Prompt Engineering

Um gute Ergebnisse durch die Textvervollständigung der Sprachmodelle zu erhalten ist es unumgänglich hochwertige Prompts zu formulieren. Hierfür habe ich mehrere E-Learnings zum Thema Prompt Engineering absolviert, in denen erklärt wird wie man hochwertige Prompts erstellt und ganze Systeme mit einer Chatbotintegration entwickelt. Besondere Methoden welche im Projekt zum Einsatz kommen sind Single-Shot-Prompting, bei dem man ein Beispiel gibt wie eine optimale Antwort auszusehen hat, und die Ausgabe als JSON festzulegen, um die erhaltenen Ergebnisse im Code nutzen und anwenden zu können. Das Abgrenzen von wichtigen Informationen wie dem mitgelieferten Kontext über sogenannte Delimiter hilft dem Modell dabei, Aufgabe und Kontext nicht zu vermischen (wenn auch nicht immer perfekt). Gut eignet sich hierfür '####', da dies als ein Token zählt und selten regulär vorkommt. Um Halluzinationen, also das produzieren von Falschinformationen oder ausgedachten

Informationen zu vermeiden wird im Prompt explizit darauf hingewiesen, dass bei nicht vorhandenen Informationen das entsprechende Feld leer gelassen werden soll.

3.5.2 OnePager

Der von PMO erstellte OnePager umfasst alle wesentlichen Information der Ausschreibung um eine fundierte Entscheidung darüber treffen zu können, ob die Ausschreibung als Projekt für iteratec geeignet ist. Informationen sind unter anderem der Projekttitel, eine kurze Projektbeschreibung, Adressdaten über das Ausschreibende Unternehmen oder Amt, wichtige Termine und Deadlines, Bewertungskriterien, Aufgaben und benötigte Unterlagen welche dem Angebot beigefügt werden müssen. Da jede dieser Punkte in unterschiedlichen Regionen der Dokumente zu finden ist und die Qualität der Ergebnisse besser ist wenn man nicht mehrere Fragen in einer Anfrage bündelt wird der OnePager in der Anwendung Frage für Frage zusammengesetzt und ergänzt. Als Format wird JSON verwendet um später den fertigen OnePager gut speichern und weiterverarbeiten zu können (zum Beispiel für die Evaluation) und um dem Frontend die Daten als Variable zur Verfügung zu stellen. Die OnePager-Klasse wurde wurde um viele Hilfsmethoden ausgestattet um das Arbeiten mit den JSON-Objekten zu erleichtert. Zu den Hilfsmethoden gehören unter anderem das initialisieren des OnePager-Template, das laden und abspeichern als Datei, einfaches hinzufügen von Teilinformationen aus den einzelnen Ergebnissen des Sprachmodells in den bestehenden OnePager und Prüffunktionen um nicht JSON-Konforme Antworten zu erkennen.

3.5.3 Verbessern der Kontextauswahl über Similarity Search

Um gezielt die fachlich richtigen Passagen innerhalb des Dokuments zu finden wird eine query gegen die über das Embedding generierte Vektordatenbank gestellt. Als Ergebnis erhält man die n-Textausschnitte, mit der höchsten Trefferquote.

```
1     results = collection.query(
2         query_texts=["This is a query document"],
```

```
3         n_results=2)
```

Listing 3.1: Code einer Similarity Search

Bei dem eben beschriebenen Verfahren sprechen wir von einer Similarity Search, also dem finden von ähnlichen Daten innerhalb einer Vektorrepräsentation des bereitgestellten Dokuments. Da es bei Ausschreibungen viele synonyme Begriffe gibt, welche Informationen zu identischen Fragestellungen geben habe ich diese Begriffe in einem Lexikon erfasst und gesammelt. Die Akquiseabteilung wurde zusätzlich darum gebeten weitere Ergänzungen vorzunehmen um fachlich möglichst viele Begriffe abzudecken. Mit dem so erhaltenen Lexikon können wichtige Keywords für die Similarity Search bereitgestellt werden, vereinzelt wird auch innerhalb der Prompts darauf zurückgegriffen um das Sprachmodell dabei zu unterstützen die richtigen Informationen der richtigen Stelle des OnePagers zuzuordnen. Über ein Prompt.Config Dictionary kann nun für jedes Prompt auf entsprechende Felder wie Suchbegriffe, Promptmethode oder Suchergebnisanzahl individuell zugegriffen werden.

3.5.4 Evaluation

Um festzustellen ob sich Prompts beim Wechsel von Modellen, Promptversionen oder unterschiedlichen PreProcessing-Verfahren verbessern oder verschlechtern habe ich 10 Testdatensätze aus öffentlichen Ausschreibungsportalen zusammmengesucht und entsprechende Musterlösungen erarbeitet, sofern dies in unserem fachlichen Rahmen möglich war. Dies hat uns ermöglicht, ausgewählte Prompts testweise gegen alle 10 Testdatensätze abzufragen. Hierbei wird die neu generierte Antwort mit der Musterlösung verglichen und anschließend anhand eines Bewertungsschemas mit Punkten zwischen 0 und 10 inklusive einer Begründung bewertet, je nachdem wie nah das gelieferte Ergebnis an die Musterlösung herankommt.

