Fakultät Informatik

Automatische Generierung von Themenspezifischen Podcasts aus einer großen Audiodatenbank

Bachelorarbeit im Studiengang Medieninformatik

vorgelegt von

Vincent Neumann

Matrikelnummer 358 5287

Erstgutachter:	Prof. Dr. Florian Gallwitz
Zweitgutachter:	Prof. Dr. Anna Kruspe

© 2023

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

Kurzdarstellung

In dieser Arbeit wird ein Ansatz für eine Automatische Erstellung einer Podcast Episode aus vorhandenem Audiomaterial vorgestellt. Dafür wird ein großes System aus mehreren Micro Services Erstellt, das den kompletten Durchschtich von der Audio-Datenbeschaffung bis zum Bereitstellung einer Podcast Episode in einem Userinterface abdeckt.

Contents

Chapter 1

Einleitung

1.1 Motivation

Podcasts sind für viele Menschen ein wichtiges Medium, um sich die Zeit zu vertreiben und sich über verschiedene Themen zu informieren. Wie Studien gezeigt haben, verbessert sich das Audioerlebnis der Zuhörer*innen dabei, wenn mehrere verschiedene Sprecher*innen dabei vorkommen [Kang 12]. Einige Podcasts werden zum Beispiel nur von einer einzigen Person vorgetragen. Außerdem steigert das Vorhandensein verschiedener Meinungen auch das Interesse des Zuhörers an einem bestimmten Thema [**?**]. Neben diesen Vorteilen, ist es für die Zuhörer*innen einer Podcast Episode auch ein besonderer Aspekt der Selbstbestimmung, wenn sie die Länge einer Podcast Episode selbst festlegen können. In dieser Arbeit wird es darum gehen, wie man diese Vorteile einer personalisierten Podcast Episode ausnutzen kann.

1.2 Zielsetzung

Das Ziel dieser Bachelorarbeit besteht darin zu untersuchen, wie sich aus umfangreichem Audiomaterial aus Radioprogrammen oder Podcasts on-the-fly ein eigener Podcast zusammenstellen lässt, der relevante Ausschnitte aus einer Vielzahl von Audiomaterial enthält.

Ein möglicher Anwendungsfall wäre ein/e Benutzer*in, die/der sich über das Thema "Überfischung der Meere" informieren möchte und dafür genau 20 Minuten während einer Autofahrt einplant. Das System erstellt nun einen Zusammenschnitt aus verschiedenen Podcast Episoden zu diesem Thema, der 20 Minuten lang ist und stellt ihn dem/der Benutzer*in zur Verfügung. Der Vorteil für den/die Nutzer*in liegt darin, dass er/sie selbst das Thema auswählen und die exakte Länge festlegen kann, um beispielsweise während einer 20-minütigen Autofahrt einen Podcast anzuhören. Außerdem werden das Thema von verschiednen Personen aus unterschiedlichen Blcikpunkten erklärt.

Für die Interaktion mit dem Benutzer soll außerdem eine Grafische Benutzeroberfläche bereitgestellt werden, die dem Nutzer die Auswahl eines Themas und die Länge der Podcast Episode ermöglicht.

1.3 Überblick

Im zweiten Kapitel wird es um die Beschaffung der Datengrundlage gehen. Als Datenquelle werden Podcast Episoden aus der Audiothek der ARD benutzt. Dann werden verschiedene Methoden zur Transkription dieser Audiodaten beschrieben und begründet, welche Wahl der Transkription für diese Arbeit verwendet wird.

Im dritten Kapitel wird die Architektur des Systems dargestellt. Dabei wird besonders auf die semantische analyse der Transkriptionen eingegangen und verschiedene Aspekte des Natural Language Processing und der Large Language Models vorgestellt.

Im vierten Kapitel werden verschiedene Methoden zur semantischen Analyse evaluiert. Dabei werden auch die verwendeten Modelle die Wahl der Anzahl und Länge der Abschnitte und die finale Zusammensetzung dieser Abschnitte erklärungt und begründet.

Im fünften Kapitel wird ein Ausblick auf zukünftige Verbesserungen gegeben.

Im sechten Kapitel wird eine Zusammenfassung der erzielten Ergebnisse gegeben.

