fhwedel
UNIVERSITY OF APPLIED SCIENCES

Projektbericht

Analyse der Kommunikationshistorie mittels Klassifikation und Auswahl geeigneter Antwortsnippets

Abgabedatum:

26. März 2025

Eingereicht von:

Niclas Rust

Marc-Louis Dederichs

Yue Cao

Luis Schwindt

Betreuer:

Prof. Dr. Hendrik Annuth
und
Marco Pawlowski
Fachhochschule Wedel
Feldstraße 143
22880 Wedel

Inhaltsverzeichnis

1

Problemstellung

Dieses Projekt entstand in Zusammenarbeit mit Allbranded. Dies ist ein Unternehmen, welches personalisierte Werbeartikel verkauft. Der Customer Support von Allbranded bearbeitet aktuell jede Kundenanfrage manuell, dabei ähneln sich die Anliegen häufig. Zudem ist das Kundensupport Team stark ausgelastet. Die Anliegen der Kunden werden meist mit Hilfe von standardisierten Antwort Snippets beantwortet. Ziel des Projekts ist die automatische Klassifizierung des Anliegens (Intent) sowie der Stimmungslage (Sentiment) der Mails. Dies ermöglicht eine gezielte Vorschlagsausgabe von Antwort Snippets und sorgt für eine schnelle Übersicht über die jeweilige Stimmung der Kundenanfragen.

2

Organisation im Team

Zu Beginn des Projekts haben wir unser Vorgehen auf Basis einer Literaturrecherche thematisch strukturiert und die anstehenden Aufgaben in Arbeitspakete unterteilt. Jedes Arbeitspaket wurde einem spezifischen Themenbereich zugeordnet, beispielsweise Datenanalyse- und aufbereitung, Labelling und Modelltraining.

Die Aufgaben innerhalb der Arbeitspakete wurden anschließend einzelnen Teammitgliedern zugewiesen, welche bis zum jeweils nächsten Termin bearbeitet wurden.

In unseren Internen Terminen hat zu zu Beginn jeder seinen aktuellen Stand seiner Arbeit vorgestellt. Im Anschluss wurden ggf. aufgetretene Probleme besprochen. Anschließend wurde anhand der aufgetretenen Probleme und des aktuellen Fortschritts die nächsten Aufgaben für das jeweilige Arbeitspaket erstellt und zugewiesen. So konnten wir flexibel auf die gegebenen Umstände reagieren.

Die Zuständigkeiten für die jeweiligen Arbeitspakete können der folgenden Abbildung 2.1 entnommen werden.

Arbeitspakete	Marc-Louis Dederichs	Luis Schwindt	Yue Cao	Niclas Rust
Analyse der Intentkategorien	3	1	1	1
API + Connexion/Flask	14	4		
Bucket Count Skript	2			
Datenbereinigung (zweites Testset)	2			3
Datenvorbereitung		2		1
Docker		10		1
Du-Erkennung Skript				3
E-Mail Extraktion	2			4
externes Meeting	5	5	5	5
Few Shot Learning	4			6
Gradio App	1	14		
Hyperparameter Skript	4			
Intentmodell		10	2	10
Intentmodell (LoRA)			2	4
internes Meeting	14	14	14	14
Konzeption	2	2	2	4
Labeling	22	24	28	20
Labeling Skript				5
Labeling Tool		5		
Paraphrasing	6		22	
Präsentation (Vorbereitung etc.)	16	11	10	18
Programmierguidelines				2
Projektbericht	4	8	4	4
Recherche	10	10	8	11
Sentimentmodell	16	4	2	4
Sentimentmodell (2 Klassen)				5
Sentimentmodell (BERT + SVM)		4	4	
Sentimentmodell (klassische ML-Ansätze)			18	
Sentimentmodell (Sample Balancing)	4		2	2
Servertest	1	3	1	1
Tensorboard	1	3	1	2
Stunden Gesamt	**133**	**134**	**126**	**130**