Die einzelnen Punkte werden anschließend für jedes Prompt zu einem Average Score zusammengerechnet und mithilfe der Visualisierungsklasse als Plot dargestellt. Dabei können einzelne Prompts über verschiedene Versionen und Zeitpunkte (siehe 3.2) oder mehrere Prompts bei Verwendung unterschiedlicher Sprachmodelle (siehe 3.3) dargestellt und verglichen werden.

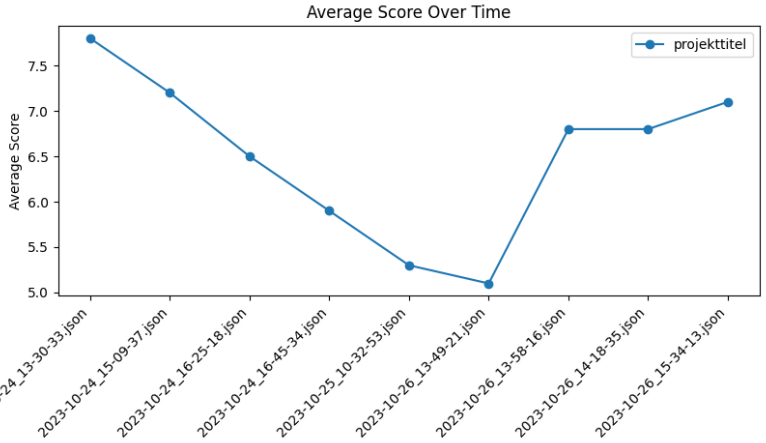

Abbildung 3.2: Visualisierung eines Prompts mit verschiedenen Prompt und Bewertungsmodifikationen über Zeit

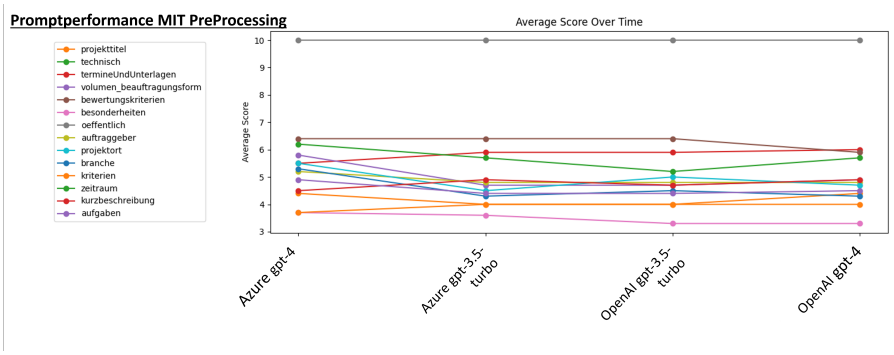

Abbildung 3.3: Visualisierung der Prompts mithilfe unterschiedlicher Modelle. PreProcessing ist aktiviert

3.5.5 Azure Migration

Da die rechtliche Lage bei der Verarbeitung Personenbezogener Daten durch Sprachmodelle wie gpt laut unserer Datenschutzbeauftragten noch sehr unklar ist bis es erste Urteile geben wird, wurde sich darauf geeinigt auf azure openAI zu migrieren da dort europäische Server verwendet werden statt amerikanischer. Ein großer Nachteil von Azure ist aber, dass man für das erstellen der Embeddings für die Vektordatenbank auf 16 Inputs beschränkt ist, während openAI hier keine Einschränkungen vorgibt. Größere Dokumente können leicht auf 1 Input pro Seite kommen, was ein Einbetten von Dokumenten ab 20 Seiten verhindert. Um das Problem zu lösen habe

ich statt die vorgegebene Embedding-Methode zu verwenden eine eigene Methode erstellt, welche die Dokumente bzw. die Inputs auf mehrere Listen mit einer Größe kleiner gleich 16 aufteilt. Anschließend werden die einzelnen Listen über eine REST API zu embeddings umgewandelt und wieder zusammengesetzt. Am ende wird eine Collection mit den Embeddings, den Dokumenten und entsprechenden Metadaten erstellt, welche zentral in einem VectorDataManager zur Verfügung gestellt wird. Gegen diese Collection können anschließend Abfragen wie Similarity-Searches (siehe 3.5.3) durchgeführt werden.

