**Projektdokumentation: Automatische Klassifizierung von Support-Tickets**

**Ein kleines KI-Modell vor einer großen Herausforderung**

**Von:** Robin Dar  
**Datum:** September 2025

**1. Die Mission: Kann eine kleine KI den Support-Alltag meistern?**

Ein bekanntes Szenario: Der Posteingang für Support-Anfragen quillt über. Von "Passwort vergessen" über "Meine Rechnung stimmt nicht" bis hin zu emotionalen Kunden-Feedbacks ist alles dabei. Die manuelle Sortierung dieser Tickets ist nicht nur repetitiv, sondern auch ein Zeitfresser und eine potenzielle Fehlerquelle.

Die Kernidee dieses Projekts war es, dieser Herausforderung mit den Mitteln kleiner, ressourcenschonender KI-Modelle zu begegnen. Das Ziel war klar definiert: Ein lauffähiges Python-Programm zu entwickeln, das eine Support-Anfrage analysiert und ihr automatisch eine passende Kategorie zuordnet – und das alles auf einem Standard-Rechner, ohne die Notwendigkeit teurer Server-Infrastruktur.

**2. Die Werkzeuge: Meine Modellauswahl und technische Basis**

Die Wahl fiel auf das Zero-Shot-Modell **MoritzLaurer/mDeBERTa-v3-base-mnli-xnli**. Der "Zero-Shot"-Ansatz ist besonders attraktiv, da das Modell keine expliziten Trainingsdaten benötigt. Man gibt ihm eine Anfrage und eine Liste von Kategorien, und es ermittelt die wahrscheinlichste Übereinstimmung. Das machte es zum idealen Kandidaten für einen schnellen und flexiblen Prototypen.

Technisch stützt sich das Projekt auf zwei entscheidende Libraries:

* **transformers:** Diese Bibliothek von Hugging Face ist das Herzstück der Implementierung. Sie agiert als universelle Schnittstelle zu unzähligen KI-Modellen und vereinfacht den komplexen Prozess des Ladens und Anwendens eines Modells auf wenige, verständliche Codezeilen.
* **torch:** Als zugrundeliegendes Deep-Learning-Framework ist torch der "Motor" des Systems. Es führt im Hintergrund die anspruchsvollen Berechnungen aus, die für den Betrieb der Neuronalen Netze notwendig sind.

**3. Die Reise zur Lösung: Ein iterativer Prozess voller Erkenntnisse**

Der Weg zum finalen Code war ein klassischer Entwicklungsprozess aus Hypothesen, Tests und der Analyse von Erfolgen und Fehlschlägen.

**Phase 1: Der "Detailreichtum"-Ansatz**  
Meine erste Annahme war, dass mehr Informationen zu besseren Ergebnissen führen. Daher versah ich die Kategorien mit extrem langen, detaillierten Beschreibungen und Beispielen.

* **Realität:** Ein Fehlschlag. Das Modell war von der schieren Textmenge in den Labels überfordert, was zu zufälligen und unbrauchbaren Ergebnissen führte.
* **Erkenntnis #1:** Kleine Modelle benötigen klare und prägnante Anweisungen, keine überladenen Prompts.

**Phase 2: Der "Minimalismus"-Ansatz**  
Als Konsequenz wurden die Labels radikal auf kurze, eindeutige Titel reduziert.

* **Realität:** Eine deutliche Verbesserung. Klare Anfragen wurden nun wesentlich zuverlässiger klassifiziert. Bei mehrdeutigen Formulierungen zeigte das Modell jedoch weiterhin Schwächen.

**Phase 3: Der Pipeline-Ansatz – Eine Lektion in Komplexität**  
Die Idee, die Schwächen durch eine Kette von Spezialisten-Modellen (Emotion -> Absicht -> Thema) auszugleichen, klang vielversprechend.

* **Realität:** Dieser Ansatz scheiterte in der Praxis. Das allgemeine Sentiment-Modell interpretierte den spezifischen Kontext von Support-Anfragen oft falsch (z.B. "Passwort vergessen" als NEGATIVE). Da die erste Stufe der Pipeline bereits fehlerhafte Daten lieferte, war das Endergebnis unzuverlässig.
* **Erkenntnis #2:** Eine komplexe Architektur ist nur dann nützlich, wenn jede einzelne Komponente ihre Aufgabe zuverlässig erfüllt.

**4. Die finale Implementierung: Pragmatisch und ehrlich**

Nach der Evaluierung der verschiedenen Ansätze führte der Weg zurück zur einfachsten und gleichzeitig robustesten Lösung: **Ein einzelnes Zero-Shot-Modell, kombiniert mit klaren, prägnanten Labels.**

Der finale Code implementiert diesen direkten Ansatz. Er wartet auf eine Nutzeranfrage, klassifiziert sie und – ein entscheidendes Feature – kommuniziert seine eigene Unsicherheit transparent. Anstatt eine potenziell falsche Antwort als Fakt zu präsentieren, nutzt das Programm einen Schwellenwert und bietet bei niedriger Konfidenz alternative Vorschläge an.

Eine vollständige Anleitung zur Inbetriebnahme befindet sich in der README.md-Datei.

**5. Mein Fazit: Ein ehrliches Urteil über kleine Modelle in der Praxis**

Dieses Projekt war ein wertvoller Praxistest, der vor allem die realen Grenzen kleiner KI-Modelle aufzeigte.

* **Kleine Modelle sind hochspezialisierte Werkzeuge, keine Mini-Alleskönner.** Im Gegensatz zu großen, generativen Modellen wie GPT-4 fehlt ihnen das tiefe, intuitive Sprachverständnis. Man muss lernen, die Aufgabe an die Fähigkeiten des Modells anzupassen, nicht umgekehrt.
* **Der Kontext ist der entscheidende Faktor.** Die größte Hürde war, dass die allgemein trainierten Modelle den spezifischen Kontext von Support-Anfragen oft falsch interpretierten. Diese semantische Lücke war die Hauptursache für die meisten Fehlklassifizierungen.
* **Ein "begrenzter Erfolg" ist ein wertvolles Ergebnis.** Das finale Programm ist kein 100%ig treffsicherer Klassifikator. Und genau das ist die wichtigste Erkenntnis dieses Projekts. Der iterative Prozess hat unmissverständlich bewiesen, dass die hier eingesetzten, allgemein vortrainierten Modelle für diese nuancierte Aufgabe an ihre Leistungsgrenze stoßen.

**Zusammenfassend lässt sich sagen:** Das Projekt hat erfolgreich einen funktionierenden Prototypen hervorgebracht. Es hat jedoch auch die klare Diskrepanz zwischen der Leistungsfähigkeit großer Sprachmodelle und den realen Einschränkungen ihrer kleinen Gegenstücke demonstriert. Das finale Programm funktioniert, aber nur bedingt.

Für eine wirklich produktiv einsetzbare Lösung gibt es nur einen logischen nächsten Schritt: das **Fine-Tuning**. Man müsste ein Basis-Modell (wie das empfohlene DistilBERT) nehmen und es auf einem Datensatz von echten, anonymisierten Support-Tickets nachtrainieren. Nur so kann die KI den spezifischen Kontext und die Sprache der Kunden lernen, die für eine hochpräzise Klassifizierung unerlässlich sind.

Das Projekt endet somit nicht mit einer perfekten Lösung, sondern mit einer professionellen und fundierten Handlungsempfehlung, die auf praktischen Experimenten und einer ehrlichen Analyse der Ergebnisse basiert.