

**«QAI»**

**«Quick Line: DIGITALER KUNDENDIENST DER NÄCHSTEN GENERATION»**

**Technische Informationen für die Jury**

**Technische Informationen für die Jury**

Aktueller Stand des Sourcecodes

* Link zu Github Repository
  + https://github.com/MonsieurWave/quick-help-ai

Ausgangslage

* Worauf habt ihr euch fokussiert?
  + Wir haben versucht eine allgemeine und intuitive Lösung zu finden, um Kunden-Fragen zu beantworten.
* Welche technischen Grundsatzentscheide habt ihr gefällt?
  + Um einer möglichst menschlichen Kundenbetreuung nahe zu kommen, haben wir das «general-purpose» Natural Language Processing Modell GPT3 benutzt.

Technischer Aufbau

* Welche Komponenten und Frameworks habt ihr verwendet?
  + Puppeteer
  + spaCy: Word to Vector Encoding, pretrained auf einem deutschen Nachrichten-Korpus
  + HuggingFace Transformer: gpt2
  + API vom GPT-3
  + React
  + Languages: Javascript, python
* Wozu und wie werden diese eingesetzt?
  + Mit Puppeteer haben wir Fragen und Antworten aus dem Community-Forum, der FAQ sowie weiterer Ressourcen der QuickLine Webseite herausgesucht.
  + Mit spaCy haben wir eine «semantic similarity measure» implementiert, um die relevantesten Daten aus den Fragen und Antwort Paaren herauszusuchen.
  + Diese Beispiele wurden dann als «Framing» für das GPT-3 Modell benutzt. Das gibt der «general-purpose» AI den nötigen Kontext um die Frage der Kunden zu beantworten ohne neu trainiert zu werden.
  + Mit dem von GPT2 Transformer von HuggingFace, finden wir die beste Grösse der Framing-Daten.
  + Mit React bringen wir unsere API in einer einfachen und benutzerfreundlichen Demo-Interface unter.

Implementation

* Gibt es etwas Spezielles, was ihr zur Implementation erwähnen wollt?
  + Unsere Implementation kommt der menschlichen Kundenbetreuung so nah wie möglich. Das heisst es kommt mit (fast) allen Kundenfragen klar und gibt eine Intuitive Antwort. Falls der Kunde nach dem Wetter fragen will, geht das auch.
* Was ist aus technischer Sicht besonders cool an eurer Lösung?
  + Unsere Lösung basiert (fast) gänzlich auf gut angepasster künstlicher Intelligenz. Dank einem Word2Vector Encoder der auf einem Nachrichten Korpus in deutscher Sprache trainiert wurde finden wir relevante Fragen/Antworten zur Frage des Kunden. Anhand dieser formuliert dann ein zweites, grösseres GPT3 Modell eine intuitive Antwort auf diese spezifische Frage.
  + Mit nur einer minimalen Anzahl en Framing-Beispielen findet das Model eine Lösung für den Kunden. Dabei ist nichts hart angeben, und das Modell ist absolut frei in seiner Wortwahl.
  + Das Model funktioniert in Echtzeit, d. h. der Kunde hat eine echte «Chat»-Experience.

Abgrenzung / Offene Punkte

* Welche Abgrenzungen habt ihr bewusst vorgenommen und damit nicht implementiert? Weshalb?
  + Unser Scraper ist nicht perfekt, und erfasst nicht alle Fragen/Antworten auf der QuickLine Webseite. Da diese in echten Bedingungen in einer Database gespeichert sind, ist ein Scraper ausserhalb eines Hackathons nicht nötig und hier nur eine Notlösung da wir auf diese Database keinen Zugriff hatten.
  + Wegen der zeitlichen Limitierung eines Hackathons, wurde keine der AI-Modelle auf den QuickLine Daten trainiert. Dies hätte eine Woche bis zu einem Monat gedauert. Dies haben wir mit dem «Framing» der Kundenfragen überbrückt.
  + Durch die zeitiche, rechnerische und Kosten Einschränkung haben wir nur eine limitierte Anzahl an Framing-Beispielen benutzt (10-20, je nach Frage). Dies kann jedoch einfach angepasst werden, und verbessert die Genauigkeit des Models erheblich. Durch weitere Beispiele, können auch Ton und Verhaltensart angepasst werden.