KI-Projekt “Einfach Erklärt” Midterm Review

KIP bei Herr Dr. Hußlein im SoSe 24

Lars Specht, Ben Reher, Simon Eiber und Felix Wippich, 07.05.24

1. Einleitung (Ben)
2. Allgemein
    1. Projektmanagement und organisatorische Herausforderungen (Felix)
    2. Überblick (alle)
    3. Kommunikation mit Nachrichtenquellen (Ben)
3. Entwicklung
    1. Scraper (Lars)
    2. MDR Scraper (Lars)
    3. Deutschlandradio (Simon)
    4. Datenstruktur und Datahandler (Felix)
    5. Weitere Nachrichtenangebote (Simon)
    6. KI-Server (Ben)
    7. Technische Herausforderungen (Simon)
4. Ergebnisse
    1. Gesamelte Daten
    2. Erkenntnisse über die Daten
    3. Matcher
    4. Dokumentation zur Übergabe
5. Fazit (alle)

1. Einleitung

Die Umsetzung eines Projekts ist oft von technischen und organisatorischen Herausforderungen geprägt. In diesem Projektbericht werden sowohl technische als auch organisatorische Aspekte beleuchtet, die bisher bewältigt wurden. Darunter sind, die remote Durchführung von Meetings sowie die Kommunikation mit Nachrichtenquellen wie Deutschlandfunk (DLF), Nachrichtenleicht (NL) und Mitteldeutscher Rundfunk (MDR). Zudem wird die Datenstruktur für die Speicherung und das Scraping erläutert sowie ein Ausblick auf den Matching-Prozess gegeben, der für die Nutzung des aufgebauten Datensatzes von zentraler Bedeutung ist. Zuletzt geben wir einen kleinen Ausblick auf den weiteren Verlauf des Projekts.

2. Allgemein

2.1. Projektmanagement und organisatorische Herausforderungen

Neben technischen Herausforderungen in der Umsetzung des Projekts stellten sich auch einige zusätzliche organisatorische Herausforderungen.

Durch die nach dem Projektstart erfolgte Zuteilung eines Teammitglieds in das Team, war eine schnelle Kontaktaufnahme und Integration des neuen Teammit-

glieds notwendig. Dies erforderte eine schnelle Einarbeitung und Anpassung der Teamdynamik. In Folge sollten auch die wöchentlichen Meetings remote absolviert werden.

Trotz dieser zusätzlichen Schwierigkeiten wurden alle Herausforderungen bisher hervorragend bewältigt. Um die Ressourcen für alle Teammitglieder ständig bereitzuhalten, war bereits ein Repository auf GitHub angelegt, dieses wurde um eine Datei `Meetings.md` zur Protokollierung der in den remote Meetings besprochenen Inhalte erweitert. Die Meetings wurden fest wöchentlich und zusätzlich nach Bedarf angesetzt und finden virtuell über Zoom statt, des Weiteren ist ein ständiger Kommunikationskanal zum Austausch über WhatsApp, für kurzfristige Änderungen oder dringende Probleme, verfügbar. Neben der bekannten Herausforderung von Videokonferenzen, bietet Zoom die Chance, den Bildschirm für "Code-Reviews" und "Code-Vorstellungen" zu teilen.

2.2. Überblick

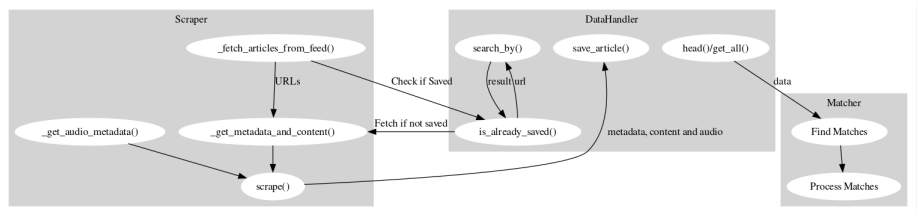


Figure 1: Die Pipeline

Die Abbildung 1 stellt den aktuellen Stand der Pipeline des Projekts dar. Unterteilt werden kann in die Scraper, den DataHandler und die Matcher, die alle miteinander interagieren. Die Scraper sammeln für die jeweilige Nachrichtenquelle Artikel, die der DataHandler dann in ein einheitliches Format bringt und mit ihrem Inhalt und den Metadaten speichert. Der Matcher vergleicht leichte und normale Artikel und versucht diese zu verbinden. Details zu den einzelnen Modulen finden sich in den jeweiligen Kapiteln.