3.5.6 Dokumentation

Sämtlicher von mir geschriebener Quellcode wurde mit Ein- und Mehrzeiligen Kommentaren ergänzt, um das Lesen und Verstehen des Codes zu vereinfachen. Ich habe die gesammelten Erfahrungen in einem Bericht festgehalten und diesen im Repository abgelegt. Zudem wurden Architecture Decision Records, kurz ADR, erstellt und ebenfalls dem Projekt angehängt. Diese ADRs sind dazu da, später Einsicht darüber zu geben warum sich für bzw. gegen ein Framework oder eine Technologie entschieden wurde.

3.5.7 Messestand auf der BUILD23

Zum Abschluss des Projektes wurden wir angefragt, ob wir aufgrund des hohen Interesses an dem Projekt eine Präsentation während des GS-Meetings und der BUILD23 halten möchten. Das erstellen der Präsentation und Vortragen dieser gehörte auch zu meinem Aufgabenbereich.

3.6 Herausforderungen und Lösungsansätze

3.6.1 Optimierung der Prompts

Um die Prompts zu verbessern ist es notwendig, diese an unterschiedlichen Ausschreibungen zu testen und das gelieferte Ergebnis mit der richtigen Antwort zu vergleichen. Da das größte Dokument über 100 Seiten hat und wir keine Musterlösungen haben war es schwierig Aussagen über die Qualität der Prompts zu treffen. Die Lösung war das Einführen einer Metrik welche die Qualität der Prompts misst. Es wurden 10 Dokumente ausgearbeitet und sämtliche wichtige Informationen in OnePager-JSON Dateien gespeichert. Anschließend wurde eine Testarchitektur geschaffen, in welcher man die gewünschten Prompts vollautomatisiert gegen die 10 Testdokumente abfragt und anschließend die Resultate zusammen mit der Musterlösung von ChatGPT auf inhaltliche Übereinstimmung überprüfen und bewerten lässt.

[SIEHE ANHANG XXX - Evaluationsprompt]

Aus den so generierten Bewertungen lassen sich nun Aussagekräftige Kennzahlen generieren. Zur Darstellung wurde eine Visualisierungsklasse geschrieben, welche die Punktzahl der Prompts graphisch darstellt (siehe 3.4) und so schnell erkennbar ist, ob der aktuelle Prompt zu einer Verbesserung oder Verschlechterung der Ergebnisse führt.

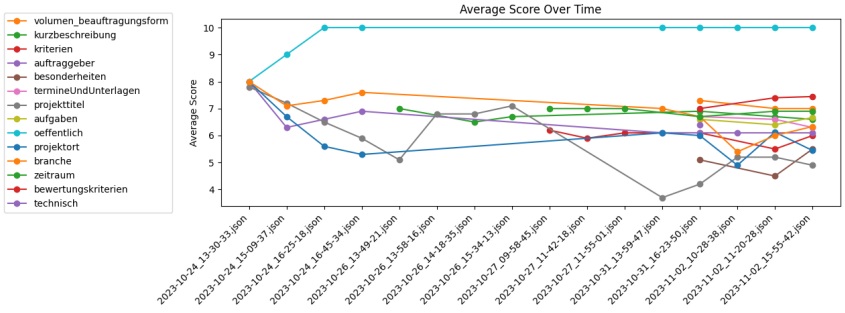

Abbildung 3.4: Visualisierung der durchschnittlichen Promptqualität im Laufe der Zeit

3.6.2 Datenschutz

Das übermitteln und verarbeiten von Personenbezogenen Daten ist in Deutschland nur mit Ausdrücklicher Genehmigung der entsprechenden Person zulässig. Laut unserer Datenschutzbeauftragten ist unklar inwiefern das Übermitteln der Daten aus den Dokumenten über die OpenAI API hierunter einzustufen ist. Offiziell heißt es, dass die Daten, welche über die API geteilt werden, nicht zu Trainingszwecken genutzt werden. Da die Server von OpenAI allerdings in den USA liegen gestalten sich Probleme mit europäischen Datenschutzrecht. Als Lösung wurde die Umstellung auf Azure in Betracht gezogen, da die Server auf europäischen Boden stehen und damit zumindest nach europäischen Recht Datenschutzkonform sind. Zusätzlich anonymisieren wir so gut wie alle Personenbezogenen Daten bevor wir diese an das Sprachmodell übermitteln, indem wir den Kontext vor dem Abschicken gegen eine Liste an Namen prüfen, und so Namen und Mailadressen herausfiltern.

