**Projektdokumentation: Automatische Ticket-Sortierung**

**Kann eine kleine KI den Support-Stress killen? Ein Praxis-Test.**

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

**1. Die Mission: Raus aus dem Ticket-Chaos!**

Jeder kennt’s: Das E-Mail-Postfach explodiert. "Passwort vergessen", "Rechnung falsch", "ALLES IST SCHLECHT!!!" – und alles landet auf einem Haufen. Das manuelle Sortieren nervt, kostet Zeit und ist fehleranfällig.

Die Idee für dieses Projekt war simpel: Kann ein kleines, ressourcenschonendes KI-Modell diesen Job für uns erledigen? Ziel war es, ein einfaches Python-Programm zu bauen, das auf einem normalen PC läuft und auf Knopfdruck sagt: "Ah, das ist eine Rechnungsfrage" oder "Hier will jemand sein Passwort zurücksetzen".

**2. Meine Werkzeuge: Was steckt unter der Haube?**

Ich habe auf ein bewährtes Zero-Shot-Modell namens **MoritzLaurer/mDeBERTa-v3-base-mnli-xnli** gesetzt. "Zero-Shot" ist genial, weil man dem Modell keine Beispiele beibringen muss. Man gibt ihm einfach eine Anfrage und eine Liste von Kategorien und es sucht sich die passendste aus. Perfekt für einen schnellen Prototypen!

Das Ganze wird angetrieben von zwei zentralen Libraries:

* **transformers:** Quasi die universelle Fernbedienung, um mit so gut wie jedem KI-Modell auf der Hugging Face-Plattform zu quatschen. Nimmt einem unglaublich viel Arbeit ab.
* **torch:** Der Motor. Kümmert sich im Hintergrund um die ganze komplexe Mathe, damit die KI überhaupt "denken" kann.

**3. Die Reise: Ein wilder Ritt mit vielen "Facepalms"**

Der Weg zum finalen Code war… holprig. Es war ein ständiges Hin und Her aus guten Ideen und brutalen Realitäts-Checks.

**Phase 1: Der "Viel-hilft-viel"-Ansatz**  
Ich dachte mir, ich füttere die KI mit superlangen, detaillierten Beschreibungen für jede Kategorie. Mehr Infos = schlauere KI, oder?

* **Realität:** Falsch. Die KI war total überfordert und hat nur noch geraten. **Erkenntnis #1: Kleine KIs mögen’s kurz und knackig.**

**Phase 2: Der "Weniger-ist-mehr"-Ansatz**  
Okay, also das genaue Gegenteil: extrem kurze Labels.

* **Realität:** Schon viel besser! Klare Anfragen hat sie super erkannt. Aber bei allem, was ein bisschen mehrdeutig war, war sie wieder aufgeschmissen.

**Phase 3: Der "Dream-Team"-Ansatz (oder auch: Der Pipeline-Albtraum)**  
Die Idee klang genial: Ein Modell nur für die Emotion, eines für die Absicht, eines für das Thema! Ein Team aus Spezialisten!

* **Realität:** Eine komplette Katastrophe. Das Gefühls-Modell hat "Passwort vergessen" als NEGATIVE Emotion eingestuft. Es hat den neutralen Support-Kontext einfach nicht kapiert. Weil der erste Experte schon Müll geliefert hat, war der Rest der Kette eine einzige Fehler-Party. **Erkenntnis #2: Eine Kette ist nur so stark wie ihr schwächstes Glied.**

**4. Die finale Version: Pragmatisch, ehrlich, funktionsfähig**

Nach all den Experimenten bin ich zur einfachsten und gleichzeitig stabilsten Lösung zurückgekehrt: **Ein einzelnes Modell mit klaren, kurzen Labels.**

Der finale Code macht genau das. Er wartet auf eine Anfrage, lässt das Modell die beste Kategorie finden und – ganz wichtig – sagt ehrlich, wenn er sich unsicher ist. Anstatt eine falsche Antwort rauszuhauen, gibt er lieber zu, dass er Hilfe braucht und schlägt Alternativen vor. Das ist zwar nicht perfekt, aber es ist ehrlich und in der Praxis viel nützlicher.

Die komplette Anleitung, wie man das Ding startet, steht in der README.md.

**5. Mein ehrliches Fazit: Was kleine KIs wirklich können (und was nicht)**

Dieses Projekt war ein echter Augenöffner. Ich wollte herausfinden, ob eine kleine KI eine große Aufgabe lösen kann. Die Antwort ist: **Jein.**

* **Kleine KIs sind keine Mini-GPTs.** Sie sind spezialisierte Werkzeuge, keine kreativen Alleskönner. Man muss lernen, ihre Sprache zu sprechen, nicht andersherum.
* **Der Kontext ist der Endgegner.** Unsere Modelle, trainiert auf allgemeinen Texten, haben den spezifischen Kontext von Support-Tickets oft missverstanden. Das war die größte Hürde und der Grund für die meisten Fehler.
* **"Funktioniert so lala" ist auch ein Ergebnis.** Und zwar ein verdammt wichtiges. Mein finales Programm ist kein 100%iger Erfolg, und das ist okay. Das Projekt hat erfolgreich bewiesen, dass die kleinen Allzweck-Modelle für diese Art von nuancierter Aufgabe an ihre Grenzen stoßen.

**Was ich also wirklich gebaut habe, ist kein perfekter Ticket-Sortierer, sondern der Beweis, dass wir für eine wirklich gute Lösung den nächsten Schritt gehen müssen: das Fine-Tuning.** Man müsste ein Basis-Modell wie DistilBERT nehmen und es mit echten Kundendaten trainieren. Nur so lernt es die Feinheiten, an denen wir hier gescheitert sind.

Das Projekt endet also nicht mit einer magischen Lösung, sondern mit einer klaren, professionellen Erkenntnis: Wir haben das Maximum aus den verfügbaren Mitteln herausgeholt und wissen jetzt genau, was als Nächstes zu tun ist, um es richtig gut zu machen.