BookSearch

Entwicklung eines Buchempfehlungs-Chatbots

Dokumentation

Hannah Bultmann
hannah.bultmann@uni-bielefeld.de

Universität Bielefeld
Fakultät für Linguistik und Literaturwissenschaft
Sommersemester 2025

Inhaltsverzeichnis

Die Projektdateien finden sich unter:

https://github.com/hannahbultmann/book-search

1 Einleitung

In vielen digitalen Buchplattformen dominieren Empfehlungen auf Basis von Genres, Bestseller-Rankings oder dem Verhalten der Nutzer*innen. Diese Systeme ermöglichen jedoch kaum eine gezielte, inhaltlich motivierte Suche über Freitextanfragen. Ziel dieses Projekts ist es daher, ein Empfehlungssystem in Form eines Chatbots zu entwickeln, das auf individuelle Suchanfragen der Nutzer*innen eingeht und so Bücher nach inhaltlicher Relevanz vorschlägt.

Die Grundlage bildet ein Korpus aus bibliographischen Daten der Open Library. Suchanfragen in natürlicher Sprache (z. B. „*A fantasy book about dragons and magic*“) werden vom Chatbot verarbeitet und mit den Buchbeschreibungen im Korpus verglichen. Dazu werden die Suchanfragen sowie die Buchbeschreibungen vektorisiert. Über die Berechnung der Kosinusähnlichkeit wird anschließend die semantische Nähe zwischen Anfrage und Beschreibung ermittelt, um relevante und thematisch passende Bücher zu finden und auszugeben.

2 Der Datensatz

Die Open Library ist ein Projekt des *Internet Archive* mit dem Ziel, alle bisher erschienenen Bücher zu dokumentieren. Sie stellt monatliche Datendumps in TSV-Dateien zur freien Nutzung bereit, welche sämtliche Informationen zu Werken, Ausgaben und Autor*innen formatiert in JSON enthalten. Für dieses Projekt werden der *works dump* und der *authors dump* aus dem Juni 2025 verwendet (Internet Archive 2025). Diese beinhalten Einträge zu 39.831.591 Büchern und 14.547.394 Autor*innen. Um mit diesen großen Datenmengen umgehen zu können und eine effiziente Weiterverarbeitung zu ermöglichen, werden die Rohdaten der beiden Datendumps in eine SQLite-Datenbank importiert.

Die SQLite-Datenbank enthält teilweise Einträge ohne Beschreibungen, die somit für das Projekt nicht nutzbar sind. Daher werden nur die Einträge *mit* Beschreibung zur weiteren Verarbeitung in einen DataFrame extrahiert. Dabei werden folgende Daten aus der *works*-Tabelle abgerufen: Titel, Autor*innen-Key, Beschreibung und Datum der Erstveröffentlichung (falls vorhanden). Der DataFrame umfasst ungefähr 1,6 Millionen Bucheinträge. Da es sich bei der OpenLibrary allerdings um eine Crowd-Sourcing Plattform handelt, ist nicht jede Beschreibung eine tatsächliche Zusammenfassung des Buches: Manche enthalten nur Informationen über das Genre, das Format, die Quelle oder die Herausgeber*innen des Buches. Auch sind sie in unterschiedlichen Sprachen verfasst. Bevor die Beschreibungen für das Chatbot-Projekt verwendet werden können, müssen sie also aufbereitet werden.

Mithilfe regulärer Ausdrücke werden irrelevante Beschreibungen wie HTML-Tags, URLs und E-Mail-Adressen herausgefiltert. Die unterschiedliche Qualität und die verschiedenen Formate der Beschreibungen erschweren die automatische Bestimmung ihrer Verwendbarkeit. Um die Daten vollständig zu bereinigen, sind detailliertere Filter oder eine manuelle Durchsicht der Daten notwendig, was jedoch den Rahmen dieser Projektarbeit überschreitet.