3. Aktueller Stand

3.1. Kommunikation mit Nachrichtenquellen

Die Kommunikation mit den Nachrichtenquellen wurde ein Bestandteil unseres Projekts. Wir wandten uns an DLF, NL und MDR, um Zugang zu ihren Nachrichteninhalten zu erhalten, da wir bereits bei DLF auf Schwierigkeiten beim Scrapen historischer Daten gestoßen sind und es als eine alternative Möglichkeit gesehen haben, dort direkt anzufragen. Es ist hier deutlich schwerer an die Daten zu kommen, da einerseits viel mehr publiziert wird im Vergleich zu NL und es keine einfach abrufbare API gibt. Über verschiedene Kanäle wie E-Mail, Instagram, TikTok und LinkedIn versuchten wir, Kontakt herzustellen.

Nach mehreren Versuchen erhielten wir zuerst eine Antwort auf TikTok, dass es an das Team von NL weitergeleitet wurde. Einen Tag später kam dann eine Antwort von Herrn Bertolaso, einem leitenden Nachrichtenredakteur bei DLF. Er leitete unsere Anfrage weiter an Frau Gnad, und wir befinden uns derzeit in der Warteposition, in der Hoffnung auf weitere Unterstützung. Nach einem Telefonat mit Frau Gnad stellte sich heraus, dass noch die Möglichkeit besteht tagesaktuelle Daten aus den Instagram Captions von NL zu scrapen. Des Weiteren hat sie uns mit dem Archiv in Verbindung gesetzt. Die Mitarbeitenden werden sich da in den nächsten Tagen bei uns melden, ob uns geholfen werden kann. Da MDR als Quelle erst letzte Woche dazu kam, ist hier leider noch kein Erfolgserlebnis zu verzeichnen, da bis jetzt nur Antworten kamen, dass es an die zuständige Redaktion weitergeleitet wurde.

Wir erwogen auch eine Zusammenarbeit mit der anderen Gruppe, die das gleiche Projekt durchführt, nachdem sich in einem Gespräch mit Prof. Baumann herausstellte, dass es sinnvoll sein könnte, sich beim Scrapen die Arbeit zu teilen. Jedoch wurde unsere Anfrage abgelehnt, da die andere Gruppe befürchtete, dass eine Auslagerung des Webscrapings zu einem Verlust in der Bewertung führen könnte, da dies ja auch Teil der Aufgabenstellung ist und auch einen gewissen Teil des Arbeitsaufwandes darstellt.

3.2 Scraper

Was ist ein Scraper? Ein Scraper ist ein Programm, das automatisch Daten von Webseiten extrahiert. Es gibt verschiedene Arten von Scrapern, die sich in ihrer Funktionsweise und ihren Anwendungsmöglichkeiten unterscheiden. Wir verwenden zwei Scraper-Bibliotheken: BeautifulSoup und Selenium. Diese unterscheiden sich in ihren Funktionen und Anwendungsbereichen.

BeautifulSoup ist eine Bibliothek, die es ermöglicht, Daten aus HTML- und XML-Dateien zu extrahieren. Sie ist jedoch nicht in der Lage, Formulare zu bearbeiten oder JavaScript auszuführen, weshalb sie nur für statische Webseiten geeignet ist. Das bedeutet, dass sie lediglich den HTML-Code der Webseite auslesen kann und nicht die dynamischen Inhalte, die durch JavaScript generiert werden (zum Beispiel nach dem Drücken eines Buttons).

Selenium ist ein Webdriver, der es ermöglicht, Webseiten zu steuern und mit ihnen zu interagieren. Ein Webdriver ist ein Programm, das die Steuerung eines Webbrowsers ermöglicht, also tatsächlich ein Browserfenster öffnet und dieses dann steuert. Selenium kann auch dynamische Webseiten auslesen, da es JavaScript ausführen kann. Selenium ist langsamer als BeautifulSoup und benötigt auch mehr Rechenleistung, deshalb wird BeautifulSoup bevorzugt, wenn es möglich ist.

