Department of Computer Science

Bachelor Thesis

Analyse der Zeitersparnis der Verwendung eines Templates bei der Erstellung einer Bachelor Thesis

Abgabedatum:

23. Februar 20XX

Eingereicht von:
Maximilian Mustermann
Musterstraße 5a
22880 Wedel
Tel.: 01234 56789
E-Mail: inf123456@stud.fh-wedel.de

Referent:
Prof. Dr. Vorname Nachname
Fachhochschule Wedel
Feldstraße 143
22880 Wedel
Telefon: 04103 - 8048 - XX
E-Mail: xx@fh-wedel.de

Betreut von:
Vorname M. Nachname
Ein FIRMA GmbH
Firmenweg 50
20148 Hamburg
Telefon: 01234 56789
E-Mail: vorname.nachname@firma.com

Inhaltsverzeichnis

Abbildungsverzeichnis **III**

Tabellenverzeichnis **IV**

1 Problemstellung **1**

2 Organisation im Team **2**
2.1 Aufgabenverteilung im Team 2
2.2 Organisation und Zusammenarbeit 2

3 Vorgehen/Grundlagen **3**

4 Architektur **4**
4.1 EDA 4
4.1.1 ANTWORTSNIPPETS 4
4.1.2 DU-Sie 4
4.2 Preprocessing 5
4.2.1 Data-Extraction 5
4.2.2 Paraphrasierer 5
4.3 Labelling 5
4.3.1 Label-Studio 5
4.3.2 Labeling Skript 6
4.4 Modelltraining 6
4.4.1 Allgemeines 6
4.4.2 Sentiment 7
4.5 Schnittstelle/GUI 9
4.5.1 Schnittstelle 9
4.5.2 GUI 9
4.6 Entwicklungsumgebung 10

5 Herausforderung **11**

6 Fazit und Ausblick **12**

Eidesstattliche Erklärung **13**

Abbildungsverzeichnis

Tabellenverzeichnis

1

Problemstellung

Dieses Projekt entstand in Zusammenarbeit mit Allbranded. Der Customer Support von Allbranded bearbeitet aktuell jede Kundenanfrage manuell, dabei ähneln sich die Anliegen häufig. Zudem führt die hohe Auslastung des Teams zu einem ineffizienten Bearbeitungsablauf, bei dem zunächst das Anliegen sowie die Stimmungslage des Kunden manuell erfasst und anschließend ein passendes, vorgefertigtes Antwortsnippet ausgewählt wird. Ziel des Projekts ist die automatische Klassifizierung des Anliegens (Intent) sowie der Stimmungslage (Sentiment) der Mails. Dies ermöglicht eine gezielte Vorschlagsausgabe von Antwortsnippets und sorgt für eine schnelle Übersicht über die jeweilige Stimmung der Kundenanfragen.

2

Organisation im Team

TBC

2.1 Aufgabenverteilung im Team

2.2 Organisation und Zusammenarbeit

3

Vorgehen/Grundlagen

TBC

4

Architektur

4.1 EDA

4.1.1 ANTWORTSNIPPETS

Ein weiterer wichtiger Bestandteil des Projekts war die Analyse von Intention-Kategorien. Ziel war es, aus einer Vielzahl von Antwortsnippets aus Customer-Support-Templates zentrale Kategorien wie „Lieferstatus“ oder „Expressversand“ zu extrahieren. Hierfür wurde eine Verteilungsanalyse durchgeführt, bei der häufige Fragen der Kunden identifiziert und in relevante Themenbereiche eingeordnet wurden. Die Aggregation der Antwortsnippets und die Abschätzung benötigter Intentionen halfen, eine klare Struktur zu schaffen und eine fundierte Basis für die weitere Verarbeitung zu legen.

4.1.2 DU-Sie

Gemäß der mit dem Kunden vereinbarten Kommunikationsform erfolgt die Unterscheidung zwischen Du- und Sie-Ansprache. Dementsprechend existieren für beide Formen separate Antwortsnippets. Für die korrekte Klassifikation der Anrede verwendet das System das Skript du_labeling.py . Dieses Script analysiert den Inhalt der Kundenmail, zählt die Anzahl an Wörtern, die für die Ansprache mit „du“ oder „Sie“ sprechen, und wählt die passende Anrede. Bei gleicher Anzahl an Wörtern, wird “Sie” als Anrede ausgegeben.