3.7 Ergebnisse und Erfahrungen

3.7.1 Proof of Concept

Aus den gesammelten Ergebnissen und Erfahrungen lässt sich herleiten, dass der PoC definitiv zeigt, dass die Anwendung so möglich ist. Auch wenn nicht immer alle OnePager korrekt generiert wurden, und teils immer noch Halluzinationen vorkommen und wichtige Informationen fehlen so funktioniert die Anwendung im Kern so wie sie es soll. Wichtig für die Zukunft ist es Klarheit beim Umgang mit Personenbezogenen Daten im Zusammenhang mit Sprachmodellen zu schaffen und gegebenenfalls eine geeignete Filterung dieser Daten zu implementieren. Durch neue Modelle, wie dem gegen Ende des Projekts angekündigten gpt-4-turbo, welches eine Kontextlänge von 128K Tokens besitzt, wird man nicht länger gezwungen sein sich auf kleine Textausschnitte zu begrenzen. Man kann ganze Kapitel übergeben und Limitierungen haben mehr wirtschaftliche Gründe als Technische. Auch die Fehleranfälligkeit wird durch einen eingebauten JSON-Parser weiter sinken. Unsere Tests zeigen keine Nennenswerte Unterschiede zwischen den Modellen gpt-3.5-turbo

und gpt-4 was die Qualität der Prompts angeht, daher ist es unwahrscheinlich, dass das neue Modell hier tatsächlich einen Unterschied macht.

3.7.2 Persönlich

Persönlich konnte ich in dem Projekt lernen, wie man mithilfe eines Jira-Boards als Gruppe iterativ Software entwickelt. Ich habe mit Python eine neue Programmiersprache gelernt, welche vielseitig einsetzbar ist und vor allem im Bereich der Künstlichen Intelligenz und Machine Learning viele Möglichkeiten bietet. Zudem konnte ich ein tieferes Verständnis für Prompts und deren Einsatzmöglichkeiten entwickeln. Das entwickeln einer ganze Applikation, welche Sprachmodelle oder andere LLMs integriert ist für mich nun verständlich und durchführbar. Ich konnte auch Erfahrungen im Dokumentieren von Entwicklungsartefakten und Entscheidungen sammeln, welche sich in zukünftigen Projekten weiter vertiefen lassen.

4 Reflexion und Bewertung

4.1 Persönliche Reflexion über das Praxissemester

Persönliche Reflexion über das Praxissemester: Was hast du gelernt? Welche Fähigkeiten konntest du verbessern?

4.2 Bewertung der Zusammenarbeit im Team und im Unternehmen

4.3 Einordnung der erlernten Inhalte und Fähigkeiten im Kontext deines Studiums

5 Fazit und Ausblick

5.1 Zusammenfassung der wichtigsten Erkenntnisse

5.2 Weiterer Werdegang

Wie wird sich das Praxissemester auf deinen weiteren Werdegang auswirken?

5.3 Einordnung der erlernten Inhalte und Fähigkeiten im Kontext deines Studiums

6 Anhang

Eventuell Codebeispiele, Diagramme, Screenshots oder sonstige unterstützende Materialien, die im Hauptteil zu umfangreich wären

6.1 Liste der verwendeten Tools und Technologien

6.2 Literaturverzeichnis

https://www.deeplearning.ai/short-courses/building-systems-with-chatgpt/

7 Eidesstattliche Erklärung

Bestätigung, dass du den Bericht selbstständig verfasst hast (sofern von deiner Hochschule gefordert).

8 Zusammenfassung und Ausblick

Im letzten Kapitel sollte die Arbeit zusammengefasst und ein Fazit gezogen werden. Außerdem sollte beschrieben werden, wie es mit dem Projekt weitergehen kann und welche Punkte vielleicht interessant wären aber im Rahmen der Arbeit nicht bearbeitet werden konnten.

9 Tutorial zu dieser LaTeX-Vorlage

Dieses Kapitel ist spezifisch für die LaTeXVorlage und sollte natürlich in der finalen Abgabe nicht enthalten sein. Dieser Teil der Arbeit kann bei Bedarf durch einen Kommentar einfach ausgeblendet werden (siehe thesis.tex Zeile 125).

9.1 Verwendung dieser Vorlage

Dieses Template ist für die Verwendung mit pdflatex gedacht. Am einfachsten ist es die Vorlage in Overleaf Overleaf c/o Digital Science, n. d. zu öffnen. Overleaf ist eine Onlineanwendung zum Arbeiten mit LaTeX, was den Vorteil hat, dass nichts lokal installiert werden muss und mit jedem Betriebssystem gearbeitet werden kann, das über einen Browser verfügt. Außerdem ist es möglich mit mehreren Personen gemeinsam an einem Projekt zu arbeiten. Die Vorlage kann auch mit einer lokalen Installation verwendet werden. Die Website von Overleaf bietet zudem einige gute Tutorials zum Arbeiten mit LaTeX: https://www.overleaf.com/learn.

Fehler und Warnungen, die beim Compilieren erzeugt werden, sollten direkt behoben werden, da es später schwierig sein kann den eigentlichen Auslöser einer Fehlermeldung zu finden. Manchmal sieht das Dokument trotz Fehlermeldung oder Warnung korrekt aus, der Fehler macht sich dann aber später bemerkbar. Die Meldungen sind leider oft nicht sehr aussagekräftig, weshalb es am einfachsten ist direkt nach dem Auftreten eines Fehlers den Teil der Arbeit anzuschauen, der als letztes geändert wurde.