Der Großteil der Beschreibungen ist auf Englisch verfasst worden. Daher werden zur automatischen Sprachidentifikation das Modell *fastText* (Joulin et al. 2016) auf den Datensatz angewendet und alle Beschreibungen entfernt, die nicht mit einer Wahrscheinlichkeit von 99,9 % als Englisch identifiziert werden. Übrig bleiben etwa 800.000 Bucheinträge. Eine so hohe Wahrscheinlichkeitsgrenze zu wählen, ist an dieser Stelle nicht notwendig, sondern dient viel mehr dazu, die Gesamtzahl der Bücher im endgültigen Datensatz zu reduzieren. Die Gesamtzahl beeinflusst maßgeblich die Geschwindigkeit des Chatbots: Je größer der Datensatz ist, umso mehr Berechnungen müssen durchgeführt werden, um die passenden Bücher zu finden. Aus diesem Grund und um einheitliche Ergebnisse des Chatbots zu gewährleisten, werden für dieses Projekt alle Bücher entfernt, für die kein Veröffentlichungsjahr vorliegt.

Der bereinigte Datensatz umfasst nun etwa 141.000 Bucheinträge, was eine angemessene Verarbeitungsgeschwindigkeit des Chatbots gewährleistet. Das jüngste Buch im Datensatz stammt aus dem Jahr 2018, während der Großteil der Bücher in den 1990ern und frühen 2000ern veröffentlicht wurde.

Abschließend werden die Autor*innen-Keys durch die Namen ersetzt, die aus der *authors*-Tabelle der SQLite-Datenbank abgerufen werden. Der DataFrame ist nun bereit für die anwendungsspezifische Vorverarbeitung, die im nächsten Kapitel beschrieben wird.

3 Preprocessing

Die Buchbeschreibungen sollen später für die Berechnungen von dem Chatbot vektorisiert werden. Um sie sinnvoll in einen Vektorraum abbilden zu können, müssen sie wie bei fast jeder natürlichen Sprachverarbeitung normalisiert werden (Jurafsky & Martin 2025: 23). Zunächst werden alle Satzzeichen und Zahlen entfernt, da dies eine Aufblähung des Vokabulars durch seltene Token mit wenig semantischem Gehalt vermeidet. Lemmatisierung wird mit *spaCy* unter Verwendung des Models *en_core_web_sm* durchgeführt, einer trainierten Pipeline für das Englische. Durch die Lemmatisierung wird jedes Token auf seine Grundform (Lemma) reduziert und so verschiedene Wortformen zu einer Dimension im Vektorraum zusammen-

gefasst. Indem nicht jeder Typ als separates Feature behandelt wird, reduzieren sich der Umfang des Vokabulars und somit die Dimensionalität der Bag-of-Words-Repräsentationen. Lemmatisierung kann besonders für ähnlichkeitsbasierte Aufgaben und *Information Retrieval* (IR) nützlich sein, da die Überbrückung morphologischer Variationen den Recall erhöht (Jurafsky & Martin 2025: 23). In diesem Projekt müssen die Buchbeschreibungen nicht mit den genauen Wortformen des Inputs übereinstimmen.

Die Vorverarbeitung wird in einer separaten Python-Datei durchgeführt, die später als Modul in den Chatbot importiert wird, sodass dieselben Funktionen auf die Eingaben der Benutzer*innen angewendet werden können.

Die vorverarbeiteten Beschreibungen werden in eine neue Spalte des DataFrames gespeichert, der nun in seiner endgültigen Version vorliegt. Er wird im JSON-Format exportiert und ist Teil der Chatbot-Architektur, die im nächsten Abschnitt vorgestellt wird.

4 Chatbot-Architektur

4.1 Information Retrieval Layer

Als Teil der Chatbot-Architektur wird ein IR-Layer implementiert, der für die Auswahl jener Buchbeschreibungen zuständig ist, die der Suchanfrage am besten entsprechen. Dieser Layer wendet eine häufig genutzte IR-Technik an, die als „ad hoc retrieval" bekannt ist (Jurafsky & Martin 2025: 291). Dabei formuliert ein*e Nutzer*in einen Informationsbedarf (welche Art von Buch gesucht wird) und das System gibt eine nach Relevanz geordnete Rangliste an Dokumenten zurück. In diesem Projekt entsprechen die Dokumente den Buchzusammenfassungen im Datensatz.