4.2 Preprocessing

4.2.1 Data-Extraction

Die E-Mails wurden von Allbranded in acht JSON-Dateien (jeweils 1000 Mails) bereitgestellt. Zunächst wurden diejenigen E-Mails ausgeschlossen, die von Allbranded selbst verfasst wurden. Das Skript mail_extractor.py liest die Mails ein und speichert sie einzeln in separaten JSON-Dateien. Erste Testmails enthielten eine Mailhistorie und zahlreiche HTML-Tags, weshalb anfänglich Funktionen implementiert wurden, um anhand der Grußformel sowie -Tags die relevante E-Mail zu extrahieren und den übrigen Inhalt zu entfernen. Mithilfe der Bibliothek BeautifulSoup erfolgte anschließend die Entfernung von restlichen HTML-Tags.

4.2.2 Paraphrasierer

Parallel dazu wurde ein Paraphrasierer in die bestehende Projektstruktur integriert. Dieser nutzt moderne Methoden wie Backtranslation sowie T5- und MT5-Modelle, um Sätze inhaltlich umzuformulieren. Die Implementierung erwies sich als anspruchsvoll, da die Ergebnisse zunächst wenig sinnvoll waren und durch iterative Anpassungen verbessert werden mussten. Im Verlauf der Arbeit war es notwendig, die zugrunde liegenden NLP-Konzepte zu verstehen, um den Paraphrasierungsprozess zu optimieren. Der Paraphrasierer mit der zugehörigen main.py wurde so konzipiert, sodass ein Skript die bestehenden manuell gelabelten Daten je Kategorien zählt und gemessen wird wie viel Samples bis zu einer definierten Anzahl fehlen. Das Hauptskript nutzt diese und die Methoden des Paraphrasieres, und ermöglicht über eine manuelle Auswahl einer Intent oder Sentimentkategorie paraphrasierte Mails zu erzeugen.

4.3 Labelling

4.3.1 Label-Studio

Label-Studio ist ein Open-Source-Labelling-Tool mit umfangreichen Funktionen, wie vorgefertigten Label-Templates und Workflows für unterschiedliche Datentypen wie

z.B. Texte und Bilder und unterschiedlichen Verwaltungsmöglichkeiten von Labeln und Usern im Projekt. Zu Beginn des Projektes wurde dies für den Eindsatz in Betracht gezogen, jedoch aufgrund unser gerningen Anforderungen an das Tool verworfen und sich für eine eigene Labellösung entschieden, welche im folgenden Abschnitt näher erläutert wird.

4.3.2 Labeling Skript

Für das Labeling der Mails auf dem Server wurde das Skript data_labeling.py entwickelt. Das Skript fordert zunächst eine entsprechende Ordnerauswahl an, um ein paralleles Labeling ohne Überschneidungen zu ermöglichen. Anschließend wird der Inhalt einer ausgewählten Mail in der Konsole zusammen mit den möglichen Klassifikationsoptionen angezeigt. Nach Eingabe einer gültigen Klassifikation des Intents und des Sentiments wird die JSON-Datei in den klassenspezifischen Sentiment- und Intent-Unterordnern gespeichert. Diese Vorgehensweise dient der besseren Übersicht über die Klassenverteilung und dem einfachen Laden der Daten in die Modelle.

4.4 Modelltraining

4.4.1 Allgemeines

Die Entwicklung des Intent- und Sentiment-Modells folgte einem systematischen Vorgehen, das auf bewährten Machine-Learning-Methoden basierte. Beide Modelle wurden zunächst unter Verwendung der Cookie-Cutter-Struktur organisiert, die eine klare Trennung von Daten, Skripten und Modellkonfigurationen ermöglicht. Anschließend kam BERT (Bidirectional Encoder Representations from Transformers) als zentraler Modellkern zum Einsatz. Hierbei wurde Transfer Learning angewandt: Anstatt das Modell von Grund auf neu zu trainieren, griffen wir auf vortrainierte BERT-Gewichte zurück und führten ein Feintuning durch, um die spezifischen Aufgaben im Bereich der Intent- und Sentiment-Erkennung abzubilden. Dieses Vorgehen ermöglichte eine effiziente Anpassung an die domänenspezifischen Daten und reduzierte den Trainingsaufwand

Metriksdatei genauer beschreiben.