9.2 Projektstruktur

Der Hauptteil einer Thesis besteht üblicherweise aus mehreren Kapiteln, die verschiedene Aspekte der Arbeit beleuchten. Es ist ratsam für jedes Kapitel ein eigene Tex-Datei anzulegen, damit der Quellcode übersichtlich bleibt. Durch einen numerischen Präfix (z.B.: `20_relatedwork.tex` siehe Abb. 9.1) werden die Quelldateien in der richtigen Reihenfolge in Overleaf angezeigt. Wir verwenden 20 anstatt 2, damit wir nachträglich auch noch 21, 22, etc. einfügen können.

9.2.1 Unterkapitel

Unterkapitel sollten ein abgeschlossenes Thema behandeln. Einzelne Unterkapitel in einem Kapitel sind zu vermeiden, also z. B. in Kapitel 2 das Unterkapitel 2.1, aber kein weiteres Unterkapitel. In diesem Fall ist es besser entweder den Inhalt von 2.1 direkt in Kapitel 2 zu schreiben, oder falls 2 und 2.1 thematisch zu weit voneinander entfernt sind, aus Unterkapitel 2.1 ein eigenes Kapitel 3 zu machen. Dieses Unterkapitel ist ein negativ Beispiel dafür.

9.3 Grafiken

In der Informatik sind die häufigsten Grafiken entweder Diagramme oder Plots. Beide Arten von Grafiken lassen sich gut als Vektorgrafiken erstellen und einbinden. Der Vorteil von Vektor- gegenüber Pixelgrafiken ist, dass beliebig weit in eine Grafik hereingezoomt werden kann, ohne dass sie unscharf wird. Zudem benötigen Vektorgrafiken meistens weniger Speicherplatz.

9.3.1 Vektor- vs Pixelgrafiken

Für die meisten Abbildungen sollte man Vektorgrafiken verwenden, diese können verlustfrei skaliert werden und bieten somit die meisten Freiheiten. Wenn man Grafiken in R erstellt, können die Plots als pdf oder svg-Dateien abgespeichert werden

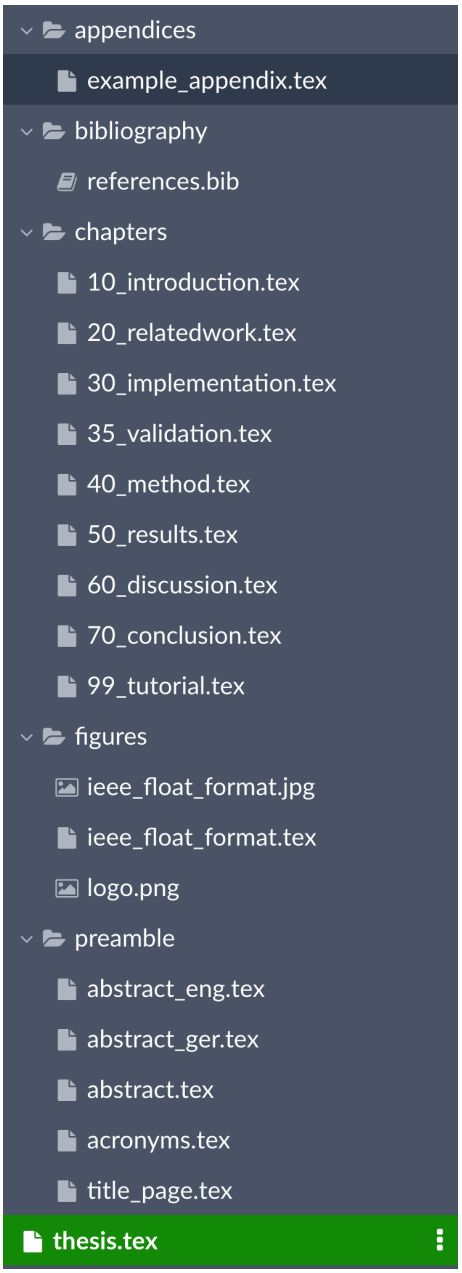

Abbildung 9.1: Ordnerstruktur eines LaTeX-Projektes

und liegen dann ebenfalls als Vektorgrafik vor. Ein weiteres beliebtes Programm zum Erstellen von Vektorgrafiken ist Inkscape Inkscape Project, 2020. Zudem bieten viele Programme die Möglichkeit eine Grafik z. B. als PDF zu exportieren, was in LaTeX als Vektorgrafik eingebunden werden kann. Adobe Indesign wird auch häufig zum Erstellen von Vektorgrafiken verwendet.

