Computational and Data Science
Computer Science
Dozent: Ana Petrus

Über Ras

Bot-Übersetzer für Schweizerdeutsch

Szabo Ivan[1,*]

[1] *Fachhochschule Graubünden*

**E-Mail Adresse:* ivan.szabo@stud.fhgr.ch

Januar 31, 2024

Abstract:
In unserer Zeit haben viele Ausländer Schwierigkeiten mit der schriftlichen Kommunikation in der Schweiz, selbst wenn sie die Hochsprache perfekt beherrschen. Ich habe diese Herausforderung selbst erlebt, und deshalb soll dieses Projekt dazu beitragen, diese Problematik, wenn nicht vollständig zu lösen, so doch deutlich zu entschärfen.

1 Einleitung

1.1 Vorstellung des Projekts

Das Projekt, das in dieser Arbeit behandelt wird, ist ein Telegram-Bot, der in der Lage ist, Schweizerdeutsch in Standarddeutsch zu übersetzen. In der Schweiz wird häufig Schweizerdeutsch als Alltagssprache verwendet, während Standarddeutsch die offizielle Schriftsprache ist. Dieser Bot soll als nützliche Hilfe dienen, um die Sprachbarriere zwischen den beiden Varianten zu überwinden, insbesondere in schriftlicher Kommunikation.

Die Funktionsweise des Bots basiert auf der Eingabe von Texten in Schweizerdeutsch, die dann automatisch ins Hochdeutsche übersetzt werden. Dies könnte für den Alltag vieler Menschen in der Schweiz, die sowohl in Schweizerdeutsch als auch in Standarddeutsch kommunizieren, eine wertvolle Unterstützung sein. Das Ziel des Projekts ist es, einen benutzerfreundlichen und effektiven Übersetzungsservice anzubieten, der auf der Telegram-Plattform zugänglich ist.

1.2 Wahl der Plattform

Die Wahl von Telegram als Plattform für den Bot basiert auf mehreren Überlegungen. Im Vergleich zu WhatsApp bietet Telegram zahlreiche Vorteile, vor allem im Hinblick auf die Kosten und die Flexibilität der Bot-Entwicklung. Telegram bietet eine kostenlose API und ermöglicht es, Bots einfach zu erstellen und zu integrieren. WhatsApp bietet zwar auch eine Bot-Unterstützung, jedoch ist es nicht umsonst, was für ein kleines Projekt wie dieses unpraktisch wäre.

Obwohl Telegram in der Schweiz und den angrenzenden Ländern weniger populär ist als WhatsApp, bietet es eine robuste und zuverlässige Plattform für die Entwicklung von Bots. Darüber hinaus ist Telegram in anderen Ländern weit verbreitet, was potenziell eine breitere Nutzerbasis ermöglicht.

2 Forschungsfrage

Die zentrale Forschungsfrage dieser Arbeit lautet: Wie kann ein Übersetzungs-Bot effektiv und autonom arbeiten? Diese Frage umfasst mehrere Aspekte, darunter die Auswahl der richtigen Technologien, die effiziente Verarbeitung von Übersetzungsanforderungen und die Schaffung eines autonomen Systems, das ohne ständige manuelle Eingriffe funktioniert.

Ein weiteres Problem, das in dieser Arbeit behandelt wird, ist der fehlende Zugang zu einer kostenpflichtigen API. Diese stellt eine erhebliche Herausforderung dar, da viele Übersetzungsdienste (in diesem Fall - ChatGPT) nur gegen Gebühr zugänglich sind. Es wird untersucht, wie diese Einschränkung überwunden werden kann und welche Auswirkungen sie auf die Funktionsweise des Bots hat.

3 Methodik

3.1 Verwendete Materialien & Bibliotheken

Für die Entwicklung des Telegram-Bots wurden verschiedene Materialien und Bibliotheken eingesetzt:

- **python-telegram-bot**: Diese Bibliothek ermöglicht die einfache Implementierung der Bot-Funktionalität. Sie ist sehr gut dokumentiert und wird häufig für die Erstellung von Telegram-Bots verwendet. Sie bietet eine einfache Möglichkeit, Nachrichten zu empfangen und zu senden sowie Benutzeranfragen zu verarbeiten.
- **openai**: Diese Bibliothek wird für die Integration der Übersetzungs-API verwendet. Sie ermöglicht den Zugriff auf leistungsstarke Sprachmodelle, die zur Übersetzung von Texten genutzt werden können. OpenAI bietet eine flexible API, die sowohl für einfache Aufgaben als auch für komplexere Anwendungen verwendet werden kann.
- **telegram.ext**: Diese Erweiterung der python-telegram-bot-Bibliothek bietet Hilfsklassen wie Application und MessageHandler, die das Erstellen und Verwalten von Telegram-Bots vereinfachen. Sie ermöglichen es, Nachrichten zu empfangen, Befehle zu verarbeiten und auf Benutzerinteraktionen zu reagieren.

4 Resultate

4.1 Projektstatus

Aktuell ist das Projekt nicht funktionsfähig, da der benötigte API-Zugang zu OpenAI noch nicht eingerichtet wurde. Ohne diesen Zugang kann der Bot keine Übersetzungen durchführen, was die gesamte Funktionalität des Systems blockiert. Dieses Problem hat sich als eine der größten Herausforderungen des Projekts herausgestellt.