4.4.2 Sentiment

BERT

Für die Sentiment-Analyse wurde ein eigenes Modell entwickelt, das auf einer modularen Architektur basiert. Zu den wesentlichen Bestandteilen gehörten unter anderem separate Dateien für das Modell selbst, den Datenladeprozess, Metriken und das Haupttrainingsskript. Durch den Einsatz eines Weighted Random Samplers, Dropout, adaptiver Lernraten und des Layer-Freezings konnten wir die Trainingsqualität erheblich verbessern. Die Hyperparameter-Optimierung, realisiert durch Grid- und Random-Search-Methoden, war eine Herausforderung, insbesondere aufgrund der technischen Gegebenheiten des Servers. Ein frühzeitiges Abbruchkriterium (Early Stopping) basierend auf der Validierungs-Loss wurde eingeführt, um übermäßiges Overfitting zu vermeiden. Letztlich zeigte sich, dass die Qualität der Daten einen entscheidenden Einfluss auf die Ergebnisse hatte. Das Problem der Datenimbalance war ein zentraler Punkt, der schließlich durch Sample-Balancing-Strategien adressiert wurde.

SVM/BERT

Aufgrund der zuvor beschriebenen Thematik der unbalancierten Klassen in den Sentiment und Intent Daten wurde der Ansatz BERT+SVM als Vergleich herangezogen. Die Beiden Ansätze wurden in Kombination verwendet, wobei die druch den BERT-Transformer generierten Embeddings als Input für die Support-Vektor-Machine (SVM) dienen. Es hat sich aber herausgestellt, dass der Ansatz torzt der ergänzung von Methoden wie PCA und Under- und Oversampling und verschiedenen Hyperparameterkombinationen der SVM zu keinen besseren Ergebnissen als das Finetuning eines Vortrainierten BERT-Transformers führt.

Vergleichs Metricen ergänzen???

Wie machen wir das generell mit den Bildern? Ich habe dark mode manchmal weißer hintergrund?? Egal?

Few-Shot Learning

Für die Entwicklung eines Fewshot-Learning-Modells wurde SetFit eingesetzt. Ausgehend von 50 bereinigten Datensamples pro Kategorie wurden verschiedene Szenarien getestet, wobei sich die besten Ergebnisse mit jeweils 10 Samples pro Kategorie ergaben. SetFit verwendet Satz-Embeddings auf Basis von SBERT und bildet Paare von Samples, um Cosinus-Distanzen zu berechnen. Über diese Distanzmaße erlernt das Modell Wahrscheinlichkeiten und klassifiziert so neue Datenpunkte. Allerdings zeigte sich, dass die Ergebnisse nicht an die Qualität des BERT-Ansatzes mit Weighted Random Sampler heranreichten.

Allerdings konnte der Ansatz dafür benutzt werden, die bisher gelabelten Daten präziser zu Labeln. Hierzu wurden zunächst je zehn Mails pro Klasse exemplarisch ausgewertet, wobei jeweils das vorhergesagte Label, das tatsächliche Label und die zugehörige Konfidenz des Modells angezeigt wurden. Aufgrund des Trainings auf sauber gelabelten Daten konnte das Modell abweichende Labels gut erkennen und Diskrepanzen zwischen tatsächlich zugewiesenem Label und Modellvorhersage identifizieren.

Self-Training

Aufgrund der Problematik der ungelabelten Trainingsdaten wurden der Self-Training Ansatz untersucht um den Aufwand für das Labeln zu senken. Self-Training gehört zum Semi-Supervised Learning und bei dem zu Beginn ein Modell auf einem kleinen Datensatz Trainiert wird. Die erlernten Labels soll das Modell dann auf die ungelabelten Daten übertragen. Die Pseudo-Labels des Modells werden nur übernommen und dem Trainingsdatensaz hinzugefügt, wenn sie einen Schwellwert übersteigen. Dieser Ablauf wird solange iterativ durchgeführt, bis ein ausreichend großer Datensatz für das Training eines großen Modells vorhanden ist.