3.2.4. Arten von Scrapern

Für unser Projekt benötigen wir zwei Arten von Scrapern:

Historische Scraper sammeln die Artikel, die in der Vergangenheit auf den Webseiten veröffentlicht wurden. Diese lassen wir einmalig laufen, um das Archiv der Webseiten zu erstellen.

Aktuelle Scraper werden regelmäßig ausgeführt, um kontinuierlich die neuesten Artikel von den Webseiten zu extrahieren.

3.2.5. Der `BaseScraper`

Für die Scraper haben wir eine Basisklasse `BaseScraper` erstellt, die die allgemeinen Funktionen und Methoden enthält, die für alle Scraper benötigt werden. Die Basisklasse enthält die folgenden Methoden:

- `base_metadata_dict()`: Gibt ein Dictionary zurück, das die Metadaten enthält, die wir für jeden Artikel speichern wollen (zum Beispiel Titel, Beschreibung, Datum, usw.).
- `_get_soup()`: Lädt die Webseite und gibt ein BeautifulSoup Objekt zurück.
- `_fetch_articles_from_feed()`: Extrahiert die Artikel-URLs aus dem Feed (oder auch anders, falls kein Feed verfügbar) der Webseite.
- `_get_metadata_and_content(url)`: Extrahiert die Metadaten und den Inhalt eines Artikels aus der URL.
- `scrape()`: Führt die einzelnen Schritte zum Scrapen der Webseite aus.

Dadurch müssen wir für jeden neuen Scraper nur noch die spezifischen Methoden implementieren, die für die jeweilige Webseite benötigt werden, und das Ganze ist ein wenig übersichtlicher.

3.3. MDR Scraper

Warum MDR? Der MDR Scraper extrahiert Daten von der Webseite des MDR. Der MDR ist ein öffentlich-rechtlicher Rundfunksender, der für Sachsen, Sachsen-Anhalt und Thüringen schreibt und eine Vielzahl von Inhalten, darunter Nachrichten, Videos, Audios und mehr anbietet. Wir haben uns aus verschiedenen Gründen für den MDR als eine der Webseiten entschieden, von der wir Daten extrahieren wollen:

- Der MDR ist ein öffentlich-rechtlicher Sender, was eine gewisse Qualität der Daten sichert.
- Der MDR bietet eine Vielzahl von Inhalten.
- Das Angebot in einfacher Sprache des MDR und die normalen MDR-Nachrichten sind unter einem Dach, was bedeutet, dass es zu jedem leichten Artikel auch einen normalen Artikel gibt.

- Die leichten Artikel verlinken immer direkt den normalen Artikel, weswegen wir hierfür keinen Matcher brauchen.
- Der MDR bietet Audios an, die von Menschen eingesprochen wurden.
- Der MDR lädt wöchentlich recht viele leichte Artikel hoch, was uns eine gute Datenbasis bietet (ca. 22 Artikel pro Woche).

Funktionsweise Der MDR Scraper extrahiert zunächst Daten vom Angebot in einfacher Sprache des MDR und lädt die Audios herunter. An die Audios zu kommen war nicht ganz einfach, da der MDR die Audios nicht direkt verlinkt, sondern sie über eine JavaScript-Datei lädt. Daher muss hier Selenium verwendet werden. Da die einfachen Artikel immer auf die normalen Artikel verlinken, benötigen wir hierfür keinen Matcher. Im Anschluss an den einfachen Artikel wird der normale Artikel gescraped.

3.4 Deutschlandradio

Deutschlandradio ist ein Bestandteil des öffentlich-rechtlichen Rundfunks und verantwortlich für die Produktion verschiedener Nachrichtenangebote, darunter DLF und NL, die als bedeutende Datenquellen dienen. Eigenständige Redaktionen sind dafür zuständig, die jeweiligen Inhalte zu konzipieren und zu veröffentlichen.

