Grzegorz Siwiecki
6316248
Bachelor of Science
Informatik
8. Fachsemester
s9888566@stud.uni-frankfurt.de

Bachelorarbeit

TwitterPooler: Ein System für das Management, Pooling und die NLP-Vorverarbeitung von Twitter-Daten

Grzegorz Siwiecki

Abgabedatum: 07.04.2022

Lehrgebiet Texttechnologie
der Goethe-Universität Frankfurt am Main
Prof. Dr. Alexander Mehler

Zusammenfassung

Im Rahmen dieser Bachelorarbeit soll ein System namens 'TwitterPooler' für das Management von Twitter-Daten entwickelt werden. Die Anwendung sollte auch in der Lage sein, NLP-Vorverarbeitung und Pooling an heruntergeladenen Daten durchzuführen. Diese Arbeit enthält Informationen über die Implementierung und Evaluierung der Software.Die Untersuchung konzentrierte sich auf die Leistung und die Auswirkungen zusätzlicher Funktionen auf die Performance. Die Forschungsergebnisse weisen auf einen besonders großen Einfluss von Maßnahmen hin, die zur Vorbereitung von Tweets auf NLP-Prozesse notwendig sind. Die erstellte Anwendung bietet innovative Lösungen und eine grafische Oberfläche. Personen, die sich mit der Analyse von Daten aus sozialen Netzwerken befassen, werden darin viele Annehmlichkeiten finden, die sich positiv auf die Qualität ihrer Arbeit auswirken. Das Ergebnis dieser Arbeit ist auf GitHub [1] zu finden.

[1] https://github.com/texttechnologylab/TwitterPooler

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
NLP	Natural Language Processing
REST	Representational State Transfers

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Motivation

Soziale Medien sind zu einem der wichtigsten Kommunikationskanäle unserer Zeit geworden. Nicht nur einzelne Personen, sondern auch Unternehmen, Organisationen (staatliche und nichtstaatliche), Politiker, Künstler und viele andere nutzen die digitalen Kommunikationsmedien. Die Nutzer tauschen dort ihre Meinungen aus, teilen ihre Erfahrungen und Neuigkeiten aus der Welt.

Die auf diesen Portalen gesammelten Daten sind eine wertvolle Quelle für die wissenschaftliche Forschung und werden sowohl für akademische als auch für industrielle Zwecke verwendet. Von den vielen populären Portalen hat Twitter das größte Interesse bei den Wissenschaftlern geweckt. Dies ist hauptsächlich auf den leichten Zugang zu den Daten und das breite Spektrum der darin enthaltenen Informationen zurückzuführen (vgl. Ahmed u. a. 2017, S. 2–3). Um Zugang zu den Daten des Dienstes zu erhalten, genügt die Registrierung des kostenlosen Entwicklerkontos. Die Kommunikation mit dem Service erfolgt über Representational State Transfers (REST)-Abfragen. Für die Benutzerauthentifizierung wird ein Identifikations-Token verwendet (verfügbar nach der Anmeldung).

Es gibt viele Möglichkeiten und Werkzeuge zur Abfrage und Erfassung von Daten. Eine davon ist z. B. die Verwendung von cURL (Curl URL Request Library). In diesem Fall genügt es, eine zuvor konfigurierte Anfrage (Twitter stellt dafür ein kostenloses Tool zur Verfügung[1]) in das Terminal zu kopieren und das private Token einzugeben. Eine andere Möglichkeit ist die Verwendung der Twitter Application Programming Interface (API) v2 Postman Collection[2]. In beiden Fällen müssen die resultierenden Daten jedoch anschließend manuell formatiert werden. Es gibt auch viele Open-Source-Projekte bzw. Bibliotheken auf dem Markt, die die Nutzung des Dienstes erleichtern. Twitterdev bietet z. B. eine search-tweets-python-Bibliothek[3], die das Erstellen von Abfragen erheblich erleichtert. Der große Nachteil ist jedoch, dass es nur zwei Endpunkte der neu veröffentlichten API unterstützt. Auf der Twitter-Seite[4] befinden sich eine ganze Reihe von Bibliotheken und Tools zur Unterstützung des Dienstes. Alle erfordern jedoch Programmier- und Anpassungsfähigkeiten.

Die Motivation dieser Arbeit ist daher, ein Programm zu erstellen, das nicht nur einen leichten Zugang zu Informationen auf Twitter bietet, sondern auch ermöglicht, diese zu verwalten und zusätzliche Funktionen wie Pooling und die Vorverarbeitung von Natural Language Processing (NLP) für weitere Analysen liefert. Ein interessanter Aspekt ist auch der Einfluss zusätzlicher Funktionalitäten auf die Geschwindigkeit des im Rahmen dieser Arbeit

[1] https://developer.twitter.com/apitools/api

[2] https://www.postman.com/twitter/workspace/twitter-s-public-workspace/collection/9956214-784efcda-ed4c-4491-a4c0-a26470a67400?ctx=documentation

[3] https://developer.twitter.com/en/docs/twitter-api/tools-and-libraries/v2

[4] https://github.com/twitterdev/search-tweets-python/tree/v2

vorgestellten Systems.

1.2. Aufgabenstellung

Ziel dieser Arbeit ist es, ein System zur Erfassung und Verwaltung von Daten aus Twitter zu entwickeln. Zusätzlich soll das System die Möglichkeit des Poolings und der NLP-Vorverarbeitung von heruntergeladenen Inhalten bieten. Alle Funktionen sollen über die Website zugänglich sein, so dass die Nutzung keine fortgeschrittenen Programmierkenntnisse erfordert.

2. Grundlagen

In diesem Kapitel sollen die zentralen Begriffe eingeführt werden, die zum besseren Verständnis der in den folgenden Kapiteln vorgestellten Funktionalitäten und Anwendungsmöglichkeiten der TwitterPooler-Software beitragen. Kapitel 2.1 beschreibt Twitter und seine zwei Komponenten (Tweet, Twitter API v2). In Kapitel 2.2 wird erklärt, was das Pooling bedeutet und ein Beispiel für seine Anwendung gegeben. Am Ende werden TextImager, das Konzept von NLP und das UIMA-Framework dargestellt.

2.1. Twitter

Twitter wurde im Jahr 2006 gegründet und gehört neben Facebook zu den beliebtesten Mikrobloggingdiensten der Welt. Im Gegensatz zu Facebook oder anderen sozialen Netzwerkplattformen steht bei Twitter nicht der soziale Aspekt im Vordergrund, denn die Aufgabe von Twitter ist nicht nur der Meinungsaustausch zwischen den Nutzern, sondern auch die Übermittlung von Informationen. Deshalb suchen die Twitter-Nutzer (im Vergleich zu den Nutzern anderer sozialer Netzwerkplattformen) keine Freunde, sondern konzentrieren sich auf den Austausch von Informationen, was Twitter einzigartig macht. Die Zahl der aktiven Nutzer liegt bei über 330 Millionen pro Monat (vgl. Brett 2017). Zu den Kontoinhabern gehören führende Politiker, berühmte Persönlichkeiten, Unternehmen, Medien, gemeinnützige Organisationen, Behörden und viele weitere. Die Vielfalt der Anwender, die vorherrschende Idee der Informationsvermittlung und die Zugänglichkeit der Daten tragen alle zu einer breit gestreuten Nutzung der Nachrichten des Portals bei. Aus diesem Grund ist Twitter nicht nur ein hervorragendes Marketinginstrument, sondern wird auch von vielen Wissenschaftlern intensiv benutzt. Die Daten des Portals werden unter anderem verwendet, um die Ereignisse in Echtzeit zu erkennen und zu verfolgen, die Entwicklung des Finanzmarktes vorherzusagen oder die Nutzer besser kennenzulernen (vgl. Bruns u. a. 2014, S. 250–251).

2.1.1. Tweet

Auf Twitter erstellte Nachrichten werden als Tweets bezeichnet. Jede Nachricht ist auf 280 Zeichen beschränkt (inklusive Leerzeichen), was den Benutzer zu einer kurzen und prägnanten Formulierung zwingt. Die Tweets bieten jedoch viel mehr als nur das Verfassen und Teilen von Inhalten. Sie wurden mit Tools ausgestattet, die das Interesse an Twitter in der akademischen und geschäftlichen Welt geweckt haben. Die wichtigsten davon sind: Hashtag (#), Mention (@) und Retweet. Jeder Hashtag beginnt mit einem #-Zeichen und ist in den Text integriert. In der Regel werden Hashtags verwendet, um eine Nachricht mit einem Thema oder einem Event zu verknüpfen. In der Praxis werden sie aber auch gerne genutzt, um Sätzen oder Wörtern einen Kontext zu geben (vgl. Dayter 2015, S. 24). Mentions werden

mit dem @-Zeichen versehen und haben die Aufgabe, andere Accounts zu markieren. Die betroffenen Personen werden darüber informiert und zu einem Gespräch eingeladen. Auf diese Weise entstehen Diskussionen, in denen Meinungen und Ansichten zwischen den Nutzern ausgetauscht werden. Retweeten bezeichnet das Weiterleiten, d. h. das erneute Veröffentlichen eines bereits vorhandenen Tweets auf der eigenen Pinnwand. Es ist eine sehr beliebte Aktivität auf dem Portal, die von den Usern auf verschiedene Arten genutzt wird, z. B. um Interesse zu zeigen, zu kommentieren oder die Richtigkeit eines bestimmten Beitrags zu bestätigen. Re-Tweeting ist auch ein geeignetes Instrument, um Unterhaltungen (Threads) zu erzeugen und neue Leser zur Interaktion einzuladen (vgl. Boyd u. a. 2010, S. 6–7). Natürlich gibt es noch weitere Funktionen, wie Anhänge und Links. Sie sind jedoch von untergeordneter Bedeutung, deshalb werden sie hier nicht im Detail beschrieben.

2.1.2. Twitter API v2

Die Twitter-API ist ein Dienst, der den Zugriff auf die auf der Website gesammelten Daten ermöglicht. Derzeit stehen vier Versionen des Produkts zur Verfügung: standardv1.1, premium v1.1, enterprise und Twitter API v2. Im folgenden Abschnitt liegt der Fokus auf der neuesten Version der Twitter API v2 (*Twitter API v2 support* 2022), da TwitterPooler nur die v2-Schnittstelle unterstützt.

Um auf den Dienst zugreifen zu können, wird einen Entwickler-Account benötigt, das auf der Website des Herstellers leicht erstellt werden kann. Derzeit gibt es drei Arten von Accounts:

- Essential: Dieser Account ist sofort nach der Registrierung verfügbar und ermöglicht den Download von 500 000 Tweets pro Monat..
- Elevated: Dieser Account ist nach der Validierung des Anmeldeformulars verfügbar und ermöglicht den Download von 2 000 000 Tweets pro Monat.
- Academic Research: Dieser Account ist ausschließlich für Forscher verfügbar und ermöglicht den Download von 10 000 000 Tweets pro Monat. Inhaber haben außerdem Zugang zu erweiterten Suchoperatoren und die Möglichkeit, die Full-Archive-Endpunkte zu nutzen, die das Herunterladen historischer Tweets ermöglichen.

Um die gewünschten Daten zu erhalten, muss eine GET-Anfrage an einen der vielen verfügbaren REST-Endpunkte gesendet werden. Aufgrund ihrer großen Anzahl werden nur diejenigen vorgestellt, die mit der von mir entwickelten Software, TwitterPooler, kompatibel sind. Dazu gehören:

- Search Tweets: Damit kann nach Tweets gesucht werden, die bestimmte Hashtags, Wörter oder URLs im Text enthalten. Dank integrierter Operatoren und boolescher Logik können komplexe Abfragen erstellt werden, um die Suche zu präzisieren. Search Tweets bietet zwei Endpunkte: Recent und Full-Archive. Recent bietet im Gegensatz zu Full-Archive nur Zugriff auf die Tweets der letzten Woche.

- User Tweet Timeline: Dieser Endpunkt ermöglicht den Zugriff auf die letzten 3200 Tweets, die von einem bestimmten Profil erstellt wurden. Als Eingabe wird eine Benutzer-ID erwartet.
- User Mention Timeline: Dieser Endpunkt ermöglicht den Zugriff auf die letzten 800 Tweets, in denen ein bestimmter Nutzer getaggt wurde. Als Eingabe wird eine Benutzer-ID erwartet.
- Liked Tweets: Dieser Endpunkt gibt alle Tweets zurück, die von einem bestimmten Benutzer geliked wurden. Als Eingabe wird Benutzer-ID erwartet.
- Liking Users: Dieser Endpunkt gibt eine Liste von Benutzern zurück, die einen bestimmten Tweet geliked haben. Als Eingabe wird eine Tweet-ID erwartet.
- Tweet Counts: Er funktioniert ähnlich wie die Endpunkte von Search Tweets, aber statt echter Daten gibt er nur ihre Anzahl zurück.(*Twitter API* 2022).

2.2. Tweet-Pooling

Tweet-Pooling ist ein Verfahren, bei dem Tweets nach bestimmten Kriterien miteinander verknüpft werden, um die Erstellung von Dokumenten mit einer größeren Textmenge zu ermöglichen. Obwohl die über den Dienst heruntergeladenen Daten in gewisser Weise miteinander in Verbindung stehen (z. B. ähnliche Wörter oder Hashtags enthalten), schränken die Kürze und die Form der veröffentlichten Inhalte auf dem Portal die Möglichkeiten der Wissenschaftler erheblich ein und können zu unzuverlässigen Ergebnissen führen (vgl. Hong und Davison 2010, S. 80).

