Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Projektbericht

Im Rahmen des Projekts „BigData“ WS 17/18

“Chatbot”

Felicitas Castrian
Merlin Sewina
Paul Offner

Inhaltsverzeichnis

1. Einleitung

Im Rahmen des Moduls „Projekt“ an der Universität Hamburg im Fachbereich Informatik wurde sich mit dem Thema „Big Data“ auseinandergesetzt. Dabei ging es in erster Linie darum, den Umgang mit großen Datenmengen und ihren charakteristischen Eigenschaften zu vermitteln. Zu diesem Zweck wurden verschiedene Unterthemen zur Auswahl gestellt. Unsere Gruppe legte ihren Fokus dabei auf das Thema „Chatbot“. Das Konzept des digitalen Ansprechpartners des Users auf Internetseiten gewinnt immer mehr an Popularität. So kann dem User schnell Hilfestellung zu simplem Problemen geleistet werden, ohne personellen Aufwand zu haben. Deshalb finden Chatbots bereits in unterschiedlichsten Bereichen und Branchen ihren Einsatz. Von Fluggesellschaften über Online-Versandhändler bis hin zu Banken. Unsere Aufgabenstellung bezieht sich auf einen Chatbot für die Website des Deutschen Klimarechenzentrums.

2. Aufgabenstellung

Ziel des Projekts ist die Entwicklung eines Chatbots, welcher Supportanfragen niedriger Level für die Website des Deutschen Klimarechenzentrums bearbeiten kann. Insbesondere soll der Chatbot im Stande sein, anhand einer Anfrage via Link auf nützliche Webseiten zu verweisen. Dabei beschränkt sich unsere Entwicklung auf die Homepage des Deutschen Klimarechenzentrums. Sie ist jedoch allgemein gehalten und somit auch anderweitig einsetzbar. Die ursprüngliche vorgegebene Themenidee war es, den Chatbot mit Werkzeugen aus dem Bereich „Machine Learning“ auszustatten. Dies ist prinzipiell durchaus sinnvoll, kann jedoch in diesem Projekt aufgrund des großen Umfangs nicht umgesetzt werden.

3. Realisierung

Der Fokus lag zuerst einmal in der Implementation eines Bots für Supportanfragen simplerer Art. Der Chatbot bekommt eine Frage des Users gegeben und soll daraufhin den Link zu der Seite herausgeben, auf der sich die passende Antwort zur Frage befindet.
Im Zuge der Überlegung der Herangehensweise ergaben sich folgende fünf Teilprogramme, die näher zu erläutern sind.

1. Crawler

   Der Crawler dient zur Speicherung aller Daten, die auf der Website des DKRZs vorhanden sind. Er bildet damit den Grundstein des Projekts.

2. Parser

   Dieser dient dazu die gesammelten Daten auf das wesentliche zu reduzieren.

3. Analyzer

   Der Analyzer analysiert die reduzierten Daten auf die Anzahl der Wörter und erstellt daraufhin eine Statistik.

4. Datenbank

   In der Datenbank werden die Analysierten Daten hinterlegt.

5. Chatbot

   Der Chatbot dient zur Kommunikation mit dem User und greift auf die Datenbank zu, um Anfragen zu beantworten.

3.1 Crawler

Design:

Damit der Bot im Stande sein kann, auf Anfragen zu reagieren und auf die jeweiligen Links zu verweisen, bedarf es der gesamten Daten der DKRZ-Website[1]. Der Crawler bekommt eine Ausgangs-URL gegeben und speichert rekursiv alle Links als HTML-Datei in einem Verzeichnis ab, um sie dem Bot so zur Verfügung zu stellen.

Implementation:

Der Crawler bekommt einer „rootUrl“ gegeben und erstellt einer leere Liste in der die gefundenen Links abgespeichert werden. Dazu durchsucht er jede Seite nach Links und speichert diese ab. Dies macht er solange, bis er keine Links mehr findet, die er noch nicht besucht hat.
Für detaillierte Informationen siehe Code.

Leistungsanalyse:

Der Crawler ist relativ langsam, da er bisher nicht parallelisiert wurde. In einer repräsentativen Messung wurden innerhalb von 160 Minuten 5.003 Links gecrawlt, mit einer gesamten Datengröße von 1,44GB.

[1] Deutsches Klimarechenzentrum (DKRZ)

3.2 Parser

Design:

Der Parser zieht aus den abgespeicherten HTML-Dateien die relevanten Texte raus und speichert sie als reduzierte HTML-Dateien ab.

Implementation:

Die Implementation erfolgte mittels der Programmierbibliothek „Beautiful Soup“

Für detaillierte Informationen siehe Code.

Leistungsanalyse:

Der Parser arbeitet schnell. Es bedarf keinerlei Verbesserung der Performanz.

3.3 Analyzer

Design:

Der Analyzer iteriert durch alle geparsten Texte und erstellt auf dieser Basis eine Statistik zu jeder Datei, welche Wörter vorkommen und in welcher Anzahl. Außerdem wird eine Gesamtübersicht aller Wörter erstellt. Er dient als Basis für unsere Datenbank.