Die Gründe für die Berücksichtigung von Deutschlandradio sind divers:

- Es ist Teil des öffentlich-rechtlichen Rundfunks.
- Die Reichweite erstreckt sich über ganz Deutschland.
- Es bietet eine breite inhaltliche Abdeckung aktueller Themen, darunter Politik, Wirtschaft, Wissenschaft, Gesellschaft und Kultur.
- Es besteht eine redaktionelle Nähe zwischen DLF und NL.
- Die Inhalte werden von eigenständigen Redaktionen erstellt, in Form von Artikeln und Audios.

Nachrichtenleicht Auf der Internetseite von NL werden jeden Freitagnachmittag etwa fünf bis sechs Artikel in leicht verständlicher Sprache veröffentlicht. Zusätzlich werden die Artikel häufig als Audio angeboten, die von menschlichen Sprechern eingesprochen werden.

Scraping Dank der redaktionellen Nähe zwischen DLF und NL sind die Internetseiten größtenteils strukturell identisch aufgebaut. Dadurch konnte ein `DeutschlandradioScraper` basierend auf dem BaseScraper entwickelt werden, um Redundanzen zu vermeiden. Die Unterschiede liegen hauptsächlich in den Metadaten und der Verfügbarkeit von Audio bei NL-Artikeln. Ausgehend vom `DeutschlandradioScraper` konnten entsprechende Scraper für DLF und NL abgeleitet werden. Für den NL-Feed wurde eine API-Schnittstelle gefunden, die das Scrapen erleichtert.

3.5. Datenstruktur

Auf Anraten von Professor Baumann wird für die Speicherung keine SQL-Datenbank benutzt, sondern wie in der Abbildung dargestellt eine Ordnerstruktur.

```
|-- data
|   |-- DLF
|   |   |-- easy
|   |   |   |-- 2024-03-15-Bundes-Wehr_beteiligt_sich_an_[...]
|   |   |   |   |-- audio.mp3
|   |   |   |   |-- content.txt
|   |   |   |   |-- metadata.json
|   |   |   |   |-- raw.html
|   |   |   |-- lookup_DLF_easy.csv
|   |   |-- hard
|   |   |   |-- 2024-04-25-Angebliche_Drohnenangriffe_[...]
|   |   |   |   |-- content.txt
|   |   |   |   |-- metadata.json
|   |   |   |   |-- raw.html
|   |   |   |-- lookup_DLF_hard.csv
|   |   |-- matches_DLF.csv
```

Figure 2: Struktur des zur Speicherung genutzten Dateisystems am Beispiel von DLF (hard) und NL (easy) (reduziert auf jeweils einen Artikel)

Für jede Nachrichtenquelle findet sich im `data` Verzeichnis ein Unterordner. Da das Matchen lediglich innerhalb derselben Quelle (also z.B. nur DLF zu NL, nicht DLF zu MDR) stattfinden soll, findet sich die Datei mit den jeweiligen Matches (`matches_<Nachrichtenquelle>.csv`) auf dieser Ebene (z.B. DLF). Jedes der Unterverzeichnisse ist wiederrum aufgeteilt in die Ordner `easy` und `hard`, wobei easy die Nachrichten in leichter Sprache enthält und hard die Nachrichten in Standardsprache. Hier findet sich für jeden gespeicherten Artikel ein eigener Ordner mit der Benennungsstruktur `<Jahr>-<Monat>-<Tag>_<Titel>`. Im Ordner zum jeweiligen Artikel findet sich jeweils `content.txt`, der Haupttext des Artikels, `metadata.json`, der Verschiedene Metadaten wie URL, Autor und Datum in einem über alle Nachrichtenquellen standardisierten JSON-Format enthält, sowie `audio.mp3`, falls der Artikel als vorgelesene Version als Audio verfügbar ist.