Profi-Tipp TikZ. Grafiken können direkt in LaTeX mit dem TikZ Paket Tantau, 2021 erstellt werden. Die Verwendung ist etwas gewöhnungsbedürftig, da Grafiken mit Code beschrieben werden, bietet aber viele Freiheiten. Außerdem werden die so erstellten Grafiken direkt in LaTeX gerendert und verwenden die selbe Schriftart wie im Text und eine konsistente Schriftgröße im gesamten Dokument.

Pixelgrafiken lassen sich nicht immer vermeiden, z. B. wenn eine Foto in die Arbeit eingebunden werden soll. In diesem Fall sollte darauf geachtet werden, dass die Grafik über eine ausreichende Auflösung verfügt. Eine Auflösung von 300 dpi ist ein guter Richtwert, um beim Drucken ein gutes Ergebnis zu erhalten.

Abbildungen 9.2 und 9.3 zeigen beide den Aufbau des IEEE Floating Point Formats. Abbidlung 9.3 ist eine Pixelgrafik, während Abbildung 9.2 mit TikZ erstellt wurde. Der Unterschied wird beim hereinzoomen deutlich.

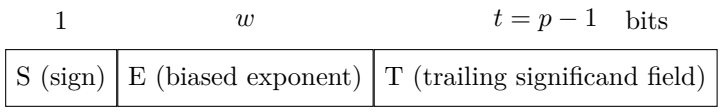

Abbildung 9.2: Aufbau des IEEE Floating Point Formats als Vektorgrafik mit TikZ erzeugt.

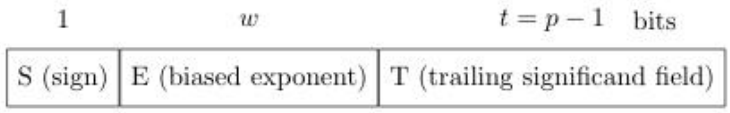

Abbildung 9.3: Aufbau des IEEE Floating Point Formats als Pixelgrafik.

9.4 Tabellen

Tabellen können in LaTeX direkt erstellt werden. Tabelle 9.1 zeigt ein Beispiel dafür. Einfache Tabellen lassen sich schnell erstellen, bei komplizierteren Tabellen ist es manchmal einfacher zusätzliche Pakete zu verwenden. Mit dem Paket `multirow` können z. B. einfacher Tabellen erstellt werden, bei denen einzelne Zeilen oder Spalten zusammengefasst sind.

Unter https://www.tablesgenerator.com/ findet man ein hilfreiches online Werkzeug zum erstellen von Tabellen. Wichtig ist es hier den „Default table style“ zum „Booktabs table style umzustellen“.

Parameter	binary16	binary32	binary64	binary128
k, storage width in bits	16	32	64	128
w, exponent field width in bits	5	8	11	15
t, significand field width in bits	10	23	52	112
emax, maximum exponent e	15	127	1023	16383
bias, $E - e$	15	127	1023	16383

Tabelle 9.1: IEEE 754-2019 Floating Point Formate als Beispiel für das Einbinden einer Tabelle.

9.5 Quellcode

Um Quellcode in die Arbeit einzubinden, können in LaTeX Listings verwendet werden. Es gibt für populäre Sprachen vorgefertigte Umgebungen, welche die Syntax farblich hervorheben. Quellcode sollte eingebunden werden, wenn eine konkrete Implementierung in einer Sprache erläutert wird. Für die Erklärung eines Algorithmus ist es oft übersichtlicher ein Schaubild oder Pseudocode zu verwenden. Es sollten nur kurze Codeabschnitte eingebunden werden, die für den Leser einfach nachvollziehbar sind und nur den für die Erklärung relevanten Code enthalten. Längere Codeabschnitte können im Anhang stehen. Der komplette Code, der für die Arbeit geschrieben wurde, sollte in einem Repository (Gitlab) abgelegt werden.

Listing 9.1 zeigt ein Beispiel für ein Codelisting in der Programmiersprache C.

Algorithmus 9.1 zeigt einen Routing Algorithmus als Pseudocode. Der Code wurde mit dem Paket algorithm2e Fiorio, 2017 erstellt.

```c
#include <stdio.h>
// comments are highlighted in green
void main() {
    // keywords of the language are highlighted in blue
    for (int i = 0; i <= 42; ++i) {
        printf("%d\n", i);
    }
    // strings are highlighted in red
    printf("Hello World!");
}
```

Listing 9.1: Beispiel für ein Codelisting in der Sprache C.

if *destination in west direction* **then**
  go West;
else if *destination in same column* **then**
  if *destination in north direction* **then**
    go North;
  else
    go South;
  end
else if *destination in north east direction* **then**
  go North or go East
else if *destination in south east direction* **then**
  go South or go East
else if *destination in same row and in east direction* **then**
  go East;
else if *at destination* **then**
  done;

Algorithmus 9.1: West First-Routing Algorithm.