Chapter 2

Ausgangslage und theoretische Grundlagen

2.1 Ausgangslage

Stand 2023 gibt es auf der Plattform Spotify rund 5 Millionen Podcasts die zusammen ungefähr 70 Millionen Episoden enthalten. Die Podcastbranche wächst seit vielen Jahren stetig und immer mehr Menschen hören regelmäßig Podcasts [Quelle]. In Deutschland ist die ARD Audiothek ein großer Podcastanbieter mit mitlerweile über 41 Millionen Audioabrufen und über 80.000 verschiedenen Audioinhalten zum Abrufen. Auch die Zahlen der Audiotheksbenutzer, sowie der App-downloads steigen weiterhin.

Gleichzeitig wächst auch der Markt an AI-basierten Podcasts. So erreichen heute schon AI-basierte Podcasts rund 45 Millionen US-Amerikaner [Quelle]

2.2 Embeddings

Das Ziel dieser Arbeit besteht darin, eine Schnittstelle für einen Hörer zu erstellen, die ihm ermöglicht, einen Zusammenschnitt aus vielen verschiedenen Podcast Episoden ist, und genau zu einem Thema passt. Die wichtigste Aufgabe besteht also darin, die wesentlichen Abschnitte aus den Episoden herauszufinden. Bis jetzt haben wir zu jedem Audiofile eine transcript.json, bei der das Transkript in regelmäßigen Zeitabschnitten mit Zeitstempeln markiert ist. Die Aufgabe, auf eine Userfrage hin die wesentlichsten Segmente herauszufinden lässt sich auf mehrere Arten lösen. Dazu werden wir verschiedene Verfahren aus der Wissenschaft im Bereich Natural Language Processing (NLP) verwenden.

2.3 Motivation für Embeddings

Die menschliche Sprache ist ein hochkomplexes Konstrukt mit einer Grammatik, die sehr viel flexibler, kreativer, vieldeutiger, und komplexer ist als maschinensprache. Es gibt viele kleine Bedeutungsnuancen, sie ist stark von dem allgemeinen Wissen der Welt geprägt

und sie ändert sich im laufe der Zeit. Das alles macht es für Computer sehr schwierig die Menschliche Sprache zu Verstehen. Der Wunsch die menschliche Sprache für den Computer verständlich zu machen ist fast so alt, wie die Computer an sich. In den 1940er Jahren, nach Ende des 2. Weltkrieges Der Bereich des NLP riesig. Es gibt [QUELLE] paper dazu. Es hat viele verschiedene Unterthemen. Die Aufgaben im Bereich des NlP sind vielseitig . Einige Ansätze

2.4 Transformer Architektur

Eine Transformerarchitektur ist eine der modersten und leistungsfähigsten Architekturen, um NLP Aufgaben zu lösen. Sie bildet dabei den Nachfolger bzw. Konkurenten zu den bis dato vorherschenden Rekurenten Neuronalen Netzen. Der Große Vorteil der Transformer ist, dass sie parallelisierbarer sind, da RNNs häufig mehrschrittige rekurente Abfragen bilden müssen. Das heißt pro Token Output müssen Sie unter Umständen mehrfach das Embedding des Inputs durchsuchen. Audio Segmentierung Die Audios müssen am Ende noch zugeschnitten werden. Für die Bearbeitung von Audio files in Python bietet sich das Python Modul Pydub an. Mit diesem Modul kann man ein Audiofile ähnlich wie ein Array behandeln, aus dem man nun einen Abschnitt von Sekunde 2 bis Sekunde 4 schneiden möchte. Für die Zeitstempel der Start und End zeit jedes Audiosegments nehmen wir die Daten aus der sortierten Ranked segments.json Datei. Diese werden dann als extra Audiofiles abgespeichert und im nächsten Schritt wieder Zusammengesetzt.

Chapter 3

Datenbeschaffung

3.1 Die ARD-Audiothek

Die ARD-Audiothek ist eine AUdiodatenbank der ARD, die öffentlich rechtlich ist und Ihre Inhalte zur Verfügung stellt.