Der IR-Layer besteht aus zwei Hauptkomponenten: (1) den *Vektor-Repräsentationen* der normalisierten Buchbeschreibungen, die auf einem Bag-of-Words-Modell mit tf-idf-Gewichtung basieren, das den semantischen Inhalt jeder Beschreibung in ein numerisches Format codiert, und (2) einem *Inverted Index*, der die einzelnen Terme den Dokumenten zuordnet, in denen sie vorkommen, und so eine effiziente Suche nach Dokumentenkandidaten unterstützt. Zusammen bilden diese Komponenten die Wissensbasis des Systems: Der Vektorraum ermöglicht ähnlichkeitsbasierte Vergleiche zwischen den Suchanfragen und den Buchbeschreibungen, während der Inverted Index sicherstellt, dass nur relevante Teilmengen von Buchbeschreibungen für die Bewertung berücksichtigt werden.

Sowohl die Buchbeschreibungen als auch die Suchanfragen werden als hochdimensionale Vektoren repräsentiert, die aus unigram word counts abgeleitet und mit dem TfidfVectorizer

von *scikit learn* implementiert werden. Die tf-idf-Gewichtung reduziert den Einfluss von häufig vorkommenden, wenig aussagekräftigen Wörtern (z. B. Funktionswörter) und betont zugleich seltene, semantisch reichhaltige Wörter, die besser zur Unterscheidung der Dokumente geeignet sind (Jurafsky & Martin 2025: 112). Dies verbessert die Relevanz der Suchergebnissen in Bezug auf die jeweilige Anfrage. Um die Relevanz eines Buches zu bewerten, berechnet das System die Kosinusähnlichkeit zwischen dem Anfrage-Vektor **q** und einem Beschreibungs-Vektor **d**. Kosinusähnlichkeit wird definiert als das normalisierte Skalarprodukt zwischen zwei Vektoren (Jurafsky & Martin 2025: 292f.):

$$score(q, d) = cos(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{|\mathbf{q}||\mathbf{d}|}$$

Da der Score von Beschreibungen, die keine Terme mit der Anfrage gemeinsam haben, null ist, ist es nicht notwendig, die Kosinusähnlichkeit für den gesamten Datensatz zu berechnen. Stattdessen wird der Inverted Index verwendet, um Beschreibungen zu identifizieren, die mindestens einen der Suchbegriffe enthalten.

Der Inverted Index wird auf Grundlage der Darstellung von Jurafsky und Martin (2025: 295) implementiert. Er wird direkt aus den Vektoren der Buchbeschreibungen zusammengestellt. Jede Buchbeschreibung wird als Zeile in der tf-idf-Matrix codiert, wobei Non-Zero-Einträge das Vorkommen und die Gewichtung eines Vokabularbegriffs in der Beschreibung angeben. Um den Index zu erstellen, iteriert das System über alle Non-Zero-Einträge und identifiziert so alle Term-Dokument-Paare mit ihren Indizes. Daraus wird eine Dictionary-Struktur erstellt, in der jeder Term des Vokabulars als Key dient. Der dazugehörige Value ist eine Liste der Indizes aller Dokumente, die diesen Term enthalten.

Durch das Nachschlagen jedes einzelnen Terms der Suchanfrage im Inverted Index und das Zusammenstellen einer Liste aller damit verknüpften Dokumentindizes werden die potentiell relevanten Buchbeschreibungen identifiziert. Nur für diese Kandidaten berechnet das System die Kosinusähnlichkeit und gibt die fünf besten Ergebnisse aus.

4.2 Dialogsystem

Der Information Retrieval Layer ist in ein Dialogsystem eingebettet, das die Kommunikation zwischen dem Chatbot und den Nutzer*innen managt. Diese Interaktionsebene ist dafür zuständig, die Anfrage aufzunehmen, die Nutzer*innen zu leiten, falls keine Ergebnisse gefunden wurden, und die endgültigen Buchempfehlungen zu präsentieren. Der Dialog ist als einfache Schleife implementiert, die die Kontinuität der Interaktion so lange gewährleistet, bis ein*e Nutzer*in diesen Prozess beendet. Um die Antwortvielfalt des Chatbots zu erhöhen,

werden bestimmte Nachrichten zufällig aus einer vordefinierten Sammlung von passenden Aussagen ausgewählt.