Ein beliebtes Verfahren für das Data Mining auf Twitter ist die Themenanalyse. Dabei werden gemeinsame Themen in Dokumentensammlungen identifiziert und zugeordnet. Diese Methode ist jedoch besonders anfällig für vage, unprofessionell geschriebene und kurze Texte. Dies trägt zu unzutreffenden Ergebnissen bei (vgl. Vayansky und Kumar 2020, S. 1). Forscher, die sich mit diesem Thema befassen und sich bemühen, ein besseres Verständnis der in den Nachrichten enthaltenen Informationen zu erlangen, haben festgestellt, dass die zusätzliche Segregation und Gruppierung von Texten in Makrodokumenten erheblich zur Effizienz der Analyse beitragen, in einigen Fällen sogar ohne Modelloptimierung (vgl. Mehrotra u. a. 2013, S. 4), weshalb das Pooling in letzter Zeit zu einem wichtigen Thema geworden ist.

2.2.1. Tweet-Pooling-Verfahren

Tweets lassen sich mit einer Vielzahl von Verfahren sortieren und filtern. In diesem Kapitel werden jedoch nur die vorgestellt, die von TwitterPooler unterstützt werden, nämlich:

- Author-wise-Pooling: Mittels dieses Verfahrens werden die Nachrichten nach ihren Autoren sortiert. Somit wird für jeden Autor ein eigenes Dokument mit allen von ihm verfassten Tweets erstellt.

- Hashtag-based-Pooling: Im Gegensatz zur obigen Methode werden die Tweets nach Hashtags sortiert.
- Temporal-Pooling: Dieses Verfahren basiert auf der Trennung nach Zeiteinheiten, zum Beispiel nach Tagen oder nach Stunden..
- Burst-Score-wise-Pooling: Dieses Verfahren basiert auf der Trennung nach Schlüsselwörtern, deren Popularität in einem bestimmten Zeitraum einen vorher festgelegten Wert übersteigt (vgl. Mehrotra u. a. 2013, S. 2).

2.2.2. Praktische Anwendung

Wie in Abschnitt 2.2 erwähnt, dient das Pooling grundsätzlich der Verbesserung der Themenanalyse. Ein Beispiel für den Einsatz von Pooling-Verfahren ist die Studie von Weng u.a. zur Identifizierung von Influencern auf Twitter. Zu diesem Zweck wurden die Konten anhand der Ähnlichkeit der Themen und der Struktur ihrer Follower (Links) miteinander verglichen. Zur Ermittlung von Themen, die sich auf bestimmte Konten beziehen, wurde die Themenanalyse verwendet. Um die gewünschten Ergebnisse zu erhalten, wurden (für jeden Autor separat) Dokumente erstellt, die alle Tweets des Autors enthielten (vgl. Weng u. a. 2010, S. 261–264), d.h. es wurde ein Author-wise-Poooling durchgeführt. Khan u.a. (Khan u. a. 2019) verwendeten in ihrer Arbeit zur Extraktion wichtiger Informationen aus Twitter (populäre Themen, Burst News) das Hashtag-Pooling. AlAgha hingegen untersuchte die Nutzermeinungen zu COVID-19 mit Hilfe von Autoren-Pooling, was die Zuverlässigkeit der Ergebnisse erheblich beeinflusste (AlAgha 2021).

2.3. TextImager

TextImager ist eine von Text Technology Lab an der Goethe-Universität Frankfurt entwickelte Software, die auf dem UIMA-Framework basiert und eine breite Palette von NLP-Werkzeugen für die Textanalyse in verschiedenen Sprachen bietet. Es ist außerdem mit einer Webschnittstelle und mit Visualisierungstools ausgestattet (vgl. Hemati u. a. 2016, S. 59). Die TextImagerClient-API bietet auch die Möglichkeit, TextImager-Ressourcen extern zu nutzen. Dieser Dienst wird im TwitterPooler für die NLP-Vorverarbeitung von Twitter-Daten verwendet.

2.3.1. Natural Language Processing

NLP gehört zu den Fachgebieten der Informatik (künstliche Intelligenz) und der Linguistik. Dank der NLP-Tools und der dort implementierten Algorithmen sind die Computer in der Lage, Texte zu analysieren und zu verstehen (vgl. Chopra u. a. 2020, Kap. 1). Dieser Bereich befindet sich derzeit in einer umfassenden Entwicklung und bietet eine breite Palette von Möglichkeiten, die von der einfachsten Tokenisierung, d. h. der Aufteilung eines Satzes auf die einzelnen Tokens bzw. seine Bestandteile, bis zu komplexen Algorithmen für die Bewertung von Emotionen oder die Ähnlichkeit von Dokumenten reichen. Die beliebtesten Bibliotheken, die NLP-Tools anbieten, sind NLTK, Spacy und CoreNLP. Es gibt daneben jedoch

noch viele weniger bekannte Bibliotheken auf dem Markt. TextImager arbeitet mit vielen von ihnen zusammen, die über die TextImagerClient-API auch für TwitterPooler bereit zur Verfügung stehen.

2.3.2. UIMA-Framework

Apache UIMA ist ein Open-Source-Framework, welches hilfreiche Werkzeuge für die Verwaltung von Objekten, beispielsweise von NLP-Dokumenten in Pipelines bietet. Daher ist es in der Branche ein sehr beliebtes Framework. Jedes Dokument (d.h. jeder Text) wird durch ein CAS-Objekt[1] dargestellt, in dem sich alle Informationen (Metadaten, Annotationen) befinden. Die Analyse des Dokuments (bzw. des CAS-Objektes) erfolgt über die Interface Analysis Engine, die auch für die Aktualisierung des CAS zuständig ist (vgl. Verspoor und Baumgartner 2013, S. 2320–2322).

[1] `https://uima.apache.org/d/uimaj-current/apidocs/org/apache/uima/cas/CAS.html`

3. Konzeptionierung

In diesem Kapitel werden die Idee und der Plan für die Anwendung vorgestellt. Zunächst werden die Anforderungen, die die Anwendung erfüllen sollte, geschildert. Im Anschluss folgt eine Beschreibung von Anwendungsfällen und Beispielszenarien. Zuletzt wird auf bisherige verwandte Arbeiten eingegangen.

3.1. Anforderungen

Wie im ersten Kapitel erwähnt, besteht der Zweck des TwitterPooler-Systems darin, einen einfachen Zugang zu Twitter-Daten zu ermöglichen. Darüber hinaus sollte das Programm mit Funktionen zur komfortablen Verwaltung der erstellten Abfragen, des Poolings und der NLP-Vorverarbeitung für die gesammelten Tweets ausgestattet sein. Im Folgenden werden diese Anforderungen ausführlicher thematisiert und erläutert. Um sie besser zu veranschaulichen, werden vier Hauptkomponenten unterschieden:

- Allgemein:
  - Das Programm soll mit der neuesten Version der Twitter-API v2 kompatible sein.
  - Die gewonnenen Daten sollen in der NOSQL-Datenbank gespeichert werden.
  - Die Software soll die gleichzeitige Ausführung mehrerer Prozesse ermöglichen (Multithreading). Dazu gehören das Herunterladen von Tweets, das Pooling, die NLP-Vorverarbeitung und das Erstellen von XMI-Dateien.
- Abfragen:
  - Das Programm soll in der Lage sein, GET-Anfragen zu erstellen und an bestimmte Endpunkte zu senden.
  - Die empfangenen Tweets sollen alle Zusatzinformationen enthalten, ordnungsgemäß formatiert sein und in der Datenbank gespeichert werden. Zu den zusätzlichen Informationen gehören unter anderem Autoreninformationen, Anhänge oder verbundene Beiträge.
  - Mit dem Programm sollen die Prozesse überwacht werden können (Informationen über die Anzahl der heruntergeladenen Tweets und den Status).
  - Die Anwendung soll die Möglichkeit der gleichzeitigen Nutzung mehrerer Entwicklerkonten bieten. Das bedeutet, dass der Nutzer das Konto wählen kann, über das die Anfrage gesendet werden soll.
  - Der Kontostand soll angezeigt und aktualisiert werden (Anzahl der heruntergeladenen Tweets pro Monat, automatische Rücksetzung des Limits).
  - Das Programm soll angehaltene Prozesse wieder aufnehmen können.

  - Der TwitterPooler soll die NLP-Vorverarbeitung von Twitter-Daten ermöglichen. Zu diesem Zweck sollen die von TextImager bereitgestellten Tools verwendet werden.
  - Das Programm soll über eine Funktionalität verfügen, die das erneute Versenden von Anfragen, die in der Vergangenheit gesendet wurden, verhindert.
  - Es soll möglich sein, die heruntergeladenen Inhalte zu löschen.

- Pooling:
  - Die Anwendung soll vier Pooling-Verfahren anbieten (Author-wise-Pooling, Hashtag-based-Pooling, Temporal-Pooling, Burst-Score-wise-Pooling).
  - Es soll ein zusätzlicher Limit-Parameter eingeführt werden, um eine Mindestanzahl von Tweets pro Dokument festzulegen. Dank des oben genannten Wertes ist es möglich, erstellte Datensammlungen mit einer geringen Anzahl von Nachrichten zu ignorieren.
  - Es soll ein zusätzlicher Size-Parameter eingeführt werden, um die Zeitspanne der Dokumente zu spezifizieren (nur für Temporal und Burst-Score-Pooling verfügbar). Die Zeitspanne bezeichnet die Zeiten, zwischen denen Tweets freigegeben wurden.
  - Das Programm soll über eine Funktionalität verfügen, die prüft, ob der ausgewählte Vorgang in der Vergangenheit durchgeführt wurde.
  - Es soll möglich sein, erstellte Dokumente zu löschen.

- Erstellung von XMI-Dateien:
  - Das Programm soll in der Lage sein, XMI-Dateien zu erstellen und auf der Festplatte zu speichern. XMI-Dateien sollen aus einem CAS-Objekt erzeugt werden.

- Benutzeroberfläche:
  - Die Benutzeroberfläche soll aus fünf Seiten bestehen (Home-Seite, Search-Seite, History-Seite, Resume-Seite, Pooling-Seite).

3.2. Use-Cases

TwitterPooler wurde hauptsächlich für die Wissenschaftler entwickelt. Die einfache Bedienung und die Möglichkeit, frei verfügbare Endpunkte zu nutzen (siehe 2.1.2), machen dieses Programm auch für ein breiteres Publikum attraktiv. Im folgenden Abschnitt wird ein Beispielszenario für die Anwendung entworfen. Zuvor wird ein Diagramm vorgestellt, in dem alle Anwendungsmöglichkeiten beschrieben sind (siehe Abbildung 3.1). Wie im untenstehenden Diagramm zu sehen ist, besteht die größte und umfangreichste Nutzungsmöglichkeit in der Suche nach Tweets und deren Speicherung in der Datenbank. Der zweite und ebenso leistungsstarke Bereich ist das Pooling. Ein wichtiger Bestandteil ist auch die Funktion zum Herunterladen von Tweets in Form von XMI-Dateien. Abbildung 3.2 zeigt ein Beispielszenario für die Nutzung des Systems. Das abgebildete Schema veranschaulicht die Suche und das