Für eine effiziente Suche der Artikel nach ihren jeweiligen Links ist jeweils eine sogenannte `lookup-<Nachrichtenquelle>-<easy|hard>.csv` implementiert. In diesem wird für jeden gespeicherten Artikel jeweils der Dateipfad und der Link im CSV-Format abgespeichert. Die URL des Artikels wird auch in den Metadaten gespeichert, dennoch entstand die Idee des redundanten Speicherns um für die Suche nach der URL nicht über das ganze Verzeichnis iterieren, sondern lediglich eine CSV-Datei analysieren zu müssen. Besonders bei großen Datenmengen ist so eine bessere Effizienz erhofft. Es ist zu erwarten, dass dieser Unterschied bei

größeren Datenmengen noch deutlicher wird.

```
'''Getestet wurde die Suche über den Link des Artikels
sowie den Titel. Die Funktion sucht bei der url automatisch
im Lookuptable ansonsten iteriert sie über das Unterverzeichnis.
Das erste Argument e steht dafür, dass das easy Unterverzeichnis
durchsucht werden soll'''
dh.search_by("e", "url", "https://www.DLF.de/ \
zahl-der-arbeitslosen-sinkt-im-april-um-20-102.html")
dh.search_by("e", "title", "Zahl der Arbeitslosen sinkt \
im April um 20.000")
# Output
"Time taken for search_by url: 0.0020 seconds"
"Time taken for search_by title: 0.0071 seconds"
```

Die Speicherung im eigenen Format bietet viel Flexibilität und Unabhängigkeit von Versionen eines Datenbankmanagementsystems. Allerdings stellt sich die Herausforderung eines komfortablen, einheitlichen und effizienten Zugriffs auf die Daten.

Der DataHandler übernimt diese Tolle. Er bietet ein Interface für den **Zugriff auf die Daten durch Funktionen wiehead, welcher die ersten n Artikel als Pandas DataFrame zurückgibt. Des Weiteren soll eine einheitliche **Speicherung** durch vordefinierte Speicherfunktionen sichergestellt werden. Auch ermöglicht der DataHandler eine **Suche** im Verzeichnis nach Metadaten. Um keine Artikel doppelt zu Scrapen gibt es außerdem die Funktionis_already_saved, welche sich die bessere Sucheffizienz der Lookuptable zunutze macht. Sie gibt zurück, ob die URL bereits gescraped und gesaved wurde. Das DataHandler Objekt muss mit der jeweiligen Nachrichtenquelle initialisiert werden (aktuell“dlf“, oder“mdr“) und kann dann für das jeweilige Unterverzeichnis genutzt werden. Die Initialisierung mit der Nachrichtenquelle soll unter anderem einer Vermischung der Daten vorbeugen. Den meisten Funktionen muss übergeben werden, ob im“hard“(“h“), oder“easy“(“e“‘) Unterverzeichnis gelesen oder geschrieben werden soll.**

Entwicklung und Debugging Als zentrales Modul für die Speicherung und den Zugriff auf die Daten war es wichtig sicherzustellen, dass der DataHandler zuverlässig und effizient arbeitet. Viele Features wurden von Anfang an (per design) angelegt, um die Effizienz des DataHandlers zu optimieren und direkt die meisten Funktionen bereit zu stellen. Dazu gehörte zum Beispiel der Lookup Table, um eine effiziente Suche zu ermöglichen. Durch die Zentralität des Moduls und die Schwierigkeit das Modul zu testen ohne, dass größere Datenmengen verfügbar waren, entwickelte sich die Robustheit des Moduls mit der Entwicklung der der anderen Module, in denen er Anwendung fand. Hier wurden des Öfteren

Issues zurückgemeldet, die gefixt werden mussten. Da der DataHandler mit dem Dateisystem arbeitet und unter Windows 11 entwickelt wurde, traten im Verlauf vor allem Fehler auf, die auf den Unterschied zwischen Linux und Windows zurückzuführen waren. Diese konnten behoben werden, indem die Pfade in den Funktionen angepasst wurden. Ziel ist es, dass der DataHandler sowohl auf Windows als auch auf Linux lauffähig ist. Auch einige kleinere Erweiterungen wurden vorgenommen, um weitere Funktionen bereit zu stellen, beispielsweise das Abrufen der Audios. Im Laufe der Entwicklung fand außerdem ein großes Refactoring statt, um den Code übersichtlicher und wartbarer zu machen. Hierfür wurden alle Funktionen die nicht direkt als Interface für den Nutzer gedacht waren, in eine eigene Klasse `DataHandlerHelper` ausgelagert.