3.1.1 Die ARD-Audiothek API

Die Inhalte in der ARD Audiothek kann man entweder dirkt über die die Webseite erreichen, oder mithilfe einer frei benutzbare GraphQL API abfragen. über diese Schnittstelle bekommt man alle Informationen, wie den Titel einer Episode, die Beschreibungen, die Autoren und auch den Link zu dem mp3 file. Diese Schnittstelle wird verwendet, um später die Audiodatein zu erhalten. Über die API kann auch in einigen Fällen direkt ein Transkript des Audiofiles angefordert werden. Allerdings ist die Transkription meist nicht sehr akkurat. Zum Besipiel steht in der Transkription des Satzes ... TODO Das liegt daran, das die Transkriptionen mithilfe der Tools vom Fraunhofer Instituts erstellt werden, welches vermutlich veraltete Technik benutzt.

Chapter 4

Architektur

In diesem Kapitel werden wir weiter auf die tatsächliche Implementierung des Systems eingehen.

4.1 Anforderungen

Um das Projekt weiter einzuordnen, bietet es sich an, vorab Anforderungen an das System zu stellen. Da das gesamte Projekt als eine Reihe von Mikroservices umgesetzt werden soll, werden folgend für jeden Mikroservice Einzelne Anforderungen gestellt:

Zunächst muss die Datengrundlage geschafft werden. Dafür müssen die Podcasts lö

4.2 Vorgehensweise

Dieses Projekt lässt sich zunächst in mehrere kleinen Projekte unterteilen. Bei einem so großen Projekt lohnt es sich eine Mikroservice Architektur anzustreben, bei der jeder Teil für sich gesehen eine Aufgabe erfüllt und nicht auf der Zuverlässigkeit eines anderen Systems beruht. Diese Methode wird auch divide-and-conquer genannt. Zunächst müssen die Daten gesammelt und aufbereitet werden. Für dieses Projekt bildet die Datengrundlage die Transkripte, bzw. Manuskripte der Podcasts der ARD-Audiothek. In der Audiothek selber gibt es keine Transkripte zu den Podcasts. Für den Podcast „radiowissen“ von bayern2 gibt es auf deren Seite die Manuskripte in PDF Format. Diese sind zwar inhaltlich hochqualitativ, da Sie exakte Wortwahl der Podcasts enthalten, als PDF Format sind sie allerdings schwierig maschinell auszulesen und weiterhin besitzen sie keine Zeitinformationen zu den einzelnen Wörtern. Die Zeitinformationen in Form von Zeitstempeln für jedes Wort sind wichtig, um die Audiofiles der Podcasts später an den richtigen Stellen zuzuschneiden.

Ein anderer Ansatz ergibt sich, wenn man die Podcasts transkribiert. Die Vorteile sind, dass die Transkription auch bei Podcasts funktioniert, für die vorab kein Transkript erstellt wurde, was die Mehrzahl aller Podcasts ausmacht. Außerdem kann man bei einer Transkription auch gleichzeitig die Zeitstempel für jedes Wort extrahieren.

4.3 Podcast zu Text

4.3.1 Fraunhofer

Seit 2015 arbeitet die ARD mit dem Fraunhofer-Institut Institut für Intelligente Analyse- und Informationssysteme (IAIS) zusammen, um „Erschließung von Mediendaten zu forcieren und dabei den Schwerpunkt auf maschinelle Verfahren zu legen“ [1] Ein Teil dieses Projektes bezieht sich auf das Audio-Mining. Das Fraunhofer IAIS entwickelte dafür ein System, welches die Audiodatein transkribiert und dabei „in der kompletten ARD, bei Deutschlandradio sowie im ZDF im Einsatz [ist]“. Leider legt das FraunhoferIAIS nicht offen, welche Technologie es dafür verwendet. Die Transkripte lassen sich allerdings sehr einfach über eine Graphql Schnittstelle abfragen.

4.3.2 Microsoft Translate

Eine weitere Möglichkeit zur Audiotranskription bietet Microsoft Translate. Soweit man einen Microsoft 365 Account besitzt kann man in dem Webinterface von Microsoft word eine Transkriptionsfunktion benutzen. Dafür müssen die Audiofiles zunächst auf Microsoft Onedrive hochgeladen werden und können dann mit einem Klick übersetzt werden. Microsoft Word stellt dann sogar Timestamps zur Verfügung für das ganze Dokument. Die Qualität ist außerdem besser als bei kostenlosen Open-Source Alternativen. Dafür skaliert diese Art der Transkription schlecht für größere Datenmengen, da sämtliche files zunächst bei OneDrive hochgeladen werden müssen und dann jedes File von Hand ausgewählt, in Word eingebunden, transkribiert werden und dann abgespeichert werden müssen.