Abbildung 2.1: Zeitentabelle

3

Vorgehen

Zur Erstellung unseres Modells für die Sentiment- und Intent-Klassifikation haben wir uns an einem etablierten Workflow aus dem Bereich der Textklassifikation orientiert. Der Workflow beschreibt die Schritte, die zur Entwicklung eines Textklassifikationsmodells durchlaufen werden können. In dem Workflow werden die folgenden sechs Schritte beschrieben: Datenbeschaffung, Datenanalyse und -labeling, Feature-Konstruktion und -gewichtung, Feature-Selektion und -Projektion, Modelltraining sowie Evaluation der Lösung. [MP18]

Im folgenden wird nur kurz auf die Schritte des Workflows eingegangen, die für unseren Anwendungsfall relevant waren. Datenbeschaffung die Daten wurden uns in Kundenmails in anonymisierter Form von Allbranded zur Verfügung gestellt. Im darauffolgenden Datenanalyse und-labelling Schritt wurden Labels für die Kundenmails erarbeitet und auf die Mails übertragen. Nach dem Labelling wurde direkt mit dem fünften Schritt dem Modelltraining begonnen. Zum abschluss eines Modelltrainings wurden diese dann anhand gäniger Metriken evaluiert und mit den anderen Modellen verglichen. [MP18]

Darüber hinaus wurde geprüft, mit welchen Tools und Frameworks sich die einzelnen Schritte umsetzen lassen

4

Architektur

4.1 EDA

4.1.1 Antwortsnippets

Ein Bestandteil des Projekts war die Analyse von Intention-Kategorien, bzw. der E-Mail-Anfragen die der Customer Support die zu bearbeiten hatte. Ziel war es, aus einer Vielzahl von Antwortvorschlägen (oder Antwortsnippets) aus dem Customer-Support-Templates zentrale Kategorien wie „Lieferstatus“ oder „Expressversand“ zu extrahieren. Parallel standen uns ca. 110 Beispiel-Mails zur Verfügung, aus der wir eine Verteilung der Anfragen abgeleitet haben. Die Verteilungsanalyse wurde benutzt um häufige Fragen der Kunden zu identifizieren und in relevante Themenbereiche einzuteilen, um die Antwortvorschläge zu aggregieren. 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 Erkennung der Anrede

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

Unter der Annahme, dass das Labeln der Daten und die daraus resultierende Menge an Beispielen je Kategorie nicht ausreichend sein wird, wurde ein Paraphrasierer in die bestehende Projektstruktur integriert, der auf einem fundierten Verständnis der zugrunde liegenden NLP-Konzepte basiert. Der Paraphrasierer kombiniert drei Ansätze zur inhaltlichen Umformulierung von Sätzen: Er nutzt einerseits eine Fill-Mask-Substitution mittels des vortrainierten Modells „bert-base-german-cased“, andererseits eine T5-basierte Paraphrasierung mit dem Modell „seduerr/t5_base_paws_ger“ und dem „t5-base“-Tokenizer sowie eine Pivot-Übersetzung unter Verwendung lokaler MarianMT-Modelle, die als Backtranslation dienen. Die Implementierung erwies sich als anspruchsvoll, da die initialen Ergebnisse wenig überzeugend waren und erst durch iterative Anpassungen optimiert werden konnten. Das zugehörige Hauptskript zählt die existierenden manuell gelabelten Daten je Kategorie, ermittelt den Bedarf an zusätzlichen Samples und ermöglicht über eine manuelle Auswahl einer Intent- oder Sentimentkategorie die Erzeugung paraphrasierter Mails, die anschließend als neue Trainingsbeispiele in das System integriert werden.

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 Einsatz in Betracht gezogen, jedoch aufgrund unser gerningen Anforderungen an das Tool verworfen und sich für eine eigene Labelling Lösung 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.

4.3.3 Sample Balancing

Ein zentrales Problem im Projekt war das Sample-Balancing. Viele Kategorien waren stark unterrepräsentiert, wodurch das Training der Modelle erschwert wurde. In einem ersten Schritt wurde der Dataloader ohne jegliche Korrektur getestet, was zu unausgewogenen Ergebnissen führte. Danach wurden verschiedene Techniken ausprobiert: Undersampling, bei dem die Kategorie mit der niedrigsten Verfügbarkeit als Grenze gesetzt wurde, und Oversampling mittels eines Weighted Random Samplers, der für ausgeglichene Batches im Training sorgte. Diese Methode erzielte letztlich die besten Ergebnisse und bildete die Grundlage für die weitere Optimierung des Sentiment-Modells. Zusätzlich wurde ein „augmented“-Modus eingeführt, bei dem Kategorien

mit wenigen Samples durch generierte Daten ergänzt wurden, unter Verwendung des zuvor entwickelten Paraphrasierers.