Examples Da der Datahandler, wie das gesamte Projekt so angelegt ist, der er zur Weiterentwicklung und Forschung dienen kann, wurden ein Notebook mit Beispielen erstellt indem die Funktionen des DataHandlers demonstriert werden. Ein Weiterer Fokus lag, in der ausführlichen Dokumentation der Klasse und Funktionen durch Python Docstrings.

3.6 Weitere Nachrichtenangebote

Zur Auswahl standen auch die Nachrichtenangebote der APA (Austria Presse Agentur), des NDR und des SR. Die APA bietet Nachrichten in leichter Sprache an, die von capito.ai generiert werden, einem vollautomatisierten KI-Tool zur Übersetzung von Texten aus der Standardsprache in leicht verständliche Sprache. Auf ihrer Webseite sind die Originalartikel sowie Übersetzungen in die Sprachniveaus B1 und A2 verfügbar. Wir haben uns gegen die Verwendung der APA als Datenquelle entschieden, da keine Audioversionen der Artikel vorhanden sind und die Artikel in leichter Sprache ausschließlich von KI generiert werden, was zu einem möglichen Bias in den Daten führen könnte. Darüber hinaus erreichen die Artikel nicht die qualitativen Standards der öffentlich-rechtlichen Sender. Sowohl der NDR als auch der SR sind Mitglieder des öffentlich-rechtlichen Rundfunkverbunds ARD und bieten ebenfalls Nachrichten in leichter Sprache mit Audio an. Allerdings unterscheiden sich die Texte in leichter Sprache formell stark von denen des Deutschlandradios und des MDR. Zudem gibt es keinen separaten Nachrichtenfeed, was das Scrapen der Artikel erschwert. Das Angebot des SR in leichter Sprache konzentriert sich hauptsächlich auf regionale Nachrichten aus dem Saarland. Aus diesen Gründen haben wir vorläufig beschlossen, diese Nachrichtenangebote nicht zu berücksichtigen.

3.7. KI-Server

Die Datenspeicherung und das Scraping (später auch das Matching der Artikel) finden über den KI-Server der OTH statt. Dies war von Anfang an die Idee, da dieser eine hohe Rechenleistung bietet und somit das Scraping und unsere kommenden Schritte schneller und effizienter gestaltet. Seit kurzem existiert auch die Ordnerstruktur und die Daten werden automatisiert gespeichert.

Mithilfe eines Cronjobs werden die Scraping-Skripte regelmäßig ausgeführt, um die neuesten Artikel zu speichern. Wie bereits erwähnt, veröffentlicht NL wöchentlich neue Artikel und DLF, sowie MDR, täglich. Demnach wird es zwei Skripte geben. Eines, das einmal am Tag ausgeführt wird und die neuen Artikel von NL speichert und eines, das alle zwei Stunden ausgeführt wird und die neuen Artikel von DLF und MDR speichert oder aktualisiert. Sollten wir mit diesen noch experimentellen Zeiträumen für die Ausführung der Skripte auf Probleme stoßen, werden wir diese gerade am Anfang der Scraping-Phase natürlich auch noch anpassen.

3.8. Technische Herausforderungen

Scraper Webseiten werden kontinuierlich aktualisiert und verbessert, was zu Änderungen in der HTML-Struktur und den CSS-Klassen führen kann, die für das Scraping verwendet werden. Diese Änderungen können dazu führen, dass die Scraping-Skripte nicht mehr ordnungsgemäß funktionieren, da sie nicht mehr in der Lage sind, die benötigten Informationen korrekt zu extrahieren. Dies erfordert eine regelmäßige Überwachung der Webseite sowie eine kontinuierliche Aktualisierung der Scraping-Skripte, um sicherzustellen, dass sie weiterhin effektiv arbeiten.

Skalierbarkeit Bei der Verarbeitung großer Datenmengen, wie beispielsweise beim Matching, kann die Leistung des DataHandlers stark beeinträchtigt werden. Im Gegensatz zu eigenständigen Datenbanksystemen ist er nicht speziell für die Bewältigung solcher Datenmengen optimiert. Dies kann zu längeren Verarbeitungszeiten, erhöhtem Ressourcenverbrauch und potenziell anderen unbekannten Problemen führen. Eine der Hauptlimitationen liegt in der Ordnerstruktur der Daten, da bei Operationen auf den Daten alle Ordner der Artikel durchlaufen werden müssen.