4.3.3 Whisper

Für die Transkription eignen sich Automatic Speech Recognition Systeme (ASR). Eines der besten kostenlosen ASR Systeme bietet Whisper von OpenAI. Dieses System wurde mit einem maschinellen lernverfahren auf 680 000 Stunden Audiomaterial in verschiedenen Sprachen trainiert. Es ist sehr leistungsstark und kann lokal auf eigener Hardware laufen. [2] [3]

Whisper bietet mehrere verschiedene Modelle zur Transkription an. Es gibt die Modelle tiny, base, small, medium und large. Das Basismodell hat ca. 74 Millionen Parameter und benötigt ca. 1 GB VRAM und ist ca. 16 mal schneller als das large Model. Bei einem Test für die Episode 1968-das-ausnahmejahr aus dem Podcast Radiowissen von br2 schneidet es aber nicht sehr gut ab. Aus dem Wort „Vietnam“ wird „Wirdnam“, aus „Panzer in Prag“ wird „Panzer-Inprac“ und aus „Ohrfeige“ wird „Urfeige“. Der vorgetragene Text wurde dabei ohne Störgeräusche und von einer Person flüssig vorgetragen.

Dagegen bietet das Medium Model von Whisper deutlich bessere Ergebnisse für die selbe Episode. Bei der Transkription konnte kein Fehler festgestellt werden. Allerdings ist der Zeitaufwand durch höhere Rechenleistung immens. Auf einem Macbook Pro 2016 mit einem Intel Core i7 benötigt die Transkription ca. 45 min. pro Episode. Auf einer T4 GPU, wie sie Google kostenlos auf Google Colab zur verfügung stellt, dauert eine Transkription immer noch 3,5 Minuten. Für ca. 1000 Episoden bräuchte man demnach ca. 3500 Minuten, d.h. ca. 58 h.

TODO large ausprobieren, auf den Servern der Uni und mit der Ebu API Eurovox

Das Problem der langen Wartezeiten lässt sich umgehen, wenn wir Cloud computing benutzen, das heißt, das Whisper model nicht auf der lokalen Hardware laufen zu lassen, sondern zum Beispiel auf den Servern von Eurovox. Eurovox ist ein Software tool von der EBU, der Europian Broadcast Union. Sie ist ein zusammenschluss von derzeit 68 Rundfunkanstalten in 56 Staaten Europas, Nordafrikas und Vorderasien mit Sitz in Genf. [4] Das tool Eurovox steht dabei allen Mitgliedern zur Verfügung. Mithilfe dieses Tools kann man Text-to-Speech, Übersetzungen und Speech-to-Text Services über eine UI benutzen. Es gibt sogar die Möglichkeit wärend eines Streams live audio captions zu erzeugen. Für dieses Projekt benutzen wir aber zunächst nur die Translation Funktion. Außerdem verwenden wir die API von Eurovox um später das ganze Projekt auf mehr Episode Transkripte ausweiten zu können. Laut den Entwicklern soll die API auch demnächst open-sourced werden [QUELLE].

Die API stellt, ebenso wie das Tool, eine reihe verschiedener Anbieter zur verfügung, über die wir die Audios transkribieren lassen können.

4.4 Transktript Segment Ranking

4.4.1 Keywordsuche

Eine Möglichkeit bietet sich in der Keywordsuche. Hierbei wird einfach überprüft, ob sich ein Keyword in einem der Dokumente wiederfindet. Ist die möchte ein User beispielsweise einen zusammengeschnittene Podcast Episode über „Zugspitze“, so schaut das System, in welchen

Episodensegmenten das Wort „Zugspitze“ auftaucht, und gibt diese zurück. Schwieriger wird es, wenn die Useranfrage mehrere Wörter beinhaltet. Möchte sich der User über das Thema „Zugspitze wandern“ informieren so müsste zunächst untersucht werden, welche Dokumente beide Worte enthalten, welche nur eines der beiden enthalten. Dann müsste man dementsprechend auch ein Algorithmus entwickeln, der diese dann sinnvoll hierarchisiert.