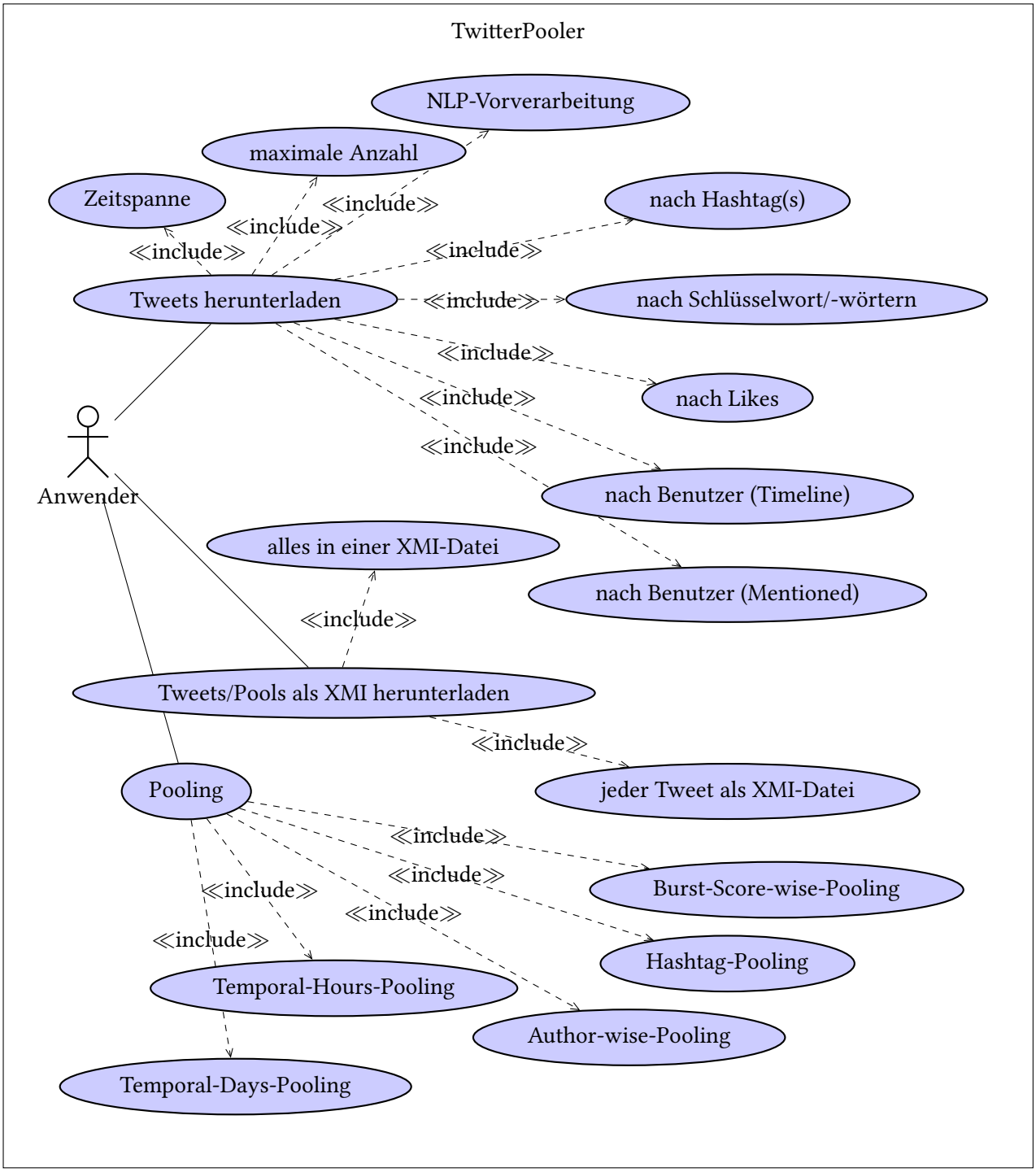

Abbildung 3.1.: Anwendungsfalldiagramm

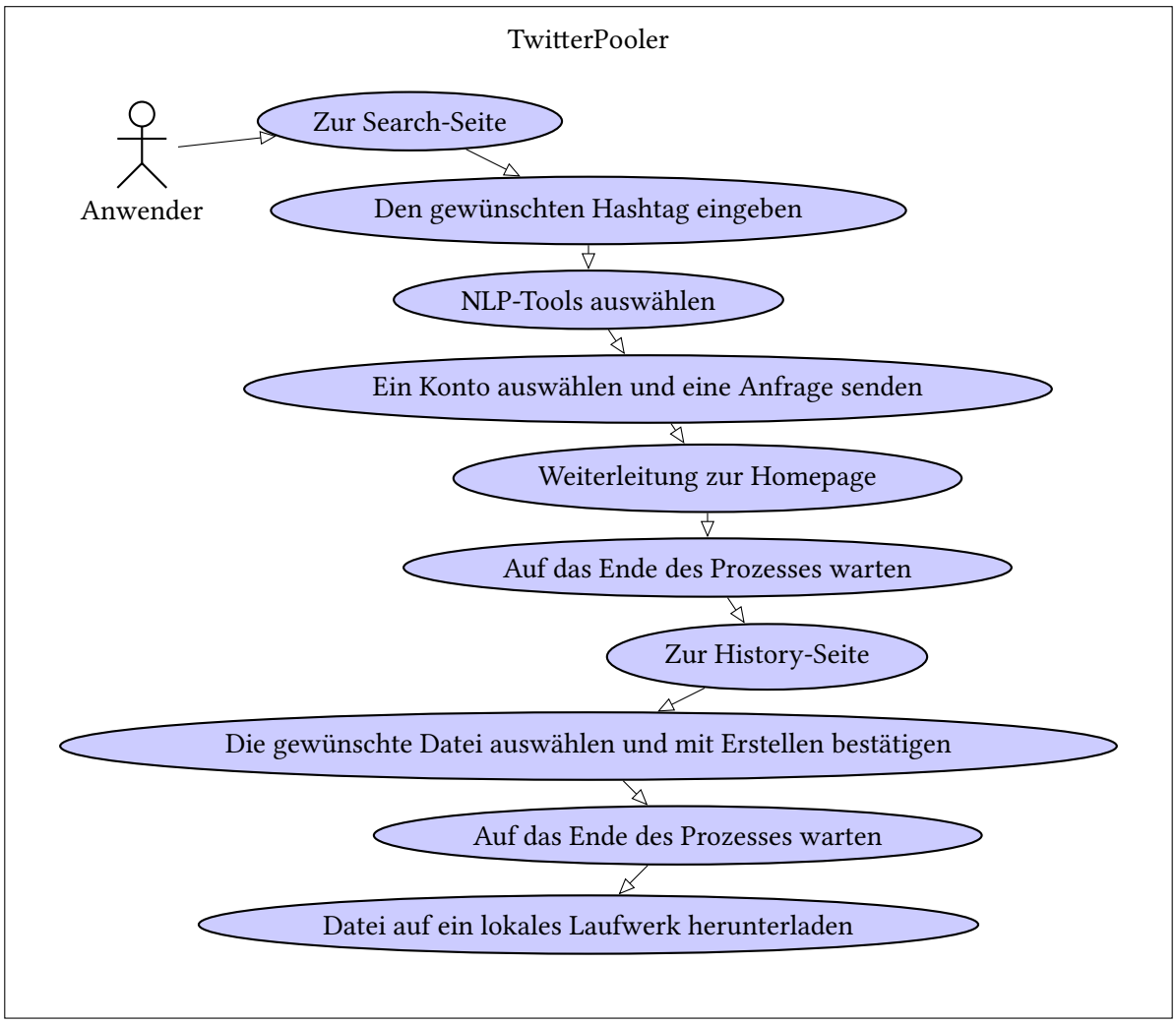

Abbildung 3.2.: Beispielszenario

Herunterladen der angegebenen Tweets auf die Festplatte als XMI-Datei. In der XMI-Datei gibt es Tweets in Form eines Strings, der ein CAS-Objekt darstellt. Da in diesem Beispiel auch NLP-Tools ausgewählt wurden, enthält jeder String (CAS-Objekt) auch vorverarbeitete Daten (z.B. Tokens, Lemmas usw.). Auf diese Weise ist der Nutzer sofort für weitere Recherchen gerüstet. Um die im Objekt enthaltenen Informationen lesen zu können, müssen die Strings in CAS (jCAS in Java) umgewandelt werden. Text Technology Lab entwickelte ein Interface, um die Serialisierung und Deserialisierung der oben genannten Objekte zu ermöglichen[1].

3.3. Arbeiten mit verwandten Themen (Related Work)

Wie bereits in der Einleitung erwähnt, gibt es auf dem Markt viele Tools und Bibliotheken, die die Kommunikation mit Twitter unterstützen. In diesem Unterkapitel wird eine Über-

[1] `https://github.com/texttechnologylab/UIMADatabaseInterface`

sicht zu den derzeit populärsten Programmen gegeben, diese werden anschließend mit der im Rahmen der vorliegenden Arbeit erstellten Anwendung verglichen. Die folgenden Tools wurden auf der offiziellen Twitter-Website[2] und durch weiterführende Recherche gefunden.

Eine der beliebtesten Bibliotheken ist Tweepy (sie hat 204 Mitwirkende auf GitHub und wird von 30,8k Nutzern verwendet[3]). Das gesamte Projekt wurde in Python geschrieben, einer der derzeit am weitesten verbreiteten Sprachen in der Data-Science-Branche. Die umfangreiche Dokumentation[4] ist ebenfalls ein großer Vorteil. Tweepy ist kompatibel mit der älteren V1.1 API und der neuesten V2. Sie bietet außerdem Zugriff auf eine große Anzahl von Endpunkten wie Spaces, Kompatibilität, Tweets, Retweets oder Benutzer und unterstützt die vollständige Full-Archive. Um eine GET-Anfrage zu senden, sind praktisch nur zwei Schritte nötig:

- tweppy.client("token") ist zu initialisieren. Dabei ist anstelle von token ein persönlicher Bearer-Token einzufügen.
- Danach müssen der Endpunkt und die gewünschten Parameter angegeben werden, z.B. answer = client.search_search_recent_tweets("#nba", max_results=100).
- Die heruntergeladene Informationen sind in answer.data, answer.includes und answer.meta verfügbar.

Es wird allerdings nur eine Anfrage geschickt, sodass maximal 100 Tweets empfangen werden können. Tweepy verfügt jedoch über einen Paginierungsmechanismus, der sicherstellt, dass nachfolgende Requests automatisch gesendet werden. Hierfür muss die Funktion tweepy .Paginator(client.search_search_recent_tweets("#nba",max_results=500)) verwendet werden. Dabei werden die 500 neuesten Tweets downgeloadet. Ohne die Angabe des Parameters max_result werden alle verfügbaren Daten heruntergeladen. Wie bereits erwähnt, werden die Informationen in den drei Objekten Data, Includes und Meta bereitgestellt. Das Objekt Data enthält die Tweets, Includes liefert zusätzliche Informationen wie Autoren, Anhänge und Posts, die mit den ursprünglichen Nachrichten in Verbindung stehen. Dies kann ein Retweet oder eine Antwort sein. Tweepy bietet jedoch keinen Mechanismus, um Includes automatisch mit den Tweets zu mappen. Dazu müssen eigene Funktionen implementiert werden, was bei großen Datenmengen zeitaufwändig und kompliziert sein kann. Eine weitere fehlende Funktion ist die Möglichkeit, den Status des Twitter-Overlays zu überprüfen. Wird das Limit überschritten, wird der Nutzer erst beim Absenden der Anfrage informiert. Tweepy hat keine eingebauten Optionen für die Speicherung von Tweets in einer Datenbank. Trotz ihrer Beliebtheit in der Forschungsgemeinschaft ist Tweepy nicht mit Methoden zur Unterstützung von NLP ausgestattet (*Tweepy Documentation* 2022). Ein großes Plus ist die Sprache, in der das Projekt geschrieben ist, denn Python hat viele nützliche Module, wie z. B. Panda[5], Numpy[6] oder PyTorch[7]. Das ist wahrscheinlich der Grund, warum Tweepy so beliebt ist.

[2] https://developer.twitter.com/en/docs/twitter-api/tools-and-libraries/v2

[3] https://github.com/tweepy/tweepy

[4] https://docs.tweepy.org/en/latest/index.html

[5] https://pandas.pydata.org/

[6] https://numpy.org/

[7] https://pytorch.org/

Die nächste Bibliothek, die hier vorgestellt wird, ist twittered. Ein Blick auf die Statistiken auf GitHub (15 Mitwirkende und 35 Nutzer[8]) zeigt, dass Twittered im Vergleich zu Tweepy nicht einmal halb so bekannt ist. Die Wahl fiel jedoch auf diese Bibliothek, weil es in der gleichen Sprache wie TwitterPooler geschrieben ist, nämlich in Java. Twittered arbeitet, ähnlich wie Tweepy, mit älteren und neueren Versionen der Twitter-API. Unterstützt ebenso eine sehr große Anzahl von Endpunkten, sowohl Recent als auch Full-Archive. Um ein Request zu senden, muss das TwitterClient-Objekt initialisiert werden. Der TwitterClient enthält viele Funktionen, die für das Senden von Anfragen an eine bestimmte Adresse zuständig sind. In diesem Fall werden jedoch keine zusätzlichen Formen (z. B. den Paginator) benötigt. Es genügt, die maximale Anzahl in den Parametern anzugeben und das Programm erneuert die Anfragen automatisch. Bei der Recherche im Projekt wurde keine Implementierung für Multithreading gefunden, daher wird dies vermutlich nicht unterstützt. Twittered (wie auch Tweepy) ordnet Tweets nicht automatisch den in Includes enthaltenen Informationen zu. Es ist jedoch mit einer Reihe von Funktionen ausgerüstet, die das Auffinden der Dokumente erleichtern. Zusätzliche Werkzeugen wie Pooling und NLP-Vorverarbeitung gehören allerdings nicht zur Ausstattung.

Im Allgemeinen gibt es viele solcher Projekte auf dem Markt. In den meisten gängigen Programmiersprachen existieren Bibliotheken von unterschiedlicher Leistungsfähigkeit, die die Kommunikation mit der Twitter-API erleichtern. Es gibt auch Programme, die die Übermittlung über ein Terminal ermöglichen, wie z.B. BluebirdPS[9] oder twarc[10].

Ein bemerkenswertes Konzept ist der Twitter-Explorer. Dieses Projekt bietet nicht nur die Möglichkeit, Informationen von Twitter herunterzuladen, sondern verfügt auch über integrierte Tools zur Analyse und Visualisierung. Außerdem ist es mit einer grafischen Oberfläche ausgestattet. Das Programm besteht aus drei Modulen: Collector, Visualizer und Explorer. Das Collector-Modul dient zum Erstellen von Abfragen und zur Sammlung von Informationen aus Twitter. Der Visualizer und der Explorer bieten die Möglichkeit, die gesammelten Daten zu analysieren und zu überprüfen. Der Twitter-Explorer ist jedoch nicht mit der neuesten API v2 kompatibel (vgl. Pournaki u. a. 2020, S. 108).

Ein weiteres erwähnenswertes Werkzeug ist DMI-TCAT[11]. Das 2013 an der Universität Amsterdam entwickelte Programm bietet (wie der Twitter-Explorer) sowohl Möglichkeiten zur Erhebung als auch zur Analyse von Tweets. Außerdem verfügt es über eine Weboberfläche und integrierte Kompatibilität mit der MYSQL-Datenbank. Besonders nützlich sind die Funktionen Sub-Sampling und Tweet-Export. Das Sub-Sampling filtert die gesammelten Informationen und ist sehr umfangreich. Mit dem Tweet-Export hingegen können Daten nach einem ausgewählten Parameter exportiert werden (z. B. Tweet-Statistiken, Nutzer-Statistiken, Hashtag-Häufigkeit usw.). Darüber hinaus ist es auch möglich, Netzwerke zu erstellen (z.B. nach Beziehungen zwischen Nutzern oder Hashtags) und sie in einem Format zu konvertieren, das mit dem kostenlosen Visualisierungsprogramm Gephi[12] kompatibel ist (vgl. Borra und Rieder 2014, S. 262, 266–273). DMI-TCAT ist somit insgesamt ein sehr leis-

[8] https://github.com/redouane59/twittered

[9] https://github.com/thedavecarroll/BluebirdPS

[10] https://twarc-project.readthedocs.io/en/latest/twitter-developer-access/

[11] https://github.com/digitalmethodsinitiative/dmi-tcat

[12] https://gephi.org/

tungsfähiges Programm mit vielfältigen Einsatzmöglichkeiten.

Derzeit scheint es noch keine kostenlose Anwendung zum Herunterladen von Twitter-Daten zu geben, die Lösungen wie Pooling oder NLP-Vorverarbeitung bietet. TwitterPooler ist daher in diesem Zusammenhang ein innovatives Projekt mit neuen Funktionen.

4. Implementierung

In diesem Kapitel wird die Implementierung des Systems beschrieben. Zu Beginn werden die verwendeten Technologien vorgestellt, dann werden der Aufbau und die wichtigsten Funktionen der Anwendung präsentiert. Der letzte Teil ist der Erstellung einer grafischen Schnittstelle gewidmet.