4.2 Analysierte Lösungsansätze

Mehrere Lösungsansätze wurden zur Überwindung dieses Problems analysiert:

- **Statische Datenbank für Übersetzungen**: Eine mögliche Lösung wäre, eine Datenbank mit vorgefertigten Übersetzungen zu erstellen. Diese Methode wäre jedoch sehr ineffizient, da sie nur eine begrenzte Anzahl von Übersetzungen unterstützen würde und schwer skalierbar ist.
- **API-Zugang kaufen**: Der Kauf von API-Zugang würde eine zuverlässige und effiziente Lösung darstellen. Allerdings wäre dies mit erheblichen Kosten verbunden und würde zu einer Abhängigkeit von externen Anbietern führen, was die Autonomie des Projekts einschränkt.
- **Nutzung eines Cloud-Servers**: Ein Cloud-Server könnte eine praktikable Lösung darstellen, da er die benötigte Rechenleistung bereitstellt. Allerdings sind die langfristigen Kosten für die Nutzung eines Cloud-Servers relativ hoch, was für ein Projekt in dieser Größenordnung problematisch sein könnte.
- **Eigenes Übersetzungsmodell trainieren**: Diese Lösung wäre optimal, da sie vollständige Autonomie ermöglichen würde. Die Entwicklung eines eigenen Modells erfordert jedoch erhebliche Ressourcen in Bezug auf Rechenleistung und Daten.

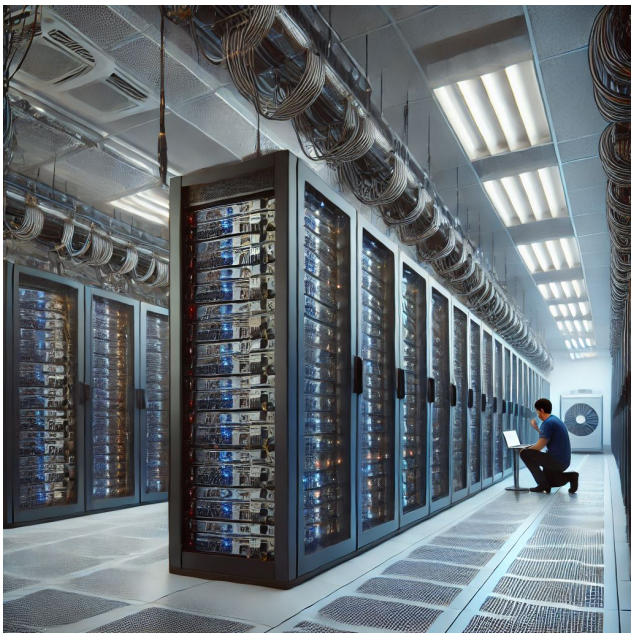

Figure 1: CDS-Traum

5 Diskussion

5.1 Optimierungsmöglichkeiten und Perspektiven

Es gibt mehrere Möglichkeiten, das Projekt weiterzuentwickeln und zu optimieren:

- **Technische Skalierung**: Der Bot könnte erweitert werden, um auch andere Schweizer Dialekte zu unterstützen. Dies würde den Nutzen des Bots für eine größere Anzahl von Nutzern erhöhen und den Bot zu einem noch wertvolleren Werkzeug machen.
- **Einbindung weiterer Sprachen**: Eine Erweiterung auf andere offizielle Sprachen der Schweiz, wie Rätoromanisch, Italienisch und Französisch, könnte den Bot für ein breiteres Publikum zugänglich machen.
- **Alternative Formate**: Anstatt sich auf Telegram zu beschränken, könnte der Bot auch als Erweiterung für andere Messenger oder sogar als Systemintegration auf Betriebssystemebene entwickelt werden.

5.2 Schwachstellen des Projekts

Ein großes Problem des Projekts sind die technischen Abhängigkeiten von externen APIs und die damit verbundenen Kosten. Ein weiteres Problem ist der hohe Ressourcenbedarf, wenn ein eigenes Übersetzungsmodell entwickelt werden soll.

5.3 Vorteile und langfristige Perspektiven

Trotz dieser Schwächen bietet die Entwicklung eines eigenen Übersetzungsmodells langfristige Vorteile, da der Bot dadurch völlig autonom arbeiten könnte. Eine autonome Lösung würde die Abhängigkeit von externen Diensten verringern und könnte die Kosten auf lange Sicht senken.

6 Fazit

Zusammenfassend lässt sich sagen, dass das Projekt derzeit noch nicht als präsentables Produkt für den Massenmarkt betrachtet werden kann. Mit einer relativ einfachen Lösung, wie dem Kauf eines API-Zugangs, könnte es jedoch in bestimmten Fällen bereits verwendet werden. Gleichzeitig bietet das Projekt großes Potenzial für Entwicklung und Skalierung. Nach meinen Berechnungen gibt es eine enorme Nachfrage nach einer solchen Lösung in bestimmten Gesellschaftsschichten. Die analysierten Lösungsansätze bieten verschiedene Möglichkeiten zur Überwindung der aktuellen Einschränkungen, wobei jede Lösung ihre eigenen Vor- und Nachteile mit sich bringt. Besonders vielversprechend erscheint die Entwicklung eines eigenen Übersetzungsmodells, auch wenn dies erhebliche Ressourcen erfordert. Der Vorteil der Autonomie durch eine eigene Entwicklung ist nicht zu unterschätzen und stellt einen wichtigen Aspekt für die Zukunft des Projekts dar.

7 Anhang

- Bot: `https://t.me/Rasberry_Projekt_Bot`
- Code: https://github.com/IvanSabov/UeberRas.git
- Latex Tutorial(YouTube): `@lectory_fpmi`
- Das Bild: generiert bei Copilot
- Die Korrektur: Für die Prüfung der Grammatik und Rechtschreibung wurde ChatGPT benutzt.