4.3.4 Self-Training

Aufgrund der Problematik der ungelabelten Trainingsdaten wurde der Self-Training Ansatz untersucht um den Aufwand für das Labeln zu senken. Self-Training gehört zum Semi-Supervised Learning 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.

Aufgrund der geringen Anzahl an Trainingsdaten und der unbalancierten Klassen wurden keine verwendbaren Labels erzeugt, weshalb sich auf Few-Shot Learning und Parahprasing fokussiert wurde.

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-Modelle von huggingface zurück und führten Fine-Tunings durch, um die Modelle für die spezifischen Aufgaben der Intent- und Sentimentklassifikation zu verwenden.

Um die trainierten Modelle zu testen, wurden mehrere Metrics-Dateien erstellt, welche die zuvor gespeicherten Modelle laden und die Performance auf den Testdaten messen. Zur Messung der Performance wurde der F1-Score, zusammengesetzt aus Precision und Recall herangezogen. Der Score wurde dabei für alle Testdaten ermittelt, und zusätzlich für die einzelnen Kategorien, was uns weitere interessante Einblicke in die Performance unserer Modelle lieferte.

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.

Logistische Regression

Die logistische Regression benötigt numerische Features, weshalb zur Umwandlung der Textdaten die Methode Term Frequency-Inverse Document Frequency (TF-IDF) verwendet wurde. Diese Methode berücksichtigt sowohl die Häufigkeit eines Wortes als auch dessen relative Seltenheit. Für jedes Sentiment wird eine lineare Kombination der numerischen Feature-Werte mit spezifischer Gewichtung berechnet. Anschließend werden die berechneten Werte mittels einer Softmax-Funktion in Konfidenzwerte umgerechnet, welche für die Klassenvorhersage genutzt werden. Das Modell ermöglicht ein schnelles Training und eine schnelle Klassifizierung, zeigte jedoch im Vergleich zu Ansätzen mit Transformern schlechtere Ergebnisse.

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. Aufgrund der guten Ergebnisse wurde das Few Shot Learning Modell genutzt, um die unsauber gelabelten E-Mails zu erkennen. Es wurden Dateien aussortiert, welche ein abweichendes Label oder eine Konfidenz von unter 70 % haben. Auf dem neu entstandenen Datensatz wurde das Sentiment Modell erneut trainiert und getestet. Es konnte eine signifikante Verbesserung erzielt werden. Es muss jedoch bedacht werden, dass Mails die uneindeutig formuliert wurden, auch im Vorhinein aussortiert wurden. Dadurch wird die Erkennung auf den Testdaten vereinfacht.

Modellergebnisse

In der folgenden Tabelle sind die besten Modelle der oben aufgeführten Ansätze für die Sentimentklassifikation dargestellt.

Modell	Accuracy	Precision	Recall	F1-Score	Anmerkungen
BERT (Basic)	0.8223	0.8824	0.8223	0.8404	Ungesäuberte Daten
BERT Few-Shot	0.6641	0.8396	0.6641	0.7199	Ungesäuberte Daten
BERT Sentiment	0.8386	0.9468	0.8336	0.8820	Ungesäuberte Daten
BERT + SVM	0.86	0.822	0.859	0.826	Ungesäuberte Daten
Logistische Regression	0.7962	0.8243	0.7962	0.8084	Ungesäuberte Daten
BERT Few-Shot	0.9123	0.9354	0.9123	0.9173	Gesäuberte Daten
BERT Few-Shot	0.9373	0.9476	0.9373	0.9401	Gesäuberte Daten

Tabelle 4.1: Übersicht Modellergebnisse Sentiment

4.4.3 Intent

Für die Intent-Analyse wurde derselbe Ansatz wie bei der Sentiment-Analyse gewählt. Grundlegende Techniken wie Weighted Random Sampling, Dropout, adaptive Lernratenanpassung und Layer-Freezing kamen zum Einsatz, um das Modell zu optimieren. Zwei unterschiedliche Ausgangsmodelle wurden untersucht: Zum einen das BERT-German-Uncased-Modell, das speziell für die Klassifikation deutscher Dokumente konzipiert ist, und zum anderen DistilBERT, ein kleineres Modell, das getestet wurde, um zu schauen wie die Performance bei einem ressourcenschonende Modell ist. Die besten Ergebnisse erzielte das BERT-German-Uncased-Modell.