4.1. Verwendete Technologien

TwitterPooler wurde in Java geschrieben. Sie gehört zu den populärsten Programmiersprachen, da sie auf vielen Plattformen eingesetzt werden kann, eine breite Palette kostenloser Bibliotheken bietet und die objektorientierte Programmierung unterstützt. Java wurde seit ihrer Entstehung zu Beginn der 90er Jahre (Sun Microsystems, Inc.) ständig weiterentwickelt und hat inzwischen 17 Versionen erreicht. TwitterPooler wurde jedoch in Version 11 geschrieben. Einer der Gründe für die Wahl dieser Programmiersprache liegt darin, dass TextImager und sein gesamtes Ökosystem in Java geschrieben sind und TwitterPooler in einer seiner Funktionen dessen Ressourcen nutzt. Weiterer wichtige Indikatoren für die Wahl waren die hohe Leistung, die Unterstützung für Multithreading und die kostenfreie Nutzbarkeit (vgl. Schildt 2021, Kap. 1).

Um externe Bibliotheken zu verwalten und das Projekt zu strukturieren, wurde Apache Maven[1] genutzt. Dabei handelt es sich um ein Open-Source-Tool zur Erleichterung des Projektmanagements. Der zentrale Ort der Administration ist die Datei pom.xml. Hier kann der Entwickler angeben, welche Komponenten in seinem Projekt verwendet werden sollen. Maven wird nach der Eingabe automatisch (während des Bauprozesses) die entsprechenden Pakete finden und herunterladen. Ein großer Vorteil dieses Werkzeugs besteht in der Möglichkeit, das Programm mit anderen Entwicklern zu teilen. Maven bietet eine Deploy-Funktion, mit der das Projekt in das zentrale Maven-Repository gestellt werden kann (vgl. Varanasi 2019, Kap. 1, 3).

Das TwitterPooler-Projekt wurde mit dem Quarkus-Framework[2] erstellt, einem der jüngsten Open-Source-Java-Frameworks auf dem Markt. Quarkus ist in erster Linie für die Entwicklung von Cloud-Anwendungen konzipiert. Es bietet eine Kubernetes-Integration und ist speziell für die Ablage von Anwendungen in einem Container optimiert. Das erstellte Projekt wird automatisch mit einem Dockerfile versehen, das die Erstellung von Docker-Containern erleichtert. Außerdem verfügt es über einen reduzierten RSS-Speicher, was seine Attraktivität deutlich erhöht. Dies war auch einer der wichtigsten Faktoren bei der Auswahl, weil TwitterPooler Multithreading erlaubt und auf langlaufenden Prozessen basiert. Sehr nützlich ist auch die Möglichkeit des automatischen Ladens im Entwicklungsmodus, was die Ent-

[1] https://maven.apache.org/

[2] https://quarkus.io/

wicklung erheblich vereinfacht und beschleunigt. Quarkus bietet viele Standardfunktionen zur Unterstützung der Erstellung von RESTfull-Services. Die Annotation @Path definiert die URI-Adresse, und mit @Get @POST @PUT @DELETE wird die Endpunktmethode spezifiziert. Quarkus ist somit dem beliebtesten Framework dieses Typs, Spring[3], sehr ähnlich. Mit Quarkus lässt sich auch der Datentyp für die Ein- und Ausgabe jedes Endpunkts festlegen. Diese Funktion ist durch die Verwendung der @Consume/@Produces-Anmerkung möglich. Das Framework stellt den Entwicklern auch Werkzeuge für die bequeme Kommunikation zwischen internen Adressen zur Verfügung, die ein einfaches Auslesen der übertragenen Parameter ermöglichen. Dank der @Context Urinfo-Annotation werden die Werte auf die angegebene Variable übertragen (vgl. Bueno und Porter 2020, Kap. 1, 3, Abs. 3.1, 3.2).

Um Webseiten zu erstellen, wurde Freemarker[4] verwendet. Das Tool ist kostenlos und dient der Erstellung von HTML-Seiten. Die Idee des Produkts besteht in der Trennung des Designs vom Backend. Die generierte Datei wird aus zwei Quellen erstellt, einem Template und Java-Objekten. Die Vorlage ist als .ftl-Format angelegt, die der HTML-Datei sehr ähnlich ist. Allerdings verfügt Freemarker über zusätzliche Funktionen, die ein direktes Mapping von Daten aus dem Programm ermöglichen (*Freemarker* 2022). Für das Layout wird das Bootstrap-Framework verwendet, welches einfach in die .ftl-Datei geladen werden kann.

Das Projekt verwendet die Programmiersprache JavaScript, eine Skriptsprache zur Erstellung dynamischer Webseiten. Die Hauptaufgabe von JavaScript ist die Behandlung von Ereignissen, die im Webbrowser auftreten (vgl. Wilton 2003, S. 1–2). TwitterPooler verwendet JavaScript hauptsächlich zur Vorbereitung der Parameter und zur Vermeidung von Fehlern auf der Client-Seite.

Für die Datenerfassung wurde die MongoDB[5] verwendet, die zu den NOSQL-Datenbanken gehört. Einer der größten Vorteile der oben genannten Technologie besteht in ihrer Leistungsfähigkeit und der Möglichkeit, verschiedene Arten von Daten auf unterschiedliche Weise zu speichern. MongoDB gehört zu den Datenbanksystemen, die Informationen als Dokumente ablegen. Dieser Weg ist besonders für API-Anwendungen zu empfehlen, da die Objekte in einem JSON-ähnlichen Format gespeichert sind, was die Kommunikation zwischen Anwendungen erheblich erleichtert (vgl. Trelle 2014, S. 2–4).

Der Quellcode von TwitterPooler befindet sich auf dem GitLab des Text Technology Lab[6]. GitLab unterstützt die Git-Technologie und ist damit ein hervorragendes Werkzeug für die Versionskontrolle und die Verwaltung des Projektverlaufs.

4.2. Datenbank

Datenbanken spielen eine wichtige Rolle und sind ein unverzichtbarer Bestandteil jeder Anwendung. Sie dienen der Speicherung und Verwaltung von Daten, die für das optimale Funktionieren der Systeme erforderlich sind. Wie bereits in Kapitel 4.1 erwähnt, verwendet TwitterPooler das zur NOSQL-Familie gehörende MongoDB.

[3] https://spring.io/

[4] https://github.com/quarkiverse/quarkus-freemarker

[5] https://www.mongodb.com/

[6] https://gitlab.texttechnologylab.org/users/sign_in

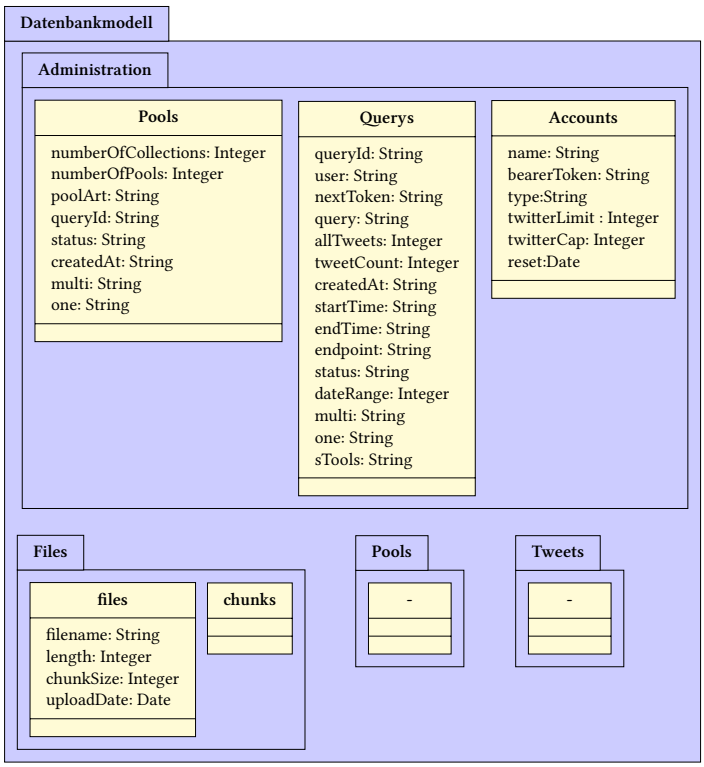

Abbildung 4.1.: Datenbankmodell

Um die Ansprüche des Programms besser realisieren zu können und die Struktur nicht zu unübersichtlich zu gestalten, wurden vier separate Datenbanken eingerichtet: Administration, Tweets, Pools und Files. Diese sind jeweils für die Speicherung einer anderen Sorte von Informationen zuständig. Das Diagramm 4.1 zeigt das entwickelte Modell. Wie zu sehen ist, weisen die Namen bereits auf den Zuständigkeitsbereich der einzelnen Komponenten hin. Aufgrund ihrer Länge und der großen Anzahl von Attributen werden jedoch alle Elemente einzeln vorgestellt und ihre Bedeutung im Detail beschrieben.

In der Administration befinden sich die erforderlichen Verwaltungsdaten für das Programm. Es besteht aus drei Sammlungen: Accounts, Querys und Pools. Die erste enthält Dokumente mit den wichtigsten Informationen über das auf Twitter eingerichtete Entwicklerkonto (siehe 2.1.2). Tabelle 4.1 zeigt eine detaillierte Beschreibung der Parameter eines einzelnen Datensatzes. Eine weitere wichtige Kollektion ist Querys. Die hier gespeicherten Einträge enthalten Informationen, die für die Verwaltung der Tweet-Abrufprozesse notwendig sind. Für jede gesendete GET-Anfrage wird ein eigenes Dokument erstellt. Tabelle 4.2 enthält eine detaillierte Beschreibung der darin vorhandenen Elemente. Die Komplexität und der Umfang der in der Querys-Kollektion gesammelten Dokumente wird durch die hohen Anforderungen von TwitterPooler bestimmt. Die große Anzahl von Parametern ermöglicht viele zusätzliche Funktionalitäten. Ein Beispiel ist DateRange, das zur Verhinderung des erneuten Herunterladens bestehender Tweets verwendet wird. TweetCount hingegen kontrolliert die Anzahl der heruntergeladenen Dateien, so dass der Prozess angehalten werden kann und nur eine bestimmte Zahl von Tweets heruntergeladen wird. Die dritte Sammlung in der

Administrations-Datenbank ist Pools, die für das Management von Pooling-Prozessen verwendet wird. Tabelle 4.3 enthält eine detaillierte Beschreibung.

Tabelle 4.1.: Accounts

Parameter	Beschreibung
name	Name des Kontoinhabers.
bearerToken	Identifikationsschlüssel für den Zugriff auf Twitter API v2 [7].
type	Kontotyp (siehe Abschnitt 2.1.2).
twitterLimit	Verfügbare Anzahl von Tweets pro Monat (siehe Abschnitt 2.1.2).
twitterCap	Aktuelle Anzahl der heruntergeladenen Tweets.
reset	Datum der Rücksetzung des Kontostands (twitterCap).

Tabelle 4.2.: Querys

Parameter	Beschreibung
queryId	Eine eindeutige Identifikationsnummer. Dient zur Identifizierung von Anfragen und gespeicherten Tweets.
user	Name des Erstellers der Abfrage, zugewiesen von Accounts.name (siehe Tabelle 4.1)
nextToken	Paginierungs-Token (siehe Abschnitt 3.3).
query	Inhalt der Anfrage (z.b. Hashtag, Schlüsselwörter, Benutzer-ID)
allTweets	Ergebnis des Endpunkts Tweet Counts, d. h. die Anzahl der verfügbaren Tweets, die heruntergeladen werden können (siehe Abschnitt 2.1.2).
tweetCount	Anzahl der Tweets, die derzeit heruntergeladen wurden.
createdAt	Das Erstellungsdatum der Abfrage.
startTime	Das Startdatum, ab dem die Tweets geliefert werden sollen.
endTime	Das Enddatum, zu dem die Tweets geliefert werden sollen.
endpoint	Der Name des Endpunktes.
status	Aktueller Stand des Prozesses (Running, Stopped und Complete).
dateRange	Stellt den Zeitraum der Tweets dar (Bsp. 2022-01-24/2022-01-30)
multi	Informationen über verfügbare XMI-Dateien.
one	Informationen über die verfügbare XMI-Datei.
sTools	Die Namen der NLP-Tools, die für das Pre-Processing verwendet wurden.

Tabelle 4.3.: Pools

Parameter	Beschreibung
queryId	Identifikationsnummer der Kollektion von Tweets.
poolArt	Art der erzeugten Pools
numberOfCollections	Anzahl der in allen Polls enthaltenen Tweets, die zu dem in poolArt angegebenen Typ gehören
numberOfPools	Anzahl der in der Pools-Datenbank erstellten Dokumente.
status	Aktueller Stand des Prozesses (Running und Complete)
createdAt	Das Erstellungsdatum der Abfrage.
multi	Informationen über verfügbare XMI-Dateien.
one	Informationen über die verfügbare XMI-Datei.