Da zu wenig Daten für diesen Ansatz zu Verfügung standen wurde sich im folgenden auf Few-Shot Learning und Parahprasing fokussiert. Der Self-Training Ansatz benötigt mehr Datensätze um gute Ergebnisse zu erzielen.

Kann man das einfach so schreiben? Oder muss man das belegen? Also mit Metriken

4.5 Schnittstelle/GUI

4.5.1 Schnittstelle

Im Rahmen unseres Projekts wurde die Schnittstelle für das Machine-Learning-Modell als REST-API implementiert. Connexion Flask übernahm dabei das Path-Handling sowie die Verarbeitung von Requests und Responses, was eine nahtlose Einbindung des Modells in unsere Anwendung ermöglichte. Diese Vorgabe des Auftraggebers brachte allerdings einige Herausforderungen mit sich, insbesondere bei der Versionierung, der korrekten Angabe von Datentypen und der Verknüpfung von App-Funktionen mit der API. Ein grundlegendes Verständnis von REST-API-Konzepten und den spezifischen Mechanismen von Connexion Flask war entscheidend, um diese Probleme zu lösen.

4.5.2 GUI

Die Containerisierung der Schnittstelle wurde mittels Docker umgesetzt. Dafür wurde ein Dockerfile auf Basis eines miniconda Docke-Image erstellt, wobei das entsprechende conda Environment vom Server exportiert und im Container genutzt wird. Die Umsetzung erfolgte mithilfe von Docker-Compose, welches zwei eigenständige Services bereitstellt:

Der erste Service namens mailanalysis ist über die Anwendung app.py realisiert und erlaubt das flexible Austauschen der verwendeten Modelle, beispielsweise um nachtrainierte Sentiment- oder Intent-Modelle einzubinden. Der zweite Service gradioapp stellt eine Weboberfläche bereit, die über die Anwendung gradio app.py umgesetzt ist. Diese Webapp ist unter der Adresse http://138.199.202.254:8000 erreichbar. Bei Nutzeranfragen greift die Webapp intern auf den Service mailanalysis zu, welcher die Verarbeitung der Anfragen mit den hinterlegten Modellen übernimmt und anschließend die Ergebnisse auf der Weboberfläche darstellt.

Die Webapp für unseren Prototypen wurde mit der Python Library Gradio implementiert und besteht aus drei Hauptseiten: einer Instruktionsseite, einer Klassifikationsseite und einer Seite für die Human-Feedback-Loop. Die Instruktionsseite erklärt die Funktionsweise und Bedienung der Webapp. Auf der Klassifikationsseite kann der Anwender eine zu klassifizierende E-Mail eingeben, welche anschließend durch den Klassifizieren-Button eine Anfrage an die Schnittstelle zur Intent- und Sentiment-Klassifikation sendet.

Die Ergebnisse der Klassifikation durch die Schnittstelle werden dem Nutzer angezeigt. Zusätzlich zu den Klassifizierten Intent und Sentiment werden noch die zum Intent passenden vorgefertigten Antworten aus einem von Allbranded zur Verfügung gestellten Dokument angezeigt, welche sich durch den Kopieren-Button in die Zwischenablage kopieren lassen.

Die Seite Human Feedback Loop bietet die Möglichkeit, bei Falschklassifikationen Korrekturen manuell vorzunehmen. Die Anpassungen sind dabei auf einzelne Intents und Sentiments beschränkt, um fehlerhafte oder unklare Nutzereingaben auszuschließen, da das Modell ausschließlich auf Single-Intent-Daten trainiert wurde. Korrigierte Klassifikationen werden noch einmal zur Kontrolle ausgegeben, während die entsprechenden E-Mails separat in der gleichen Struktur wie die Trainingsdaten gespeichert werden. Dies ermöglicht ein einfaches Nachtrainieren des Modells anhand der während des Betriebs gesammelten, korrigierten Daten.