4.4.2 TF-IDF

Einen solchen Ansatz bietet das TF-IDF Maß (Term Frequency – Inverse Document Frequency). Im Bereich des NLP verwendet man das TF-IDF Maß um zu untersuchen welche Wörter in verschiedenen Dokumenten welche Gewichtung erfahren. Dazu wird zunächst die TF-Matrix, also die Term Frequenzy Matrix berechnet. Hierbei wird erst das Vokabular ermittelt, also die Gesamtheit aller Tokens (Wörter) die es in allen Dokumenten (Transkript Segmenten) des Korpuses (alle heruntergeladenen Episoden) gibt. Dann wird für jedes einzelne Segment die Anzahl jeder in ihm auftretenden Tokens ermittelt. Für den Satz: „Auf der Zugspitze gibt es viele Wanderer, die die Zugspitze lieben“. In diesem Fall würde in der TF Matrix an der Stelle Zugspitze eine 2 Stehen, in der Zeile Wandern aber 0, da zwar das Wort Wanderer, aber nicht das Wort wandern vorkommt. Da diese beiden Worte aber sehr ähnlich sind und der User bei einer Anfrage nicht immer nach verschiedenen Versionen eines Wortes suchen will, um dann ein zufriedenstellendes Audio zu erhalten

Dabei gibt es allerdings keine Möglichkeit, die verschiedenen Treffer dieser suche nach Relevanz zu hierarchisieren. Sucht man zum Beispiel nach dem Stichwort „Klimakrise“ würden dabei mehrere Stunden Material zusammenkommen [QUELLE]. Man könnte nun einfach die Ersten Segmente nehmen, die zusammen die vorgegebene Zeit überbrücken. Allerdings ist dieser Ansatz wenig Vielversprechend.

4.4.3 Embeddings

Ein relativ moderner Ansatz besteht in der Suche mithilfe von Embedding Vektoren. Dabei versuchen wir die Semantik (also die Bedeutung) der einzelnen Segmente Mathematisch mithilfe eines Vektors zu repräsentieren. Dieser Vektor soll dann zur Suche nach Ähnlichkeit dienen. Ein Vektor, der aus dem Satz „Ich sitze auf einer Bank im Grünen“ und „Im Park steht eine alte Bank“ sollen dabei sehr Ähnlich zueinander sein, der Satz „Ich gehe in die Bank und hebe Geld ab“ aber nicht sehr Ähnlich. Hierbei soll erkannt werden, dass sich die ersten beiden Sätze auf eine Bank zum Sitzen in der Natur beziehen und der dritte Satz das Geldinstitut meint. Ein solch filigranes Verständnis der Bedeutung ist nicht einfach zu erreichen. Ein Regelbasierender Algorithmus würde die Unterschiede bei solchen Homonymen in keinem Fall erkennen können. Regelbasierte Algorithmen wie z.B. [QUELLE]

haben auch Probleme mit negierung wie zum Beispiel in „Ich habe nichts gegessen.“ und „Nichts habe ich heute gemacht außer gegessen“.

4.4.4 Cosinus Similarity

Mithilfe der Embedding Vektoren können können wir Sätze finden, die zueinander Ähnlich sind. Aber was bedeutet überhaupt ähnlich? Die Vektoren des BERT Models sind 768 Dimensional, haben also 768 Gleitkommazahlen gespeichert, die zwischen -1 und 1 liegen. Diese Gleitkommazahlen Vektoren könnte man auch als Feature Vektoren begreifen. Zum Beispiel könnte die erste Zahl dieses Vektors für die Erwähnung von Professoren in dem Satz stehen (-1 für keine Professoren; 1 für viele Professoren). Die zweite Zahl könnte für das Thema Essen stehen (-1 für wenig mit Essen zu tun; 1 für sehr viel mit Essen zu tun). Damit hätte der Satz „In der Mensa gibt es jeden Tag Currywurst mit Pommes“ an der ersten Stelle vielleicht eine 0,1, weil der Begriff „Mensa“ leicht mit Uni und Professoren konnotiert wird und die zweite Stelle würde bei 0,94 liegen, da es in dem Satz offensichtlich um das Essen handelt. In der Realität wird das Model sehr wahrscheinlich nicht so für Menschen offensichtlichen Merkmale lernen. Ein Grund dafür ist, dass das Model vor allem pro Eintrag eine linearkombination von verschiedenen Menschenoffensichtlichen Merkmalen lernen wird, also jede Zahl eine überlagerung verschiedener Eigenschaften darstellt. Eine forschungsrichtung, die versucht solche Modelausgaben Menschenlesbar zu gestalten liegt in der Explainable AI

Um die Ähnlichkeit von diesem Satz zu der Frage „“ zu bestimmen nutzen wir die Cosinus distanz als Maß. Es gibt auch die Euclidische Distanz, allerdings gestaltet sich dabei das Problem der Vector Normalisierung.