Der Dialogprozess umfasst vier Schritte:

1. Die Anfrage: Der Chatbot fordert dazu auf, das gewünschte Buch anhand von Schlüsselwörtern zu beschreiben. Die Eingabe wird dann auf die gleiche Weise wie die Buchbeschreibungen vorverarbeitet und vektorisiert.
2. Dokument-Retrieval (und Clarification, falls notwendig): Wie in Kapitel 4.1 beschrieben werden potentiell relevante Bücher mithilfe des Inverted Index abgerufen. Falls keine Dokumente gefunden werden, weil keiner der Suchbegriffe im Vokabular (und damit in den Buchbeschreibungen) enthalten ist, bittet der Chatbot um eine neue Anfrage. Dadurch wird der Dialogprozess erneut gestartet.
3. Ähnlichkeitsberechnung und Ergebnispräsentation: Die abgerufenen Beschreibungen werden anhand der Kosinusähnlichkeit zwischen dem Beschreibungsvektor und dem Anfragevektor bewertet. Die fünf Ergebnisse mit der höchsten Ähnlichkeit werden mit Titel, Namen der Autor*innen, Erscheinungsjahr und Buchbeschreibung ausgegeben. Werden weniger als fünf Dokumente gefunden, werden alle nach ihrer Ähnlichkeitsbewertung sortiert ausgegeben.
4. Fortsetzung oder Ende des Dialogs: Nach der Präsentation der Buchempfehlungen fragt der Chatbot, ob der*die Nutzer*in die Suche durch Eingabe einer neuen Suchanfrage fortsetzen möchte. Enthält die Antwort ein „yes“, wird eine neue Iteration der Dialogschleife initiiert. Enthält die Antwort ein „no“, verabschiedet sich der Chatbot und beendet das Programm. Bei unklaren Antworten fragt der Chatbot so lange nach, bis er eine eindeutige Antwort erhält.

5 Evaluation

Ein entscheidender Aspekt der Leistungsfähigkeit des Chatbots liegt in der Wahl der Textrepräsentation für die Beschreibungen und die Anfragen. Das in diesem Projekt verwendete System basiert auf einem Bag-of-Words-Modell mit tf-idf-Gewichtung, das durch seine Einfachheit und Interpretierbarkeit überzeugt, jedoch bestimmte konzeptionelle Einschränkungen aufweist.

Das Bag-of-Words-Modell behandelt jeden Term als unabhängige Dimension und repräsentiert jedes Dokument als hochdimensionalen Vektor. Trotz der Sparseness hat dieser Ansatz den Vorteil der Transparenz: Die Ausgabe des Systems lässt sich leicht anhand der

Schlüsselwörter erklären, die sich zwischen der Anfrage und dem Dokument überschneiden. Darüber hinaus lässt sich das Modell effizient mit einem Inverted Index implementieren, der selbst bei größeren Datenmengen einen schnellen Retrievalprozess unterstützt.

Die Unabhängigkeitsannahme des Bag-of-Words-Modells führt jedoch zu Schwächen hinsichtlich der Repräsentation der syntaktischen Struktur eines Textes sowie der semantischen Relationen zwischen Wörtern. Synonyme werden beispielsweise als unterschiedliche Features codiert, obwohl sie dasselbe Konzept ausdrücken, während die verschiedenen Bedeutungen eines polysemischen Terms auf ein Feature reduziert werden. Das Modell ist auch nicht in der Lage, Bedeutungsunterschiede zu erfassen, die sich aus der Wortstellung oder dem syntaktischen Kontext ergeben. Phrasen wie „*romantic comedy*“ und „*comedy without romance*“ werden auf eine Weise repräsentiert, die ihre semantische Unterscheidung kaum widerspiegelt, was zu sehr ähnlichen Outputs führt. Das System gibt oft Beschreibungen aus, die zwar die Suchbegriffe enthalten, aber nur teilweise der Intention der Suchanfrage entsprechen. In vielen Fällen finden sich nur Fragmente der Anfrage in der Buchbeschreibung, während die Wahrscheinlichkeit einer vollständigen inhaltlichen Übereinstimmung sehr gering ist.