4.6 Entwicklungsumgebung

Für die Umsetzung des Projektes wurde uns von Allbranded ein Server zur Verfügung gestellt. Der Server hat einen 8 Kern Prozessor und 16 GB RAM sowie eine Festplatte mit einer Kapazität von 150 GB. Auf den Server haben wir über eine SSH-Verbindung zugegriffen.

Die Entwicklungsarbeiten haben parallel auf dem Server sowie Lokal stattgefunden wobei das Modelltraining aus Datenschutzrechtlichengründen ausschließlich auf dem druchgeführt wurde. Auf dem Server haben wir die Skripte für das Preprocessing und das Modelltraining innerhalb einer Conda-Environment entwickelt. Sie Skripte die Lokal entwickelt wurden haben wir über ein Git-Repository mit dem Server synchronisiert.

Für das Betreiben der Schnittstelle und der Weboberfläche ist auf dem Server auch Docker installiert.

5

Herausforderung

Im Verlauf unseres Projektes sind verschiedene Herausforderungen aufgetreten, die sich auf unser Projektergebnis ausgewirkt haben. Ein zentrales Problem stellte dabei die späte Bereitstellung sowohl der benötigten Server-Infrastruktur als auch der Trainingsdaten durch unser Partnerunternehmen dar. Aufgrund datenschutzrechtlicher Bedenken, die insbesondere darauf zurückzuführen waren, dass es sich bei den Trainingsdaten um echte Kundenmails handelte, kam es zu Verzögerungen. Dies führte dazu, dass einige Folgeprobleme erst im späten Projektverlauf auftraten, was wiederum die Möglichkeiten einer zeitnahen und effektiven Problemlösung erheblich einschränkte.

Wie schon in vorherigen Kapiteln erwähnt mussten wir die Trainingsdaten selbst labeln, welches einen erheblichen Zusatzaufwand darstellte. Ebenfalls hatte das fehlende Verständnis über die veschiedenen Prozesse bei Allbranded einen Einfluss auf die Qualität der Labels. Eine Intensivere Abstimmung mit unserem Partnerunternehmen hätte die Qualität der Labels erhöhen können, diese war aber leider nur zum Teil möglich, da unsere Ansprechpartner selber sehr stark ausgelastet waren.

Darüber hinaus stellte die unbalancierte Verteilung der Kategorien innerhalb des Datensatzes eine Schwierigkeit dar, welche die Aussagekraft und Performance der Modelle negativ beeinflusste. Verstärkt wurde dieses Problem zusätzlich durch die generell geringe Menge an verfügbaren Daten verstärkt.

Trotz der genannten Herausforderungen erzielte das Projekt gute Ergebnisse, welche durch die im folgenden Kapitel dargestellten Punkte weiter verbessert werden können.

Partnerunternehmen oder Allbranded

6

Fazit und Ausblick

Der Einsatz der Modelle über die entwickelte Schnittstelle erweist sich als praktikabel. Das Sentiment wird mit dem trainierten Modell zuverlässig erkannt, während Intents, je nach Klassencharakteristik, unterschiedlich gut identifiziert werden. Insbesondere häufig vorkommende Anfragetypen können effektiv klassifiziert werden, wodurch eine Zeitersparnis im Kundensupport erzielt wird. Für zukünftige Verbesserungen ist eine Erweiterung des gelabelten Datensatzes erforderlich, um ein feineres Tuning, insbesondere bei unterrepräsentierten Klassen, zu ermöglichen. Hierbei kann die implementierte Human-Feedback-Loop unterstützend wirken. Zudem sollte das Few-Shot-Learning-Modell erneut auf den vorgefilterten, sauberen Daten trainiert werden, um deren Qualität weiter zu optimieren. Des Weiteren empfiehlt sich eine Ergänzung der Klassen mit ergänzenden Antwortsnippets, um potenziell fehlende Antwortmöglichkeiten zu adressieren. In Zukunft muss die bereitgestellte Schnittstelle mit dem ERP-System von Allbranded verbunden werden, um die Vorhersagen in der Praxis anwenden zu können.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Ort, Datum Maximilian Mustermann