Neben der Administration gibt es drei weitere Datenbanken, die die Zieldaten (Tweets) enthalten. Zwar speichern sie alle Twitter-Nachrichten, allerdings unterscheiden sie sich in ihrer Einteilung oder in ihrem Format. Daher wurden drei separate Datensammlungen angelegt. In Tweets sind alle von Twitter geladenen Nachrichten gespeichert. Für jede Abfrage wird eine eigene Kollektion mit einem Namen erstellt, der der queryId aus dem entsprechenden Administrartion.Querys-Dokument entspricht. Jeder Post wird als separates Dokument im JSON-Format gespeichert. Abbildung 4.2 stellt die Strukturelemente dar:

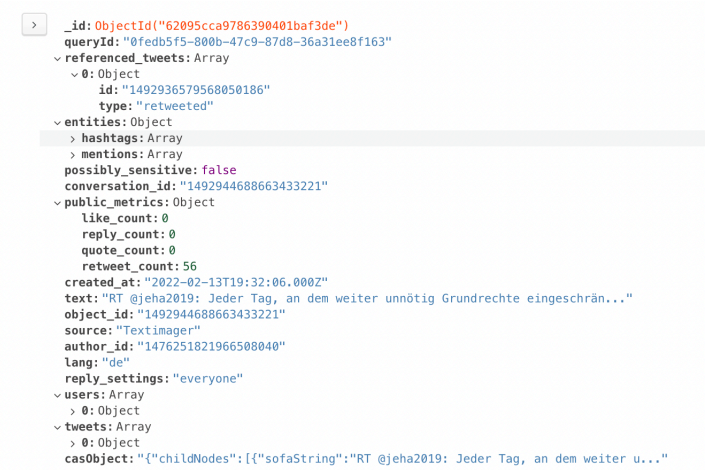

Abbildung 4.2.: Tweet

Die Pools-Datenbank besteht aus Kollektionen, die jedem erstellten Pool entsprechen, und speichert Tweets in genau der gleichen Form wie in Abbildung 4.2. Um die Pools-Kollektionen auf einfache Weise zu identifizieren, wurde ein spezielles Benennungssystem eingeführt. Der Name eines jeden Pools besteht aus drei Teilen:

- Prefix: Der Teil beschreibt die Art des Poolings und seine Parameter (siehe Tabelle 4.5).

- Kennzeichen: Der Teil beschreibt ein gemeinsames Merkmal der enthaltenen Dokumente (z.b. Hashtag oder UserId).
- QueryId: Der Teil bezeichnet die Identifikationsnummer der Quelldaten.

HL20-lufthansa-51794669-6f7d-4343-9d2f-5e3ed308dc44 bedeutet z. B., dass die Kollektion Tweets beinhaltet, die durch das Hashtag-Pooling-Verfahren mit den Parametern L20 (siehe Abschnitt 4.3.1 für eine detaillierte Beschreibung) erzeugt wurden und den Hashtag lufthansa enthalten, der aus den in der Tweet-Datenbank gesammelten Daten unter dem Namen 51794669-6f7d-4343-9d2f-5e3ed308dc44 abgeleitet wurde.

Der letzte Bestandteil des Modells ist die Datenbank Files (siehe Abbildung 4.1), in der die XMI-Dateien gespeichert sind. Um die Speicherung großer Dateien zu ermöglichen, wird das GridFS[8] benutzt, das speziell für diese Zwecke entwickelt wurde. Files bestehen aus zwei Sammlungen: fs.chunks und fs.files. Die Chunks enthalten die XMI-Dateien (im Binärformat). In Files gibt es Informationen über gespeicherte Datei, wie z. B. den Dateinamen.

4.3. Programmstruktur

Die Architektur des TwitterPooler-Systems basiert auf dem MVC-Modell (Model, View, Controller). Es ist eines der beliebtesten Entwurfsmuster für Developer. Das Konzept besteht darin, die Anwendung in drei Hauptkomponenten zu unterteilen:

- Model: Die in diesem Teil implementierten Klassen sind für die Programmlogik, die Datenverwaltung und die Verarbeitung zuständig. Diese Komponente lässt sich daher als Herzstück des Systems ansehen.
- View: Diese Komponente umfasst die HTML-Vorlagen, die Daten präsentieren.
- Controller: Diese Komponente ist für den Betrieb des Programms verantwortlich. Hauptsächlich übermittelt sie die Daten von der Ansicht zum Modell und umgekehrt (vgl. Späth 2021, Kap.1).

TwitterPooler wurde in sechs separate Pakete aufgeteilt. Jeder Inhalt dient einem anderen Zweck. Die Motivation für die Erstellung einer so großen Anzahl von Modulen war nicht nur dem MVC-Muster zu folgen, sondern auch die darin enthaltenen Klassen kontextuell zu trennen. Daher sind bis zu vier Pakete für die Hauptlogik der Anwendung zuständig, eine für die Ansicht und eine für die Verarbeitung und Übertragung der vom Benutzer eingegebenen Daten. Diagramm 4.3 zeigt die Achitektur des Programms.

Wie zu sehen ist, enthalten die Pakete textimager, twitter, tweets und database Klassen und Funktionen, die für die Systemlogik verantwortlich sind. Sie unterscheiden sich jedoch in ihrem Einsatzgebiet. Twitter wird hauptsächlich zur Verwaltung von Download- und Pooling-Prozessen verwendet. Tweets umfasst Typen, die das Abrufen von Informationen aus der Datenbank erleichtern, database bereitet Daten zum Speichern vor und textimager ermöglicht NLP-Prozesse. Die Funktionen des Controllers werden durch das webapp-Paket erfüllt,

[8] `https://docs.mongodb.com/manual/core/gridfs/`

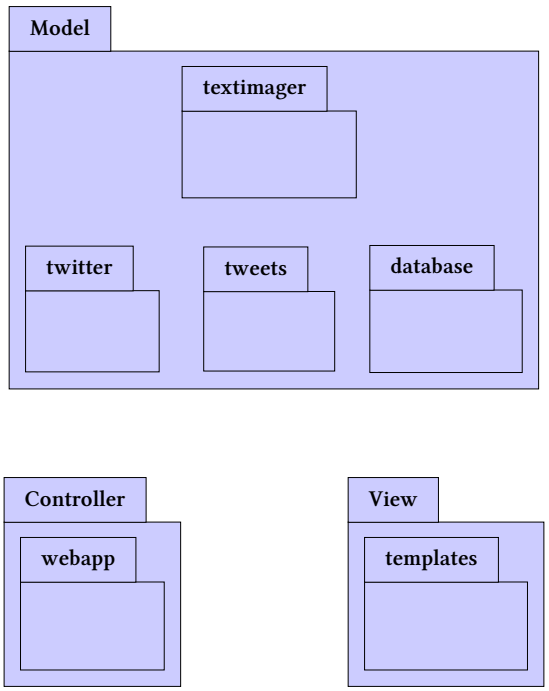

Abbildung 4.3.: Programmstruktur

in dem die Klassen implementiert sind, die die Endpunkte der Anwendung repräsentieren, während templates HTML-Seiten im ftl-Format bereitstellt. Die Aufteilung des Systems ermöglicht nicht nur die Aufrechterhaltung der Ordnung und Übersichtlichkeit des Codes, sondern trennt auch die Logik von der Ansicht, was deren Erweiterung erheblich erleichtert.

4.3.1. Überblick und Ablauf

Die in Abschnitt 3.2 aufgeführten Beispielszenarien stellen auf sehr einfache Weise ein Anwendungsbeispiel aus der Sicht des Benutzers dar. In diesem Absatz erfolgt eine detaillierte Beschreibung der Funktionsweise des Systems. Abbildung 3.1 zeigt, dass das Programm in drei Hauptteile unterteilt werden kann: Tweets downloaden, Pooling und Tweets/Pools als XMI herunterladen.

Das Herunterladen der Tweets ist der Prozess, der für die Vorbereitung, die Ausführung der Abfrage und die Speicherung der Daten in der MongoDB-Datenbank verantwortlich ist. Tabelle 4.4 zeigt alle möglichen Eingaben zum Erstellen und Senden einer Query. Die ersten fünf Felder stehen für die adressierten Endpunkte und die nächsten vier für zusätzliche Attribute. TwitterPooler bietet auch die Möglichkeit, eine zuvor gestoppte Aktivität wieder aufzunehmen. Abbildung 5.1 veranschaulicht die einzelnen Schritte und deren Reihenfolge. Die Ausgangspunkte sind Start und Resume. Die Ellipsen stehen für HTML-Seiten und die anderen Formen für Funktionen.

Tabelle 4.4.: Abfrage-Parameter

Feld	Eingabe
Search	Inhalt der Anfrage. Weiterhin gibt es die Möglichkeit, die Endpunkte Recent und Full-Archive zu wählen.
User Timeline	Benutzer-ID
User Mentioned	Benutzer-ID
User Liked	Benutzer-ID
Tweet Liking	Tweet-ID
Max result	Gewünschte Anzahl der Tweets
Start time	Das Startdatum, ab dem die Tweets geliefert werden sollen.
End time	Das Enddatum, zu dem die Tweets geliefert werden sollen.
NLP Tools	Auftragstools für NLP-Vorverarbeitung. Die Auswahl von mehr als einem Werkzeug ist möglich.

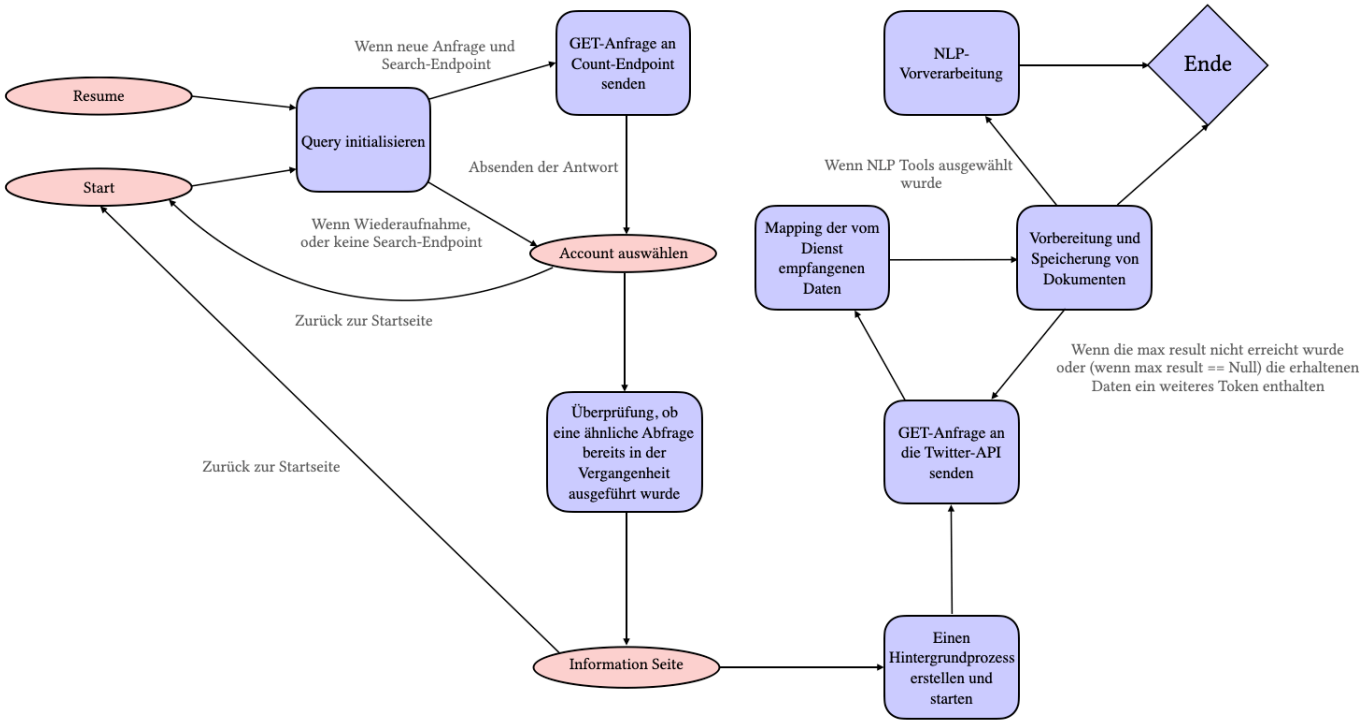

Abbildung 4.4.: Schema des Tweet-Downloads

Pooling ist eine der interessantesten und wichtigsten Funktionen von TwitterPooler. In Abschnitt 2.2 werden die Grundsätze und Beispieltypen des obigen Verfahrens beschrieben. Dieser Prozess ist für das Laden, Sortieren und Speichern der gesammelten Daten in der Datenbank verantwortlich. Wie in Abschnitt 2.2.1 erwähnt, unterstützt die Anwendung vier Arten von Pooling: Author-wise-Pooling, Hashtag-based-Pooling, Temporal-Pooling und Burst-

Score-wise-Pooling. Das Temporal Pooling wurde in Temporal-Days und Temporal-Hours-Pooling aufgeteilt. Außerdem wurden zusätzliche Parameter eingeführt, um mehr Optionen zu bieten:

- Limit: Dieser Parameter gibt die Mindestanzahl von Tweets in der erstellten Kollektion (Pool) an. Alle anderen, die weniger als die angegebene Anzahl von Dokumenten enthalten, werden ignoriert und gespeichert.
- Size: Dieser Parameter steht nur für das Temporal und das Brust-Score-Pooling zur Verfügung. Size definiert den Zeitbereich der erstellten Kollektionen. Wird beispielsweise das Temporal-Days-Pooling gestartet, Size=2 gewählt und davon ausgegangen, dass die Abfrage am 26.02 gesendet wurde, werden Variablen mit folgenden Zeiträumen erzeugt: 26/25.02, 24/23.02, 23/22.02 usw. Dann werden die Tweets nach dem Tag, an dem sie geschrieben wurden, getrennt und der entsprechenden Variable zugeordnet. Schließlich wird jeder dieser Datensätze in ein Format konvertiert, das in MongoDB gespeichert werden kann.

Die folgende Tabelle zeigt alle verfügbaren Pooling-Typen mit den in Kapitel 4.2 genannten zusätzlichen Attributen und dazugehörigen Präfixen. Anstelle der Platzhalter sind die Werte der entsprechenden Parameter eingefügt.