Implementation:

Als erstes überprüft der Analyzer die im Verzeichnis gespeicherten Dateien auf unzulässiges Dateiformat. Daraufhin werden die Satzzeichen entfernt und alle Wörter gezählt. Auf Basis dessen wird zu jeder Link-Datei eine Statistik-Datei mit Wörtern und deren Anzahl angelegt, sowie eine Datei für alle Wörter insgesamt.
Für detaillierte Informationen siehe Code.

Leistungsanalyse:

Der Analyzer arbeitet relativ schnell, könnte zur Verbesserung jedoch auch parallelisiert werden.

3.4 Datenbank

Design:

Die Datenbank besteht aus zwei Tabellen. In einer sind alle gefundenen Wörter hinterlegt und in der zweiten ihre zugehörigen Links. Jedes Wort „weiß“ hierbei, welche seine zugehörigen Links sind, so dass bei einer Anfrage alle passenden Webseiten ausgegeben werden können.

Implementation:

Die Implementation erfolgte mittels SQLite.

Für detaillierte Informationen siehe Code.

Leistungsanalyse:

Standard Datenbank

3.5 Chatbot

Design:

Der Chatbot besteht verfolgt einen einfachen vorgefertigten Dialog, dem der User folgt, bis er ihn verlässt. Grundsätzlich muss eine Nutzeranfrage auf ihre Schlüsselwörter reduziert werden, sodass der Chatbot eine entsprechende Anfrage an die Datenbank stellen kann. Die Links, die als Ergebnis der Abfrage herauskommen, werden an den User weitergegeben.

Implementation:

Nach seinem Start begrüßt der Chatbot den User und wartet auf die erste Anfrage. Smalltalk ist nicht vorgesehen, daher behandelt der Chatbot jedwede Anfrage als ernsthaften Informationsbedarf. Eine Anfrage wird auf eine Liste kleingeschriebener Wörter reduziert und von Stopwörtern, also Wörtern ohne inhaltliche Relevanz, bereinigt. Anhand dieser Wortliste wird dann eine Anfrage an die Datenbank gestellt und dem User eine Liste von Links übergeben. Nach jeder Anfrage vergewissert sich der Chatbot, ob der User die Kommunikation fortsetzen möchte und entsprechend seiner Antwort beendet er den Dialog oder wartet auf die nächste Anfrage.
Für detaillierte Informationen siehe Code.

Leistungsanalyse:

Der Chatbot läuft sehr schnell.

3.6 Probleme & Lösungen

Im folgenden Abschnitt werden Probleme und Lösungen im Verlauf des Projekts geschildert:

Eines der größten und zeitaufwendigsten Probleme des Projekts stellte das Missverständnis über die Programmiersprache dar. Entgegen unserer ursprünglichen Annahme, das Projekt könne in Java umgesetzt werden, welches uns die Umsetzung einer Rest-API mittels Spring-Boot vereinfacht hätte, wurde zu einem deutlich späteren Zeitpunkt kommuniziert, dass eine Umsetzung in Python erforderlich ist. Dies führte zum zeitintensiveren Umschreiben des bis dahin bereits implementierten Java Codes zu Python Code.

Weitere Probleme gab es unter anderem mit dem Crawler. Da sich auf der Seite des Deutschen Klimarechenzentrums auch diverse Links zu PDF Dateien und Videos befinden, hat der Crawler ebenfalls diese gecrawlt und im Verzeichnis abgelegt. Da der Parser aber keine Informationen aus den PDF Dateien oder gar den Videos ziehen kann, sollte dieser die entsprechenden Links ignorieren. Dieses Problem haben wir durch das Überprüfen der Dateien auf ASCII-Zeichen gelöst. Es werden nur jene Dateien geparst, die ausschließlich ASCII-Zeichen enthalten.

Bei dem Parser gab es ebenso Probleme, da unser ursprünglicher Ansatz zur Reduzierung der HTML-Dateien auf die reinen Informationen nicht wie gewünscht funktionierte. Hierbei war die Grundidee, den Aufbau einer HTML Seite zu nutzen und die Seite auf die „Paragraphs“ im „Body“ zu beschränken. Da jedoch viele der Links auf der DKRZ Seite sehr verschachtelt strukturiert sind, war dieser Ansatz nicht tauglich. Gelöst wurde das Problem durch den Einsatz der Programmierbibliothek „Beautiful Soup“, welche HTML und XML Dateien parsen kann und dabei den vollständigen Inhalt aus den Dateien zieht.

Bei dem Analyzer traten insofern Probleme auf, als dass aufgrund von diversen Rechtschreibfehlern und fehlendem einheitlichen Schreibformat (wie etwa gleiche Anzahl an Leerzeichen zwischen Wörtern in den HTML-Dateien) Wörter nicht richtig gezählt werden

konnten. Dieses Problem konnte nicht gelöst werden. Es wurde dabei belassen, die Abweichungen in der Zählung hinzunehmen, in der Annahme, dass es sich dabei um keine ausschlaggebende Anzahl handelt.