Als weitere Fine Tuning Technik wurde LoRA (Low-Rank Adaptation) eingesetzt, das eine ressourcenschonende Anpassung von vortrainierten Modellen ermöglicht. Die Hauptgewichtungen eines vortrainierten Modells wird dabei eingefroren und stattdessen trainierbare, nieder-rangige Matrizen eingefügt. Dadurch werden nur wenige Parameter angepasst, was einen ressourcenschonenden und effizientes Fine Tuning ermöglicht. Der Einsatz von LoRA führte jedoch nicht zu einer signifikanten Leistungsverbesserung. Aufgrund von Zeitbeschränkungen wurde der Großteil der Evaluierungen zunächst im Bereich der Sentiment-Erkennung durchgeführt, sodass der Fokus bei der Intent-Erkennung etwas zurückgestellt werden musste.

Die Erkennungsgenauigkeit des Modells variiert stark je nach Klasse. Klassen mit häufig verwendeten Begriffen und einer großen Anzahl von Beispielen, wie etwa die Freigabe einer Druckskizze, wurden deutlich präziser klassifiziert als Intents mit sehr heterogenen Beispielen. Ein wesentlicher Problempunkt stellt zudem eine ungenaue Labelvergabe dar, welche durch fehlenden Kontext und unterschiedliche Interpretationen der Klassen entsteht. Des Weiteren können Mails auch mehrere Intents thematisieren. Es kann jedoch nur ein Label vergeben werden. Zukünftig könnte ein Few-Shot-Learning-Ansatz zur Datenbereinigung, ähnlich wie bei der Sentiment-Analyse, zur Verbesserung der Qualität der Labels beitragen.

Modellergebnisse

In der folgenden Tabelle sind die beiden besten Modelle für die Intentklassifikation aufgeführt.

Modell	Accuracy	Precision	Recall	F1-Score
BERT (bert-base-uncased)	0.5920	0.6170	0.5920	0.5527
distilbert-base-uncased	0.5946	0.6044	0.5946	0.5853

Tabelle 4.2: Übersicht Modellergebnisse Intent

4.5 Schnittstelle und Webapp

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.

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 mail_analysis 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 gradio_app stellt eine Weboberfläche bereit, die über die Anwendung gradio_app.py umgesetzt ist. Diese Webapp ist auf dem Server unter der Adresse http://138.199.202.254:8000 erreichbar. Bei Nutzeranfragen greift die Webapp intern auf den Service mail_analysis zu, welcher die Verarbeitung der Anfragen mit den hinterlegten Modellen übernimmt und anschließend die Ergebnisse auf der Weboberfläche darstellt. Auf die genauen Funktionen der Webapp wird im folgenden Abschnitt geanuer eingegangen.

4.5.2 Webapp

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 wesentliches Problem war die verspätete Bereitstellung der Server-Infrastruktur und der Trainingsdaten durch Allbranded. Die Verzögerung entstand vor allem aufgrund datenschutzrechtlicher Bedenken, da es sich bei den Daten um echte Kundenmails handelte. Dies führte dazu, dasseinige 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 Allbranded 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.

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. Das Klassifizieren des Sentiments ermöglicht die Erhebung einer Kundenzufriedenheitsmetrik bei Allbranded. Für zukünftige Verbesserungen ist eine Erweiterung des gelabelten Datensatzes, insbesondere bei unterrepräsentierten Klassen, erforderlich. 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. Das bereitgestellte User Interface kann genutzt werden, den Mitarbeitern im Kundensupport die Nutzbarkeit nahe zu bringen.

Literatur

[MP18] Marcin Mirończuk und Jaroslaw Protasiewicz. „A Recent Overview of the State-of-the-Art Elements of Text Classification“. In: *Expert Systems with Applications* 106 (März 2018). DOI: 10.1016/j.eswa.2018.03.058.