Diese Komplexen semantischen Unterschiede, oder Gemeinsamkeiten zu erkennen erfordert etwas mehr Raffinesse.

Um einen Embeddingvektor zu erstellen, benutzen wir das BERT Model. BERT, das Akronym für Bidirectional Encoder Representations from Transformer, ist ein Sprachmodel, das 2018 von Google entwickelt, und zur Benutzung freigegeben wurde. BERT ist ein Neuronales Netzwerk mit 12 Schichten, das für zwei verschiedene Aufgaben gleichzeitig trainiert wurde. Zum einen wurde es auf eine Masked Language Modeling Aufgabe und zum Anderen auf eine Next Sentence Prediction trainiert.

Masked Language Modeling Bei dieser Aufgabe soll das Model versuchen, aus dem Kontext eines Satzes ein maskiertes Wort in diesem Satz vorherzusagen. Dafür wird dem Model ein Satz gegeben, in dem zufällige Wörter einfach versteckt werden und das Model soll für diese Wörtereine Vorhersage treffen.

4.5 Audio Zusammensetzung

Um die Audios nun wieder zusammenzusetzen verwenden wir das gleiche Modul Pydub. Es bieten sich mehrere Möglichkeiten an, die Audiosegmente wieder zusammenzusetzen. Man könnte die Segmente einfach ohne Pause hintereinander abspielen. Dabei folgen allerdings mehrere Probleme: Zum einen werden die Audiofiles nicht immer an sehr passenden Stellen getrennt, sodass manchmal mitten im Wort abgebrochen wird und das nächste Segment beginnt. Um dieses Problem zu lösen könnte man die Audiofiles langsam ausfaden lassen.

Außerdem sollte dem Hörer bewusst sein, dass ein Audiosegment aus einer Episode aufhört und das nächste beginnt. Dafür bietet sich ein kurzer Signalton zwischen den einzelnen Audiosegmenten an. Dieser sollte nicht nervig sein, da er dem Hörer öfter vorgespielt wird.

Eine weitere Möglichkeit wäre, zwischen jedem Segment dem Hörer eine kurze Vorstellung der Episode und der Sprecher*in zu ermöglichen oder sogar Kontext zu dieser zu geben.

Die Datengrundlage bilden zunächst einige Podcast-Reihen aus der ARD Audiothek. Diese Audiodateien müssen zuerst transkribiert werden, falls kein Transkript bereits vorliegt. Anschließend werden diese Transkripte mit Zeitstempeln im Audio versehen. Danach müssen die Transkripte einer semantischen Analyse unterzogen werden, wobei Schlagwortextraktion oder Word Embeddings als mögliche Ansätze in Betracht kommen.

Auf Anfrage des Benutzers werden dann die relevantesten Ausschnitte aus den Transkripten herausgesucht und die entsprechenden Audiodateien herausgeschnitten. Ein weiteres System ist dafür verantwortlich, diese Dateien nach ihrer Bedeutung oder inhaltlichen Nähe zu sortieren und sicherzustellen, dass der zuvor festgelegte Zeitrahmen eingehalten wird. Die Audiodateien werden dann zusammengeführt und dem Benutzer über eine benutzerfreundliche Schnittstelle als Stream zum Abspielen angeboten.

Der Hauptteil dieser Arbeit liegt in der semantischen Analyse der Audiotranskripte und der sorgfältigen Auswahl passender Abschnitte zum gewählten Thema. Dies soll durch den Einsatz von Embeddings oder einer Schlagwortsuche realisiert werden. Für die Verwendung von Embeddings könnten maschinelle Lernverfahren, wie beispielsweise das Google Bert, herangezogen werden. Eine zusätzliche Herausforderung besteht darin, die Audioausschnitte entsprechend zuzuschneiden, sodass Sätze sauber getrennt werden und die Übergänge zwischen den verschiedenen Audioausschnitten nicht zu abrupt sind. Dadurch wird gewährleistet, dass der Benutzer nicht zu stark zwischen verschiedenen Themenaspekten hin und her geworfen wird.