Schwierigkeiten gab es auch bei der Schlagwortsuche des Chatbots. Um sinnvoll nach entsprechenden Links zu suchen, bedarf es der Reduzierung der Anfrage auf die entscheidenden Schlagwörter. Dabei haben wir eine Liste mit „Stopwords“ eingesetzt, um die Anfrage zu reduzieren. Hierbei handelt es sich um Wörter, die keine inhaltliche Relevanz haben (beispielsweise „bei“, „dass“, „was“) und daher herausgefiltert werden können. Dieser Ansatz reduziert die Wortliste bereits, allerdings bleiben in den häufigsten Fällen mehr als nur die Schlüsselwörter übrig. Um die Wortanzahl weiter zu reduzieren, war der Einsatz von überwachtem Lernen angedacht. Dieser Ansatz musste in Ermangelung von genügend Trainingsdaten aber wieder verworfen werden.

3.7 Ausblick

Um das Projekt performanter, sowie universell einsetzbarer zu gestalten, gäbe es einige sinnvolle Ansatzpunkte. Angefangen bei der Umsetzung des Chatbots als Webanwendung. Da es dem User möglich sein soll Anfragen an den Chatbot über die Website stellen zu können, wäre dies für den vollen Funktionsumfang essentiell. Angelehnt an die Webanawendung, sollte das Hosten der Datenbank als Container auf einem Server umgesetzt werden.

Generell könnte man die Teilprogramme zusammenfügen und dadurch das manuelle nacheinander Ausführen vermeiden, dies wäre auch deutlich effizienter. Um die Allgemeine Laufzeit zu verbessern, wäre einer Parallelisierung des Crawlers und des Parsers sinnvoll.

Am wichtigsten ist jedoch eine Verbesserung der Schlagwortanalyse. Performanz und Benutzbarkeit werden nebensächlich in Anbetracht der Funktionalität. Der Chatbot muss vor allem in der Lage sein, die Anzahl der hinterlegten Links zu senken oder zumindest zu gewichten und zu priorisieren. Dies könnte zum Beispiel durch die Bewertung der Tauglichkeit eines Links seitens der User realisiert werden. Allerdings wäre der Einsatz damit zu Beginn sehr unzufriedenstellend. Alternativ könnten Trainingsdaten und maschinelles Lernen eingesetzt werden, wobei dafür natürlich erst einmal Trainingsdaten vorhanden sein müssen.

Für den User wertvoll wäre mit Sicherheit auch Mehrsprachigkeit. Die Seite des DKRZ ist mehrsprachig verfügbar. Zu einem späteren Zeitpunkt könnte der Chatbot so auch für mehrere Klimarechenzentren einsetzbar gemacht werden.

4. Anhang

4.1 Genutzte Software

- Eclipse (IDE)
- Python
- Sublime Text
- Beautiful Soup
- Python mechanize
- Jsoup
- Maven
- Git
- SQLite
- Word

4.2 Hardware

- Computer Informatikum
- MacBook
- Home Computer

4.3 Erfahrungen

Besonders herauszustellen ist zum die Erkenntnis, dass es – entgegen unserer Annahme – deutlich leichter gewesen wäre, das Programm nicht in fünf komplett voneinander getrennte Teilprojekte aufzuteilen. Stattdessen wäre es vermutlich effizienter und simpler gewesen, von Anfang an alles in einem Programm zu verknüpfen. So müsste nicht alles einzeln nacheinander ausgeführt und geladen werden. Dies könnte unter Umständen auch eine verbesserte Laufzeit bewirken.

Ebenso erwähnenswert ist die Feststellung, dass die Implementationen von externen Bibliotheken nicht immer wie erwartet oder vorgegeben funktionieren. Im Rahmen unserer Implementation des Parsers mittels „Beautiful Soup“ stellte sich heraus, dass aufgrund des eher unüblicheren Aufbaus der Html-Seite, die vorgegebene Funktion nicht die erwarteten Werte zurück geliefert hat. Gelöst werden konnte das Problem dennoch, da Beautiful Soup weitere Funktionen anbietet, die in diesem Fall geholfen haben.

Bezüglich der Erfahrungen über den allgemeinen Ablauf des Projekts, lässt sich sagen, dass die Zeitintensivität der einzelnen Schritte doch relativ hoch war und dadurch die ursprünglich vorgenommenen Ziele nicht vollkommen erfüllt werden konnten. Dies hing unter anderem mit der doppelten Implementierung in unterschiedlichen Programmiersprachen zusammen. Diesbezüglich wäre es wichtig, in Zukunft auf eine bessere Kommunikation zu achten.

4.4 Aufteilung Arbeitsbereiche

Bei der Bearbeitung der Teilaufgaben gab es innerhalb der Gruppe keine feste Aufgabenverteilung. Alle Gruppenmitglieder haben sich sowohl an der Planung, der Implementation, der Präsentation, als auch der Erstellung des Projektberichts beteiligt.

4.5 Literaturangaben

- https://t3n.de/news/chatbots-messenger-marketing-2-837706/