Es ist daher entscheidend, den DataHandler gegebenenfalls entsprechend zu optimieren. Dies kann durch verschiedene Maßnahmen erfolgen, darunter Parallelisierung, Optimierung der Datenstrukturen oder die Nutzung eines dedizierten Datenbanksystems. Durch diese Optimierungen kann die Leistungsfähigkeit des DataHandlers verbessert und die Effizienz bei der Verarbeitung großer Datenmengen gesteigert werden.

Historische Artikel Das Scrapen historischer Artikel birgt seine eigenen Herausforderungen, insbesondere in Bezug auf die Zugänglichkeit und Verfügbarkeit der URLs zu den Artikeln sowohl beim DLF als auch beim MDR. Zusätzlich dazu besteht die Schwierigkeit, die Konsistenz der gesammelten Daten sicherzustellen, da Artikel nur einmal gescraped werden und zukünftige Änderungen der Redaktionen an den Artikeln nicht überprüft und aktualisiert werden. Eine effektive Lösung hierfür könnte eine Funktionalität im DataHandler sein, die nicht nur das Datum des Scrapings berücksichtigt, sondern auch eine Versionierung der Artikel implementiert. Dadurch ließe sich diese Problematik beheben, da verschiedene

Versionen eines Artikels zur Verfügung stehen und Änderungen der Redaktionen verfolgt werden könnten.

3.8. Matcher

Matcher sind toll. #FIXME:

3.9 TF-IDF

Für das Matching der Artikel wurde das Tf-idf-Maß (Term Frequency - Inverse Document Frequency) verwendet, ein weit verbreitetes Verfahren im Bereich der Informationsretrieval und Textanalyse. Der Prozess umfasst folgende Schritte:

1. Vektorisierung des Artikels
2. Transformation in die Tf-idf Darstellung
3. Vergleich der Artikel-Vektoren mit Cosine-Similarity
4. Evaluation des Matchers mit zusätzlichen Kriterien

Zur Vektorisierung wurde eine Klasse auf Basis der sklearn API entwickelt, um den Tokenisierungsprozess vollständig zu kontrollieren. Der Article Vectorizer arbeitet ähnlich wie der CountVectorizer. In der .fit()-Funktion wird das Vokabular aus dem Corpus erstellt und in eine Häufigkeitsmatrix umgewandelt. Spezifische Preprocessing-Schritte und Tokenisierung für Nachrichtenartikel umfassen:

- Berücksichtigung von n-grams: Einbeziehung von Wortgruppen unterschiedlicher Länge
- Kombination segmentierter Wörter: Zusammenführung für die leichte Sprache typischerweise segmentierter Wörter
- Extraktion von Substantiven und Eigennamen naives named entity recognition
- Einheitliche Kleinschreibung: Umwandlung aller Wörter Kleinbuchstaben am Ende aller Preprocessing-Schritte
- Ein- und Ausschluss von Zahlen

Eine sklearn Pipeline wurde genutzt, um die vektorisierten Artikel mit dem TfidfTransformer zu verarbeiten. Die Tf-idf-Matrix des gesamten Korpus wurde mittels Cosine Similarity bewertet, und der Artikel mit dem höchsten Score wird als Matches identifiziert.

Der aktuelle Stand erlaubt die Definition eines Matchers, der das Preprocessing durch den Article Vectorizer und das Matching zwischen leichten und schweren Artikeln ermöglicht. Weitere artikelbezogene Matching-Kriterien könnten implementiert werden:

- Berücksichtigung des Veröffentlichungsdatums
  - Maximale Differenz
  - time-decay in Evaluation
- Zeitrahmen des Korpus Einschränken des Zeitraums, aus dem Artikel stammen

- Vokabularbeschränkung (z.B. nur NL- bzw. DLF-Artikel)
- Kombination aus Titel, Teaser, Beschreibung und Inhalt
- Berücksichtigung der Platzierung im Ranking
  - Auswahl aus den Score-Plätzen

Der zeitliche Rahmen des Projekts ermöglichte leider nicht die vollständige Entwicklung des Matchers. Zukünftig könnte der Matcher durch eine Ensemble-Methode verbessert werden. Es wäre sinnvoll, den Article Vectorizer mit verschiedenen Parametern und Datensätzen auf die zu matchenden Artikel anzuwenden und durch ein Voting-System den "passenden" Artikel auszuwählen.