[Bart] [Choi 18] [Clar 20] [Du 17] [Jone 21] [Kang 12] [Karp 20] [Laba 22] [Loch 18] [Maro 20] [Mold 00] [Zhan 20] [**?**]

Chapter 5

Evaluation verschiedener semantischer Verfahren

5.1 BERT Embeddings

BERT (Bidirectional Encoder Representations from Transformers) [**?**] ist ein auf NLP Aufgaben spezialisierter Transformer. Er wurde von Google 2019 entwickelt und war seinerzeit das Beste Modell um verschiedene Aufgaben im Bereich des NLP zu lösen. Unter diesen Aufgaben befinden sich Next Sentence Prediction (NSP) und Masked Language Modeling, auf das es auch trainiert wurde. Außerdem kann man das Modell mit wenig aufwand auf andere Aufgaben finetunen und damit sehr gute Ergebnisse im Bereich der Named Entity Recognition, der Sentiment Analyse oder

5.2 Sentence Transformer Embeddings

Das Sentence Transformer Projekt baut auf der Architektur von BERT auf. Es wird auch SBERT für Sentence BERT genannt. Unter dem

5.3 LLama2 Embeddings

5.4 Ranking

Chapter 6

Ausblick

In diesem Kapitel wird ein Ausblick über verschiedene Möglichkeiten der Verbesserung des Systems erläutert.

Bibliography

[Bart] M. Barthel, A. Mitchell, D. Asare-Marfo, and C. Kennedy. "Measuring News Consumption in a Digital Era".

[Choi 18] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer. "QuAC : Question Answering in Context". Aug. 2018.

[Clar 20] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators". March 2020.

[Du 17] X. Du, J. Shao, and C. Cardie. "Learning to Ask: Neural Question Generation for Reading Comprehension". Apr. 2017.

[Jone 21] R. Jones, B. Carterette, A. Clifton, M. Eskevich, G. J. F. Jones, J. Karlgren, A. Pappu, S. Reddy, and Y. Yu. "TREC 2020 Podcasts Track Overview". March 2021.

[Kang 12] M. Kang and U. Gretzel. "Effects of Podcast Tours on Tourist Experiences in a National Park". *Tourism Management*, Vol. 33, No. 2, pp. 440–455, Apr. 2012.

[Karp 20] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. "Dense Passage Retrieval for Open-Domain Question Answering". Sep. 2020.

[Laba 22] P. Laban, E. Ye, S. Korlakunta, J. Canny, and M. Hearst. "NewsPod: Automatic and Interactive News Podcasts". In: *27th International Conference on Intelligent User Interfaces*, pp. 691–706, ACM, Helsinki Finland, March 2022.

[Loch 18] M. Lochrie, R. De-Neef, J. Mills, and J. Davenport. "Designing Immersive Audio Experiences for News and Information in the Internet of Things Using Text-to-Speech Objects". In: *Proceedings of the 32nd International BCS Human Computer Interaction Conference*, 2018.

[Maro 20] D. Maroni, J. Köhler, J. Fisseler, and S. Becker. "KÜNSTLICHE INTELLIGENZ IM PRODUKTIVEN EINSATZ FÜR DIE AUTOMATISIERTE ERSCHLIESSUNG IM MULTIMEDIALEN PRODUKTIONSPROZESS". 2020.

[Mold 00] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus. "The Structure and Performance of an Open-Domain Question Answering System". In: *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 563–570, Association for Computational Linguistics, Hong Kong, Oct. 2000.

[Zhan 20] J. Zhang, Y. Zhao, M. Saleh, and P. Liu. "PEGASUS". In: *Proceedings of the 37th International Conference on Machine Learning*, pp. 11328–11339, PMLR, Nov. 2020.

Anhang

Hier sind noch ein paar Zusatzinformationen.

Glossary

library A suite of reusable code inside of a programming language for software development. i

shell Terminal of a Linux/Unix system for entering commands. i