9.6 Literatur

Ein Literaturverzeichnis sollte mit dem apa Paket für BibLatex erstellt werden. Dazu wird für jede Quelle ein Eintrag in der Datei `references.bib` angelegt. An der

passenden Stelle im Text können diese Einträge mit dem `\cite{}` Befehl zitiert werden. Für jede Quelle die zitiert wird, legt LaTeX im Literaturverzeichnis einen Eintrag an.

Beschreibungen der Quellen im Bibtex-Format müssen meistens nicht selbst erstellt werden, sondern können direkt bei vielen Verlagen und Bibliotheken direkt generiert werden. Google bietet mit dem „Scholar-Button“ ein Chromium Plugin, mit dem schnell bibtex-Einträge generiert werden können. Bei Google-Scholar generierten Bibtex-Einträgen muss auch eine manuelle Endkontrolle stattfinden, um zu prüfen, ob die Daten in Google korrekt gespeichert waren (z.B. fehlende Autoren, falsche Jahreszahl, etc.).

Es gibt verschiedene Arten wie man mit biblatex zitiert.
`\autocite{valdez2015reducing}` führt zu (Calero Valdez et al., 2015).
`\parencite{valdez2015reducing}` führt zu (Calero Valdez et al., 2015).
`\textcite{valdez2015reducing}` führt zu Calero Valdez et al. (2015).
`\cite{valdez2015reducing}` führt zu Calero Valdez et al., 2015.
`\autocite[siehe auch][S. 13]{valdez2015reducing}` führt zu (siehe auch Calero Valdez et al., 2015, S. 13).

Hier[1] gibt es ein hilfreiches Cheat-Sheet.

Je nach zitierter Dokumentsorte, sieht die Referenz im Literaturverzeichnis anders aus.

- Beispiel für einen Konferenzbeitrag (Nielsen & Molich, 1990)
- Beispiel für einen Journal-Artikel (Hollan et al., 2000)
- Beispiel für ein Buch (Zobel, 2014).
- Beispiel für eine Norm (*Ergonomie der Mensch-System-Interaktion - Teil 220: Prozesse zur Ermöglichung, Durchführung und Bewertung menschzentrierter Gestaltung für interaktive Systeme in Hersteller- und Betreiberorganisationen*, 2020).

[1] https://tug.ctan.org/info/biblatex-cheatsheet/biblatex-cheatsheet.pdf

- Beispiel für einen Weblink[2]

9.7 Abkürzungen

Für jede verwendete Abkürzung kann ein Eintrag in der Datei `acronyms.tex` angelegt werden. Wenn diese Abkürzung im Text zum ersten Mal auftaucht, sollte der Begriff ausgeschrieben werden mit der Abkürzung in Klammern dahinter. Bei weiteren Vorkommen im Text kann dann die eigentliche Abkürzung verwendet werden. In LaTeX gibt es dafür spezielle Befehle. Beispiel für ausgeschriebene Abkürzung (siehe Quelltext des Dokuments für die entsprechenden Befehle).

Erste Verwendung mit Erläuterung: Network Interface Controller (NIC).
Beispiel für das Verwenden der Abkürzung: NIC.

Die verwendeten Abkürzungen werden automatisch im Abkürzungsverzeichnis aufgelistet.

[2]http://imis.uni-luebeck.de („Institut für Multimediale und Interaktive Systeme“, 2023)

9.8 LaTeX Eigenarten

Hier noch ein paar latexspezifische Dinge beim Schreiben.

Non-Breaking Space. Mit dem Tildezeichen ~ bekommt man ein sog. non-breaking Space (nbsp). Das ist hilfreich, wenn man verhindern möchte, dass z.B. die Zeile mit einer Referenz beginnt (Beispiel: Listing 9.1 – Hier wird Listing niemals von der Zahl getrennt werden). Das nbsp wird auch verwendet, um den Abstand nach einem Punkt in einer Abkürzung (1 Einheit) nicht auf die Länge des Abstandes nach dem Satzende zu setzen (1,5 Einheiten). Das fällt nicht immer jedem auf, wenn man es aber einmal sieht, kann man es schwer wieder abschalten.
Beispiel: Dies ist eine Abk. in einem Satz. (korrekt) Die Leertaste wird hier immer gleich lang gehalten.
Beispiel: Dies ist eine Abk. in einem Satz. (falsch) Die Leertaste kann (!) hier vom Setzer verlängert werden, um die Satztrennung deutlicher zu machen.

Binde- und Gedankenstriche. Es gibt drei Stricharten in Latex. Diese haben unterschiedliche Bedeutungen. Der Bindestrich - (engl. hyphen) ist ein einfaches Minus und wird verwendet, um Trennung von Silben anzuzeigen oder Mehrsprachige Komposita zu ermöglichen (z.B. Dashboard-Anzeige). Siehe auch https://www.duden.de/sprachwissen/rechtschreibregeln/bindestrich.