Das System reagiert am besten auf Suchanfragen im Keyword-Stil, da es den gesamten Input für die Suche nach passenden Dokumenten verwendet und nicht zwischen strukturierenden Phrasen und inhaltlich relevanten Begriffen unterscheidet. Phrasen, die in natürlicher Sprache häufig verwendet werden, um beispielsweise einen Informationsbedarf auszudrücken, wie „*I am looking for a book about …*“, werden ebenfalls im Suchvektor codiert. Obwohl die tf-idf-Gewichtung dazu beiträgt, dieses Problem zu verringern, verursachen solche Phrasen dennoch eine gewisse Störung und erhöhen die Wahrscheinlichkeit für unpassende Ergebnisse.

Es gibt verschiedene Lösungen für diese Probleme, wobei jede mit eigenen Kompromissen einhergeht. Eine Option ist die Einbeziehung von N-Grammen, die zumindest eine teilweise Modellierung der Wortreihenfolge und des lokalen Kontexts ermöglichen. Dies würde jedoch die Sparseness der Vektoren erhöhen und damit die Effizienz der Suche reduzieren. Eine leistungsfähigere Alternative ist die Verwendung von distributionellen Repräsentationen wie vortrainierten, statischen word embeddings wie GloVe (Pennington et al. 2014). Diese Repräsentationen können semantische Ähnlichkeiten zwischen Wörtern erfassen und das Synonymieproblem abmildern. Ihre Anwendung ist jedoch rechenintensiver, und Dokumentvektoren basieren typischerweise auf Durchschnittswerten der word embeddings, wodurch wichtige Unterschiede verloren gehen.

Vortrainierte Modelle wie BERT, die auf kontextuellen embeddings basieren, gehen über diese kleinen potentiellen Verbesserungen hinaus (Jurafsky & Martin 2025: 298). Sie bilden Wörter, Sätze oder ganze Dokumente als dichte Vektoren in einem kontinuierlichen semantischen Raum ab. Ausdrücke mit ähnlichen Bedeutungen liegen in diesem Raum nah beieinander, sodass Anfragen und Dokumente auch dann einander zugeordnet werden können, wenn sie nicht das gleiche Vokabular teilen. Sie berücksichtigen auch den Kontext, wodurch es möglich ist, verschiedene Bedeutungen eines Wortes zu unterscheiden. Die Qualität der kontextualisierten embeddings hängt jedoch von der Repräsentativität der Trainingsdaten ab. Das Training und die Berechnung der embeddings sind wesentlich resourcenintensiver. Während der Inverted Index mit word-count-Vektoren eine schnelle Suche ermöglicht, erfordert die Berechnung der Ähnlichkeit zwischen embeddings eine nearest neighbor search, die weniger effizient ist (Jurafsky & Martin 2025: 301).

6 Fazit

Das Projekt zeigt, dass bereits mit einfachen Methoden der natürlichen Sprachverarbeitung ein funktionierender Chatbot mit Information-Retrieval-System entwickelt werden kann, der große Textmengen verarbeitet und durchsucht. Der Einsatz von Bag-of-Words-Repräsentationen erweist sich im Rahmen dieses Projekts als praktikabler Kompromiss zwischen Interpretierbarkeit, Rechenaufwand und Genauigkeit. Trotz seiner Grenzen kann der Chatbot sinnvolle und nachvollziehbare Ergebnisse liefern.

Weitere Anknüpfungspunkte eröffnen sich zum einen in der Entwicklung semantisch reichhaltigerer Textrepräsentationen, die über Wortfrequenzen hinausgehen. Ziel wäre es, die Effizienz der Berechnungen mit einer höheren inhaltlichen Tiefe zu verbinden, etwa durch den Einsatz von embedding-basierten Verfahren. Zum anderen bietet sich eine Erweiterung und Vertiefung des Datenkorpus durch Integration aktuellerer Veröffentlichungen sowie eine detailliertere Qualitätsprüfung der bestehenden Daten an.

Literaturverzeichnis

Internet Archive (2025): Open Library Data Dump, 2025-06-30. https://archive.org/details/ol_dump_2025-06-30 (retrieved 27.07.2025).

Joulin, Armand; Grave, Edouard; Bojanowski, Piotr; Mikolov, Tomas (2016): Bag of Tricks for Efficient Text Classification. arXiv, doi:10.48550/arXiv.1607.01759.

Jurafsky, Daniel; Martin, James H. (2025): Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models. Third Edition draft.

Pennington, Jeffrey; Socher, Richard; Manning, Christopher D. (2014): GloVe: Global Vectors for Word Representation.