Tabelle 4.5.: Pooling-Arten

Art	Limit (L)	Size (S)	Prefix
Author-wise Pooling	Ja	Nein	AL_
Hashtag-based Pooling	Ja	Nein	HL_
Temporal-days Pooling	Ja	Ja	TDaysS_L_
Temporal-hours Pooling	Ja	Ja	THoursS_L_
Burst-score wise Pooling	Ja	Ja	BL_

Zwei der obengenannten Typen benötigen weitere Erläuterungen, nämlich Temporal-Hours und Brust-Score-wise-Pooling. Temporal-Hours bezeichnet die Aufteilung der Tweets nach dem Zeitpunkt ihrer Veröffentlichung. In diesem Fall gibt es jedoch strenge Regeln für ihre Erstellung. Jeder Tag wird im Voraus in bestimmte Zeitabschnitte unterteilt. Tabelle 4.6 zeigt alle verfügbaren Optionen mit den entsprechenden Zeiträumen:

Tabelle 4.6.: Temporal-Hours-Pooling

Wert	Zeitraum
1	Jede Stunde
2	00-02, 02-04, 04-06, 06-08, 08-10, 10-12, 12-14, 14-16, 16-18, 18-20, 20-22, 22-24
4	00-04, 04-08, 08-12, 12-16, 16-20, 20-24
6	00-06, 06-12, 12-18, 18-24
8	00-08, 08-16, 16-24
12	00-12, 12-24

Brust-Score-wise ist eine der kompliziertesten Arten von Pooling, die in der Anwendung implementiert wurden. Deshalb bedarf es einer separaten Erklärung. In Abschnitt 2.2.1 findet sich eine Erläuterung des Grundprinzips seiner Funktionsweise. Wegen der Notwendigkeit, die Popularität der vom Benutzer angegebenen Schlüsselwörter zu berechnen, wurde ein zusätzliches Feld zur Verfügung gestellt. Es können beliebig viele Wörter eingegeben werden, die jedoch durch ein Leerzeichen getrennt sein müssen. Der Parameter Limit steht in diesem Fall für den Beliebtheitswert und Size für den Zeitraum, für den er berechnet werden soll. Diagramm 4.5 zeigt die Schritte und den Ablauf des Pooling-Prozesses.

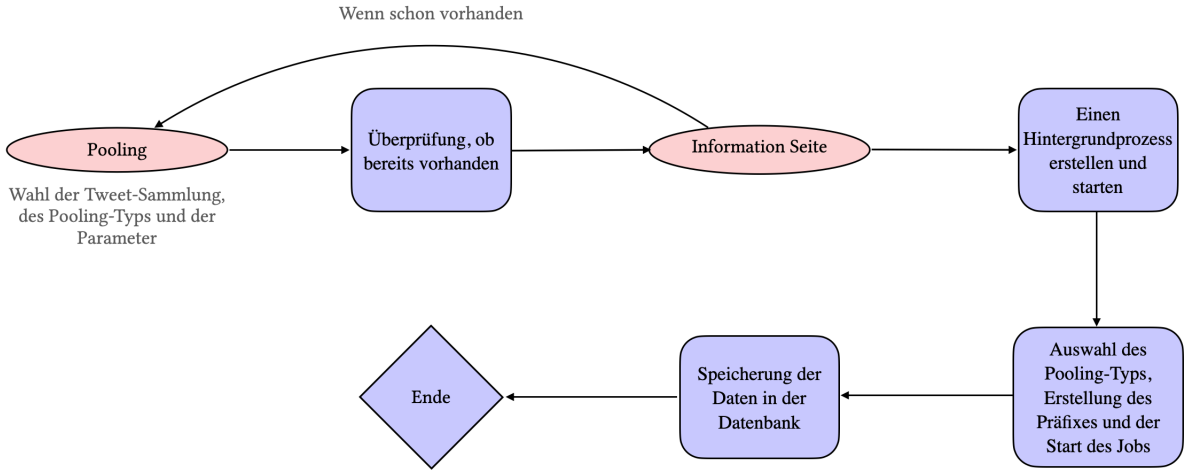

Abbildung 4.5.: Schema des Pooling-Prozesses

TwitterPooler bietet auch die Möglichkeit, gespeicherte Daten im XMI-Format auf die lokale Festplatte herunterzuladen. Das bedeutet, dass der Inhalt jeder fertigen Abfrage oder jedes Pools in das XMI-Format konvertiert und gespeichert werden kann. Es gibt zwei Möglichkeiten, solche Dateien vorzubereiten. Die erste besteht darin, alle zusammenhängenden Tweets in einer XMI zu speichern. Die zweite Variante ist, sie separat zu erstellen und in ein Zip-Paket zu packen. Die Phasen des gesamten Prozesses sind in Abbildung 4.6 dargestellt.

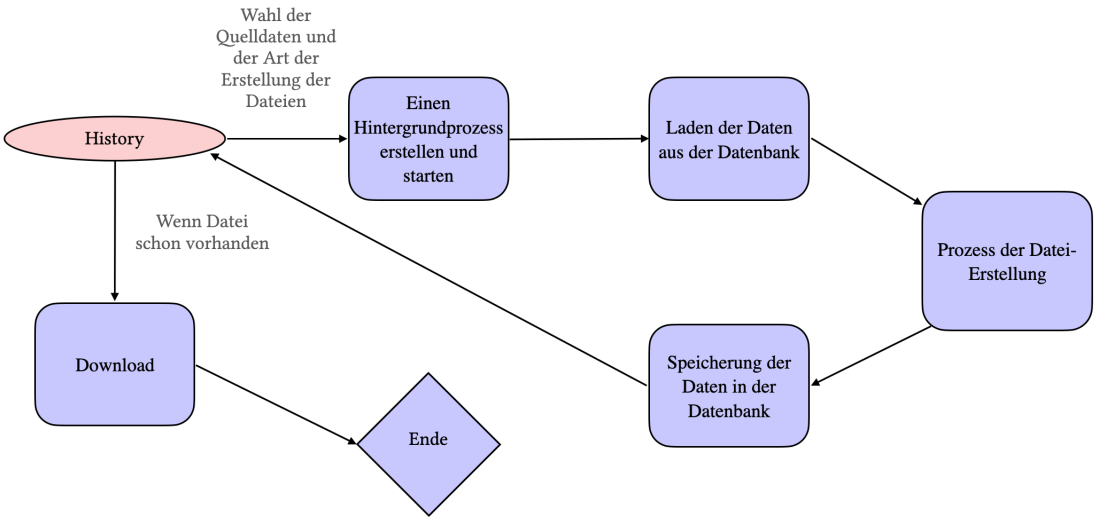

Abbildung 4.6.: XMI-Download-Schema

4.3.2. Klassendiagramm

Das folgende Klassendiagramm 4.7 veranschaulicht die implementierten Klassen und die Beziehungen zwischen ihnen. Aufgrund der Komplexität und des Platzmangels werden im Diagramm die zugehörigen Attribute und Funktionen nicht dargestellt. Die Anwendung besteht aus 23 Klassen, die in fünf Pakete unterteilt sind. In twitter sind Typen implementiert, die für die Kommunikation und das Lesen von Daten aus der Datenbank zuständig sind. Jeder von ihnen ist eine Erweiterung von PanacheMongoEntity[9], die den Zugriff auf MongoDB-Dokumente erheblich erleichtert. Database besteht aus DBHelper-Klasse, die empfangene Tweets zum Speichern vorbereiten. Zu den wichtigsten gehören das Mapping von Daten aus Includes und die Erstellung von JCas-Objekten. Das Paket tweets enthält Klassen, die Hintergrundprozesse für das Pooling und Herunterladen von Tweets erstellen und verwalten. Sie werden unter Verwendung der Java-Schnittstellen ExecutorService[10] und Runnable[11] implementiert. Webapp stellt die verfügbaren Endpunkte dar. Ihre Aufgabe ist der Datenaustausch zwischen der Ansicht und dem Systemmodell. Die Parameterübergabe erfolgt direkt an die URL, unter Verwendung des Interfaces UriInfo[12] und der Freemarker-Bibliothek[13]. Das textimager-Paket bietet die notwendige Funktionalität, um NLP-Vorverarbeitung durchzuführen. Für die Verbindung zur TextImager-Client API [14] wird der TextImagerClient und für die Erstellung des Hintergrundjobs der TextimagerRunnable verwendet.

[9] https://quarkus.io/guides/mongodb-panache
[10] https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ExecutorService.html
[11] https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html
[12] https://docs.oracle.com/javaee/7/api/javax/ws/rs/core/UriInfo.html
[13] https://freemarker.apache.org/
[14] https://github.com/texttechnologylab/textimager-client

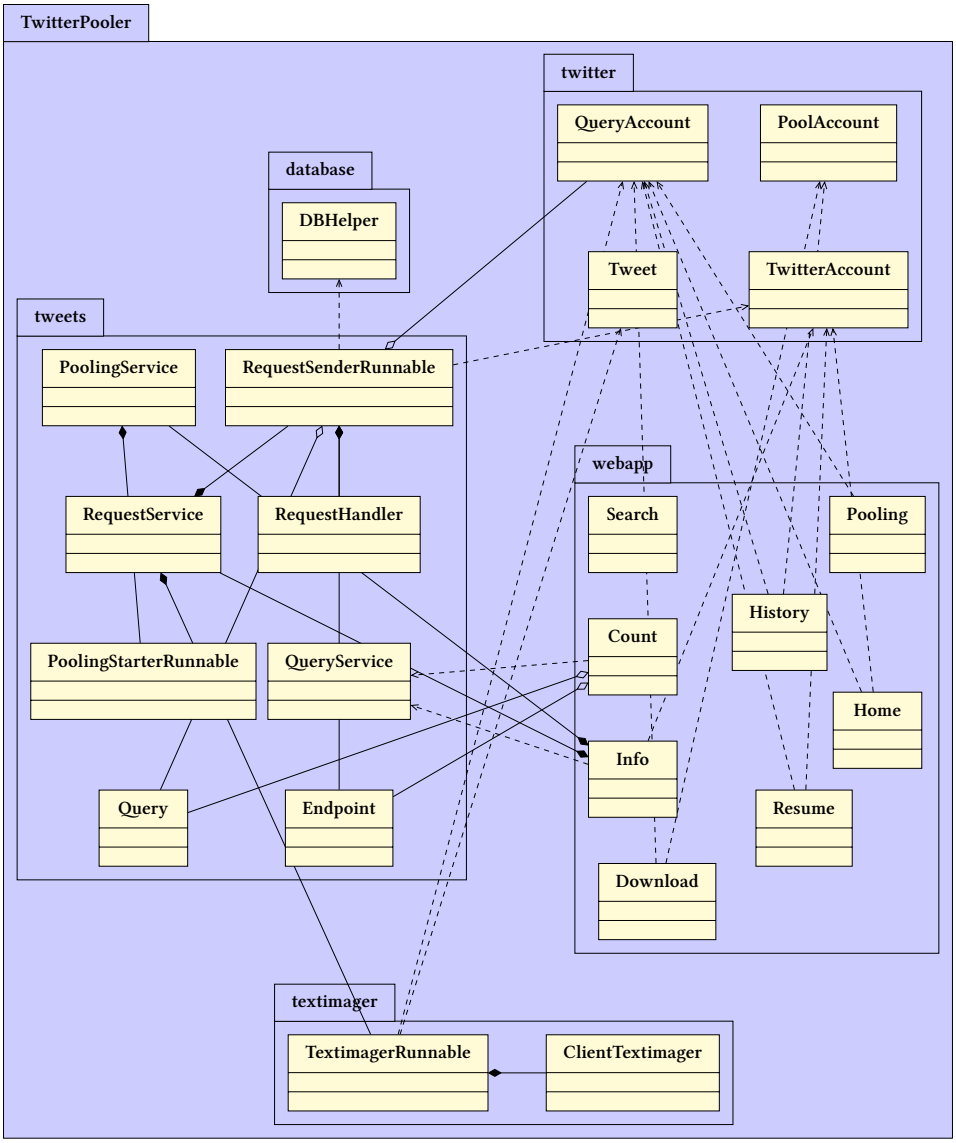

Abbildung 4.7.: Klassendiagramm

4.3.3. Übersicht über wichtige Funktionen

TwitterPooler ist ein komplexes Programm mit einer großen Anzahl von Funktionen. Im Folgenden werden drei Beispielfunktionen vorgestellt.

Die erste davon ist toCas(), die in der DBHelper-Klasse zu finden ist. Diese Funktion hat die Aufgabe, aus den in der übermittelten Nachricht enthaltenen Informationen ein JCas-Objekt zu erzeugen. Der Aufruf erfolgt im Rahmen der Erstellung eines Dokuments, das den Tweet darstellt und anschließend in der Datenbank gespeichert wird. Zu Beginn der Funktion wird der Inhalt des Beitrags bereinigt. Mit Hilfe von Regex werden alle Zeichen entfernt, die einen negativen Einfluss auf die nachfolgende NLP-Vorverarbeitung haben könnten. Danach folgt die Initialisierung eines JCas-Objekts, das den zuvor bereinigten Text und einen speziell

erstellten UIMA-Annotator-Typ-Tweet enthält.

Bei StartHashtagPooling() handelt es sich um eine Funktion, die für die Erstellung von Pools in Bezug auf Hashtags verantwortlich ist. Die aufrufende Klasse ist PoolingStarter-Runnable und der Eingabewert ist eine MongoDB-Kollektion. Die Idee der Funktion besteht darin, alle Dokumente in der Sammlung zu durchlaufen und sie zur initialisierten Datenstruktur HashMap<String, ArrayList<Document» hinzuzufügen. HashMap speichert Einträge in Form von Schlüssel/Wert-Paaren und eignet sich daher gut für Aufgaben, die z. B. darauf abzielen, Elemente voneinander zu trennen. Der in der Struktur implementierte Schlüssel ist der Hashtag, während der Wert ein Array ist, das Dokumente von Tweets speichert. Der gesamte Aggregationsvorgang besteht aus zwei Schleifen. Die äußere durchläuft alle Objekte in der Datenbanksammlung und die innere alle Hashtags darin. Am Ende gibt die Funktion die erstellte HashMap zurück. Auf sehr ähnliche Weise funktioniert startAuthorWisePooling(), welches Pools entsprechend der Id des Autors erstellt.