Der deutsche Gedankenstrich (engl. en-dash) – der selten (!) bei Einschüben verwendet wird – sind zwei Minuszeichen und wird mit Leerzeichen (oder nbsp) abgesetzt. Siehe auch https://www.duden.de/sprachwissen/rechtschreibregeln/gedankenstrich. Im Englischen wird der en-dash u.a. dazu benutzt, Zahlenbereiche zu beschreiben: z.B. pages 4–9.

Der englische Gedankenstrich — (engl. em-dash) sind drei Minuszeichen und wird im Englischen für Einschübe benutzt und wird **nicht** mit Leertasten abgesetzt. Beispiel: People—at least most of them—are suprised at how the em-dash is used properly. Im Deutschen kommt dieser Gedankenstrich nicht vor. Verwen-

dung im Englischen siehe auch: https://www.merriam-webster.com/words-at-play/em-dash-en-dash-how-to-use

Anführungszeichen. Wichtig: Anführungszeichen im Deutschen sind anders als im Englischen und Französischen.

Deutsche Anführungszeichen – nach innen gewölbte doppelte Tief- und Hochkommata – werden mit Anführungszeichen und Backtick ("`) geöffnet und mit Anführungszeichen und Apostroph ("') geschlossen. Beispiel: „Latex sollte bei korrekt eingestellter Dokumentsprache aber die korrekten Anführungszeichen wählen.“ sagte er leichtsinnig und irrte sich.

https://www.duden.de/sprachwissen/rechtschreibregeln/anfuehrungszeichen

Englische Anführungszeichen – nach außen gewölbte doppelte Hochkommata – werden mit doppeltem Backtick geöffnet (``) und mit doppeltem Apostroph ('') geschlossen. Beispiel: “This is a correct direct citation in British English”. Der Punkt steht im englischen Zitat ausserhalb der Anführungszeichen (nicht im American English).

Mit dem Command \flqq erhält man das einleitende französische Anführungszeichen («). Mit \frqq bekommt man das Gegenstück (»).

10 Literatur

Calero Valdez, A., Brauner, P., Schaar, A. K., Holzinger, A., & Ziefle, M. (2015). Reducing complexity with simplicity-usability methods for Industry 4.0. *Proceedings 19th Triennial congress of the IEA*, *9*, 14 (siehe S. 33).

Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research. *ACM Trans. Comput.-Hum. Interact.*, *7*(2), 174–196. https://doi.org/10.1145/353485.353487 (siehe S. 33).

Nielsen, J., & Molich, R. (1990). Heuristic Evaluation of User Interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 249–256. https://doi.org/10.1145/97243.97281 (siehe S. 33).

Zobel, J. (2014). *Writing for Computer Science*. Springer London. https://doi.org/10.1007/978-1-4471-6639-9 (siehe S. 33).

Online Quellen

Fiorio, C. (2017, 18. Juli). *algorithm2e Package* [https://ctan.space-pro.be/tex-archive/macros/latex/contrib/algorithm2e/doc/algorithm2e.pdf, accessed on 2022-01-17] (5.3). (Siehe S. 32).

Inkscape Project. (2020, 16. April). *Inkscape* [https://inkscape.org, accessed on 2022-01-13] (0.92.5). (Siehe S. 30).

Institut für Multimediale und Interaktive Systeme [Zugriff am 29.09.2023]. (2023). https://web.archive.org/web/20230929110531/https://www.imis.uni-luebeck.de/de (siehe S. 34).

Overleaf c/o Digital Science. (n. d.). *Overleaf* [https://www.overleaf.com/project, accessed on 2022-01-13]. (Siehe S. 27).

Tantau, T. (2021, 15. Mai). *The TikZ and PGF Packages* [https://github.com/pgf-tikz/pgf, accessed on 2022-01-13] (3.1.9a). (Siehe S. 30).

Normen

Ergonomie der Mensch-System-Interaktion - Teil 220: Prozesse zur Ermöglichung, Durchführung und Bewertung menschzentrierter Gestaltung für interaktive Systeme in Hersteller- und Betreiberorganisationen (Standard). (2020, Juli). International Organization for Standardization. Geneva, CH. (Siehe S. 33).

A Anhang

Der Anhang kann Teile der Arbeit enthalten, die im Hauptteil zu weit führen würden, aber trotzdem für manche Leser interessant sein könnten. Das können z. B. die Ergebnisse weiterer Messungen sein, die im Hauptteil nicht betrachtet werden aber trotzdem durchgeführt wurden. Es ist ebenfalls möglich längere Codeabschnitte anzuhängen. Jedoch sollte der Anhang kein Ersatz für ein Repository sein und nicht einfach den gesamten Code enthalten.

B Abbildungsverzeichnis

C Tabellenverzeichnis

D Listings

Erklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Unterschrift

Lübeck, Tagesdatum