Ein lernbarer Zusammenhang zwischen den Parameterkonfigurationen und der Genauigkeit der einzelnen Matcher könnte hergestellt werden. Ein naiver Ansatz wäre Soft Voting, aber auch lineare Regression könnte als Evaluationsmethode dienen. Zum Trainieren eines solchen Modells könnte der bereits gematchte Datensatz der MDR-Artikel verwendet werden, da hier eine bijektive Zuweisung besteht.

Zusammengefasst bietet dieser Ansatz eine flexible und anpassbare Methode zur Artikelverarbeitung und -matching, die durch weitere Verfeinerungen und die Implementierung zusätzlicher Kriterien noch präziser und leistungsfähiger gemacht werden kann.

3.10. Dokumentation

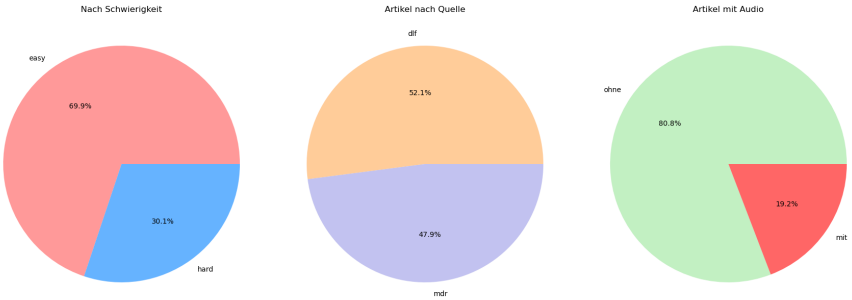
Wie bereits erwähnt ist auf Wunsch von Prof. Baumann Ziel des gesamten Projekts, dass es als Grundlage für weitere Forschung beispielsweise eine Bachelorarbeit dienen kann. Dies wurde nicht nur beim Aufbau berücksichtigt, sondern besonders auch in der Dokumentation. Die Dokumentation nimmt deswegen bei diesem Projekt einen wichtigen Stellenwert ein. Sicher sind viele Ergebnisse auch im Bericht verarbeitet, es ist aber unklar in welchem Umfang dieser in Zukunft zur Verfügung stehen wird. Deshalb enthält das Repository im `README.md` eine Art kurzen Developer Guide. Hier wird nicht nur der Ursprung, der ganz grobe Aufbau dokumentiert, sondern auch wichtige Hinweise die sich zum Beispiel auch im Bericht finden wie die Datenstruktur. Da die Scraper darauf angelegt sind regelmäßig auf einem Server aufzuführen, um stets neue Daten zu generieren, findet sich hier auch eine Tabelle, die die Executables der Scraper kurz beschreibt und einen Hinweis gibt in welchem Intervall sich eine Ausführung anbietet. Für den DataHandler wurde wie bereits erwähnt ein Beispiele Notebook erstellt, in dem die Funktionen des DataHandlers demonstriert werden.

4. Ergebnisse

4.1. Gesamelte Daten

Da ein Hauptziel des Berichts war Daten zu sammeln findet an dieser Stelle ein Auswertung des gewonnen Datenmaterials anhand der Features statt. Durch Matching und Analyse der Daten konnte weit-

ere Erkenntnisse über die Beschaffenheit der Daten gewonnen werden. Diese sind in 4.2. Erkenntnisse über die Daten zusammengefasst.



4.2. Erkenntnisse über die Daten

Nicht nur die Quantitative Analyse der Daten, sondern auch die Qualitative Analyse der Daten ist von Bedeutung. Dies spielte zum Beispiel bei der Forschun an einem geeigneten Matching verfahren eine Rolle.

Unterschiedlichkeit in der Wortwahl

4.3. Matcher

c

5. Fazit

TODO: Fazit schreiben