StartTemporalDaysPooling() liefert eine Aufteilung der Tweets nach dem Tag der Veröffentlichung. Der Eingabewert ist eine Mongodb-Kollektion und eine Ganzzahl, die den Size-Parameter darstellt (siehe Abschnitt 4.3.1). Der Aufbau der Funktion ist sehr ähnlich zu StartHashtagPooling(). Sie besteht jedoch nur aus einer einzigen Schleife, die die empfangenen Dokumente durchläuft. Der Schlüssel der HashMap ist ein String, der das Jahr und den Tag der Erstellung der Tweets angibt. Dieser wird mitells der Funktion createTempDaysKey() erzeugt. Die Eingangswerte von createTempDaysKey() sind das Datum und der Parameter Size. Das vorhandene Datum wird in eine ganze Zahl umgewandelt, die einen Tag des Jahres darstellt (getDayOfYear()[15]). Dann wird in einer for-Schleife eine Liste von Werten (für die Tage eines Jahres) in der Form i + Größe erstellt, beginnend mit i=1 (siehe Listing 4.1). Am Ende werden (dank der zweiten for-Schleife) die entsprechenden Tage ausgewählt, aus denen ein Schlüssel der Form "_Jahr_-_erster Tag_/_letzter Tag_ërstellt wird. StartTemporalDaysPooling() liefert ähnlich wie andere Funktionen, eine HashMap mit sortierten Einträgen.

Listing 4.1: CreateTempDaysKey()

```
int dayOfYear = date.getDayOfYear();
List<Integer> keys = new ArrayList<>();
for (int i = 1; i<date.lengthOfYear(); i=i)
{
    if(i==1)
    {
        keys.add(i);
    }
    i = i + size;
    keys.add(i);
}
```

[15] https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html

4.3.4. Design

Dieser Abschnitt befasst sich mit der graphischen Ebene der Anwendung. TwitterPooler besteht aus sieben Webseiten. Zwei davon, die Home und Search-Seite, werden im Folgenden vorgestellt.

Die Startseite des Systems ist Home. Hier befinden sich Tabellen, die über den aktuellen Status von Pooling-Prozessen, Tweet-Downloads und den Status von Entwicklerkonten informieren. Des Weiteren wird sie verwendet, um zwischen anderen Adressen zu navigieren. Abbildung 4.8 zeigt den Aufbau.

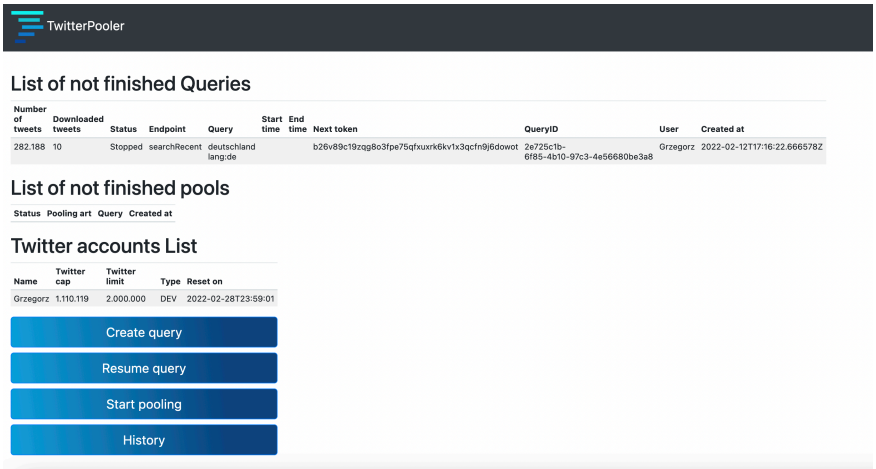

Abbildung 4.8.: Homeseite

Auf der Search-Seite 4.9 stellt der Benutzer eine Query zusammen. Die Umsetzung ist so konzipiert, dass die Angabe von falschen Informationen möglichst vermieden wird. Daher werden nach der Eingabe eines Wertes manche Felder deaktiviert, um das Senden fehlerhafter Abfragen zu minimieren.

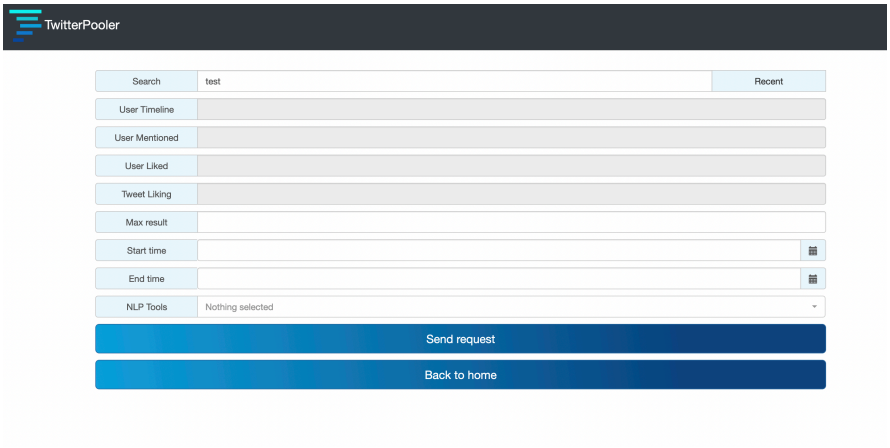

Abbildung 4.9.: Searchseite

5. Evaluation

In diesem Kapitel werden die Ergebnisse der Evaluierung des Programms TwitterPooler vorgestellt. Es besteht aus drei Unterkapiteln. Im ersten Teil wird die Anwendung im Hinblick auf die Zeit analysiert, die für das Herunterladen von Tweets und den Pooling-Prozess benötigt wird. Weiterhin werden das Volumen der erstellten Pools und die NLP- Vorverarbeitung untersucht. Im zweiten Teil wird das System mit dem alternativen kostenlosen Tool twarc verglichen. Der dritte Teil ist der Diskussion der erzielten Ergebnisse gewidmet. Ziel der Analyse ist es, Erkenntnisse zu gewinnen, die es ermöglichen, das Programm von der technischen Seite her zu bewerten. Für die Durchführung der Tests und die Darstellung der Ergebnisse wurden TwitterPooler, twarc und die Programmiersprache R verwendet.

5.1. Evaluation des Systems

Um die Laufzeitanalyse durchzuführen, wurden 25 separate Prozesse zum Download von Tweets realisiert. Sie unterschieden sich entweder in der Menge der Daten oder in der Durchführung der NLP-Vorverarbeitung, für die das von Text Technology Lab bereitgestellte Programm SpacMultitaggerv3 verwendet wurde. Abbildung 5.1 zeigt die Ergebnisse des Tests. Bei dem schnellsten der fünf vorgestellten Verfahren wurde keine NLP-Vorverarbeitung angewandt. Die Bearbeitung von 16 000 Beiträgen dauerte 84 Minuten. Die rot markierten Punkte stehen für das Herunterladen von Nachrichten mit SpacMultitaggerv3. Die Texte der Tweets wurden jedoch sequentiell mit nur einem Thread verarbeitet. Die gesamte Prozedur dauerte daher wesentlich länger und erforderte 254 Minuten. Um bessere Ergebnisse zu erzielen, wurde eine Verbesserung vorgenommen und eine Lösung für die parallele Verarbeitung von NLP implementiert. Die anderen drei Punkte stehen für diese Methoden. So wurde die Dauer des Prozesses (mit 16 000 Tweets) im Falle von zwei Threads auf 206, vier Threads auf 128 und sechs Threads auf 136 Minuten reduziert. Es zeigte sich, dass der Einsatz von NLP-Werkzeugen eine erhebliche Wirkung hat. Ohne parallele Verfahren musste der Benutzer dreimal so lange auf die erwarteten Ergebnisse warten. Die Unabhängigkeit der vorbereiteten Texte erlaubt jedoch die problemlose Nutzung der Gleichzeitigkeit, was im Falle der Verwendung von vier Threads eine Reduzierung der Wartezeit um fast 50% ermöglichte. Interessant ist auch die Tatsache, dass vier Threads effizienter sind als sechs. Dies hängt jedoch mit der Hardware zusammen, auf der die Tests durchgeführt wurden. Konkret handelt es sich um die Anzahl der Prozessorkerne (Apple Air M1, 8Cores). Daher sollten bei der Verwendung von Prozessoren mit einer höheren Kernzahl noch bessere Ergebnisse erzielt werden.

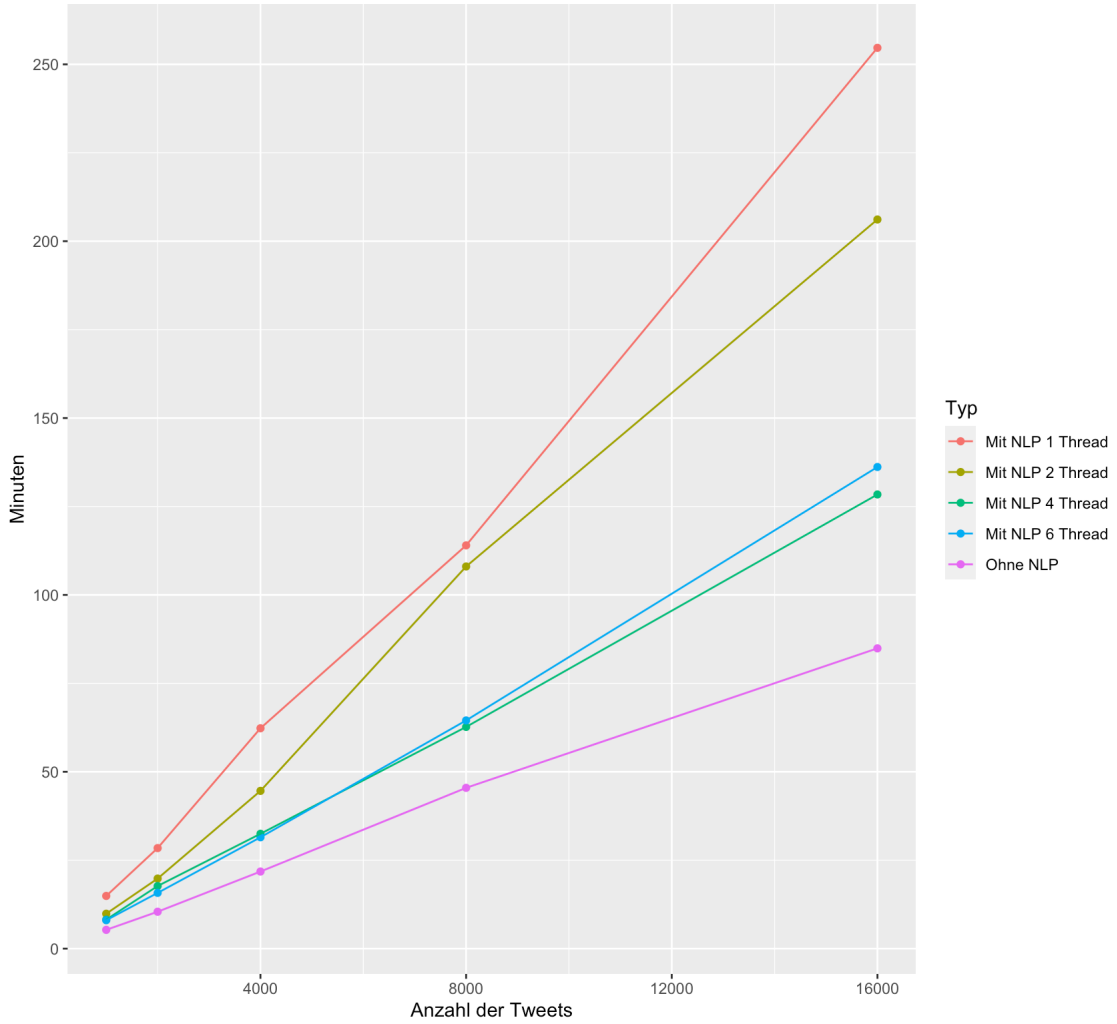

Abbildung 5.1.: Tweet-Download

Das Pooling ist eine der wichtigsten Funktionen des TwitterPoolers und erfordert daher eine genauere Analyse. In diesem Abschnitt werden die Ergebnisse des Geschwindigkeitstests zur Erstellung von Pools am Beispiel von zwölf Arten vorgestellt. Jeder Prozess wurde auf sechs unterschiedliche Kollektionen von Tweets durchgeführt, die das Schlüsselwort frankfurt enthielten. Die Grafik 5.2 zeigt die Ergebnisse der Tests.

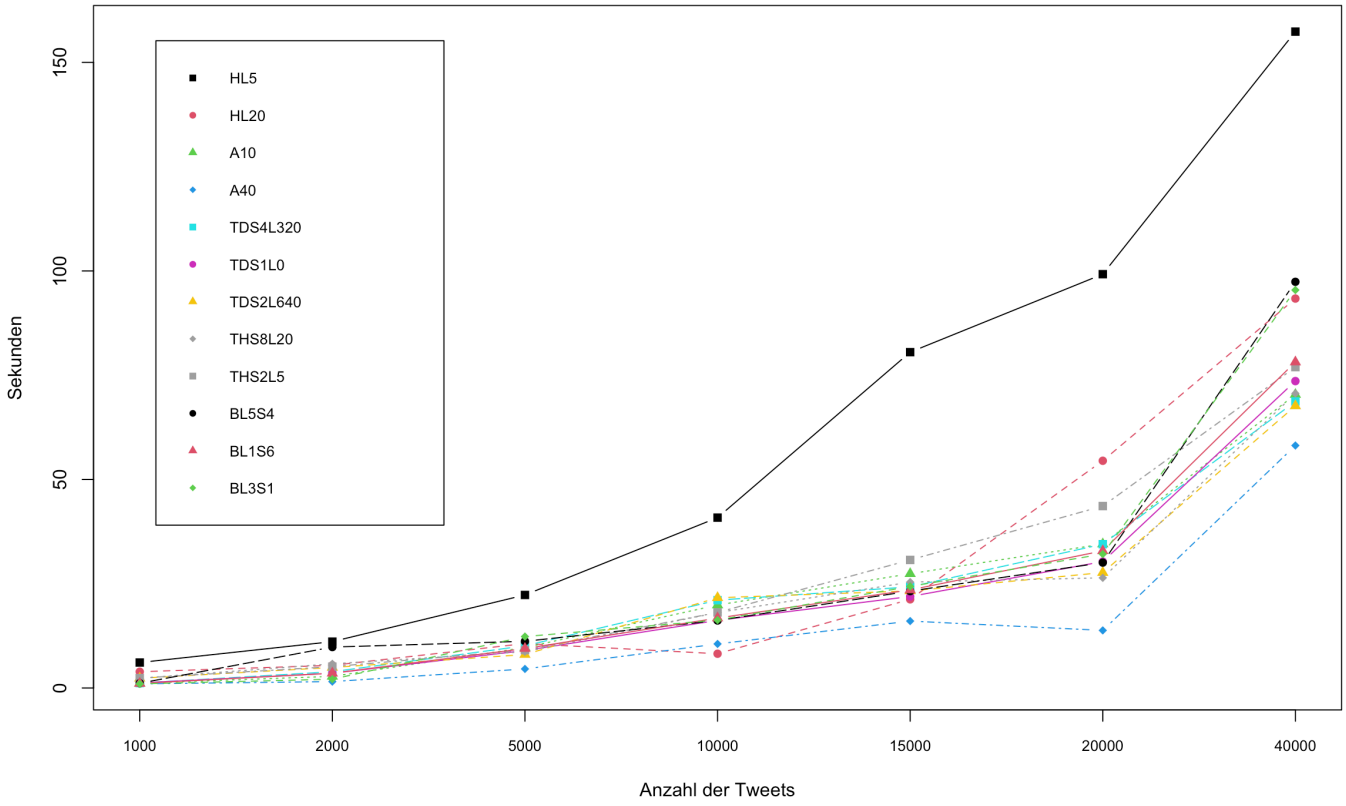

Abbildung 5.2.: Pooling-Zeitmessung

Der aufwändigste Prozess ist das Hashtag-Pooling. Bei der Bearbeitung von 40 000 Einträgen und dem niedrigen Parameter L=5 muss der Benutzer etwa 157 Sekunden warten. Alle anderen Verfahren haben mehr oder weniger ähnliche Ergebnisse von 60-70 Sekunden, obwohl die Laufzeit der Burst-Score-wise-Pooling-Methode bei größeren Datenmengen deutlich ansteigt. Um die Ursachen besser zu verstehen, sollte auch ein Blick auf das Volumen der erstellten Sammlungen geworfen werden, das in Abbildung 5.5 dargestellt ist. Alle getesteten Methoden zur Aggregation nach Zeit oder Datum zeichnen sich durch die größte Anzahl von Datensätzen aus. Auch das Hashtag-Pooling zeigt einen höheren Gehalt an Tweets und bei 40 000 fällt ein deutlicher Sprung von BL3S1 und BL5S5 auf. Ein Blick auf beide Diagramme deutet jedoch darauf hin, dass nicht nur das Volumen einen Einfluss auf die Bearbeitungszeit hat. Dies zeigt sich an den Punkten, die Temporal-Pooling (siehe Abbildung 5.2) bezeichnen. Sie befinden sich mehr oder weniger an denselben Positionen wie die anderen Typen, die sich durch wesentlich geringere Datenmengen auszeichnen. Einige von ihnen stehen für eine sehr kleine Anzahl von enthaltenen Nachrichten, wie A10 oder BL1S6. Die beiden teuersten Pooling-Typen (Hashtag und Burst-Score) unterscheiden sich durch die zusätzliche Iteration innerhalb jedes Dokuments, die einen entscheidenden Aspekt der Prozessdauer dartsellt.

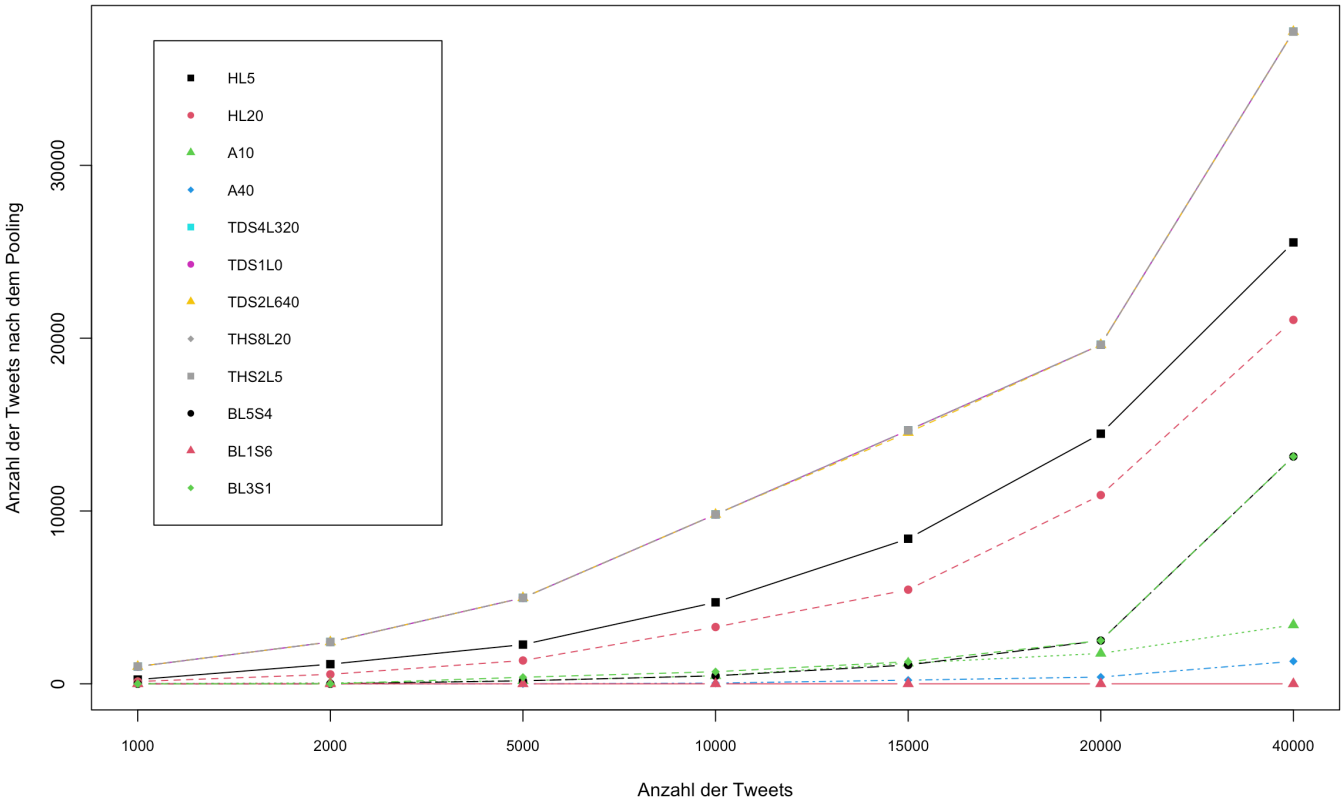

Abbildung 5.3.: Pooling-Anzahl der Tweets

Anschließend wurden die Funktionen analysiert, mit denen die Dokumente für die Vorverarbeitung vorbereitet werden. Eine der wichtigsten Aufgaben der Anwendung ist es, ein kompatibles Format (JCas-Objekt) zu erstellen, das die Verwendung der zahlreichen NLP-Tools von TextImager ermöglicht. Für die Erstellung dieser Objekte wurden die Klasse JCasFactory (UIMA-Framework[1]) und die Klasse MongoSerialization (UIMADatabaseInterface[2]) verwendet. Zur besseren Veranschaulichung der Ergebnisse wurde das gesamte Verfahren in zwei Teile unterteilt. Die erste zeigt alle Funktionen zusammen mit ihrer Ausführungszeit beim Tweet-Download. Die zweite hingegen bei der NLP-Analyse. Für die Tests wurden zehn Tweets abgerufen, wobei die Zeiten die Summe aller durchgeführten Abläufe darstellen. Die Tabellen 5.1 und 5.2 stellen die Ergebnisse der gesammelten Beobachtungen dar. Diese zeigen, dass die Erstellung eines JCas-Objekts am aufwendigsten ist. JCasFactory.createText() benötigt im Durchschnitt 778,1 ms, während JCasFactory.createJCas() 621,4 ms braucht. Die Serialisierung des obigen Elements (die notwendig ist, um in MongoDB zu speichern) ist ebenfalls eine zeitaufwändige Aufgabe. Für das Abrufen von Tweets selbst wird 18% der Gesamtzeit verbraucht. Darüber hinaus ist anzumerken, dass allein der Teil der Datenabfrage

[1] https://uima.apache.org/d/uimafit-current/api/org/apache/uima/fit/factory/JCasFactory.html
[2] https://github.com/texttechnologylab/UIMADatabaseInterface

(ohne NLP-Vorverarbeitung) insgesamt 14405 ms in Anspruch nahm. Das bedeutet, dass 72% der entstandenen Kosten mit der Erstellung des JCas-Objekts und seiner Konvertierung in ein für die Datenbankspeicherung geeignetes Format zusammenhängen.

Tabelle 5.1.: Tweet-Download

Funktion	Zeit (ms)	Durchschnitt (ms)
JCasFactory.createText(Tweet.Text, lang)	7781	778,1
MongoSerialization.serializeJCas(JCas)	2602	260,2
Summe	10383	1038,3

Tabelle 5.2.: NLP-Vorverarbeitung

Funktion	Zeit (ms)	Durchschnitt (ms)
JCasFactory.createJCas()	6214	621,4
MongoSerialization.deserializeJCas(JCas)	394	39,4
NLP Prozess	1842	184,2
MongoSerialization.serializeJCas(JCas)	1098	109,8
Summe	9548	954,8

Die Grafik 5.4 zeigt ein Beispiel für einen serialisierten JCas. Dieses Objekt stellt einen Tweet dar und enthält die Ergebnisse der von SpacyMultitagger3 durchgeführten Analyse. Aufgrund seines Umfangs wurde ein Großteil des Inhalts entfernt. Es wurden nur die erforderlichen Informationen zur Veranschaulichung der Struktur und der Ergebnisse der NLP-Vorverarbeitung stehen gelassen. Der rote Bereich beschreibt die verwendeten Werkzeuge. Die farbigen Blöcke darunter veranschaulichen Beispiele für die Komponenten, die bei der Verarbeitung entstehen, wie Token, Part of Speech (POS), Lemma oder Named Entity.

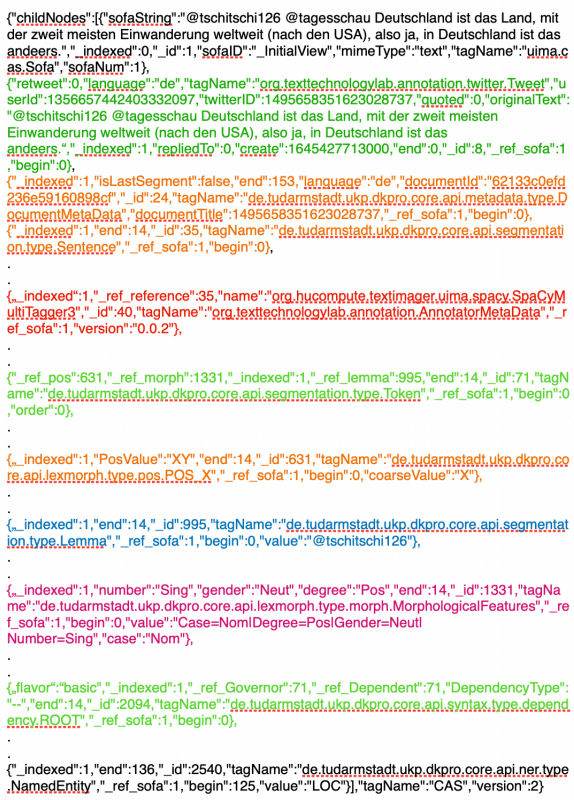

Abbildung 5.4.: JCas Objekt

5.2. Vergleich mit twarc

In diesem Abschnitt wird ein Vergleich zwischen TwitterPooler und einem alternativen Programm durchgeführt. Aufgrund des Mangels an Systemen, die NLP-Vorverarbeitung und Pooling anbieten, erfolgt ausschließlich eine Konzentration auf die Funktion des Tweet-Downloads. Zur Durchführung des Vergleichs wurde die Anwendung twarc gewählt. Twarc[3] ist ein Open-Source-Konsolentool, das hauptsächlich zum Herunterladen von Tweets verwendet wird. Die Anwendung wurde in Python geschrieben und ist mit der neuesten Version der Twitter API v2 kompatibel. Außerdem ist sehr populär, was die breite Anzahl von Mitwirkenden auf GitHub[4] bestätigt. Seine Installation erfordert keine große Konfiguration und die Bedienung ist nicht kompliziert. Einen der wichtigsten Vergleichspunkte stellt die Abrufgeschwindigkeit dar. Zu diesem Zweck wurden sieben Verfahren mit einer unterschiedlichen Anzahl von Tweets getestet. Abbildung 5.5 zeigt die Resultate der Tests. Die blauen Punkte stehen für die Ergebnisse des TwitterPoolers. Der Unterschied ist signifikant. Bei 4 000 Tweets betrug die Wartezeit auf Daten beispielsweise 21,795 Minuten (TwitterPooler) und 0,935 Minuten (twarc). Dies bedeutet, dass die im Rahmen der vorliegende Arbeit

[3] https://twarc-project.readthedocs.io/en/latest/

[4] https://github.com/DocNow/twarc

erstellte Anwendung um das 23,3-Fache langsamer war. Ähnliche Ergebnisse sind im untenstehenden Diagramm zu sehen, wo bei einer Anzahl von 16 000 und 32 000 Tweets die Geschwindigkeit von TwitterPooler 25-mal langsamer war und der Unterschied zu twarc damit sogar noch größer ausfiel. Optimistischerweise wurde der Zeitabstand bei 64 000 jedoch deutlich reduziert (TwitterPoller war 16-mal langsamer).

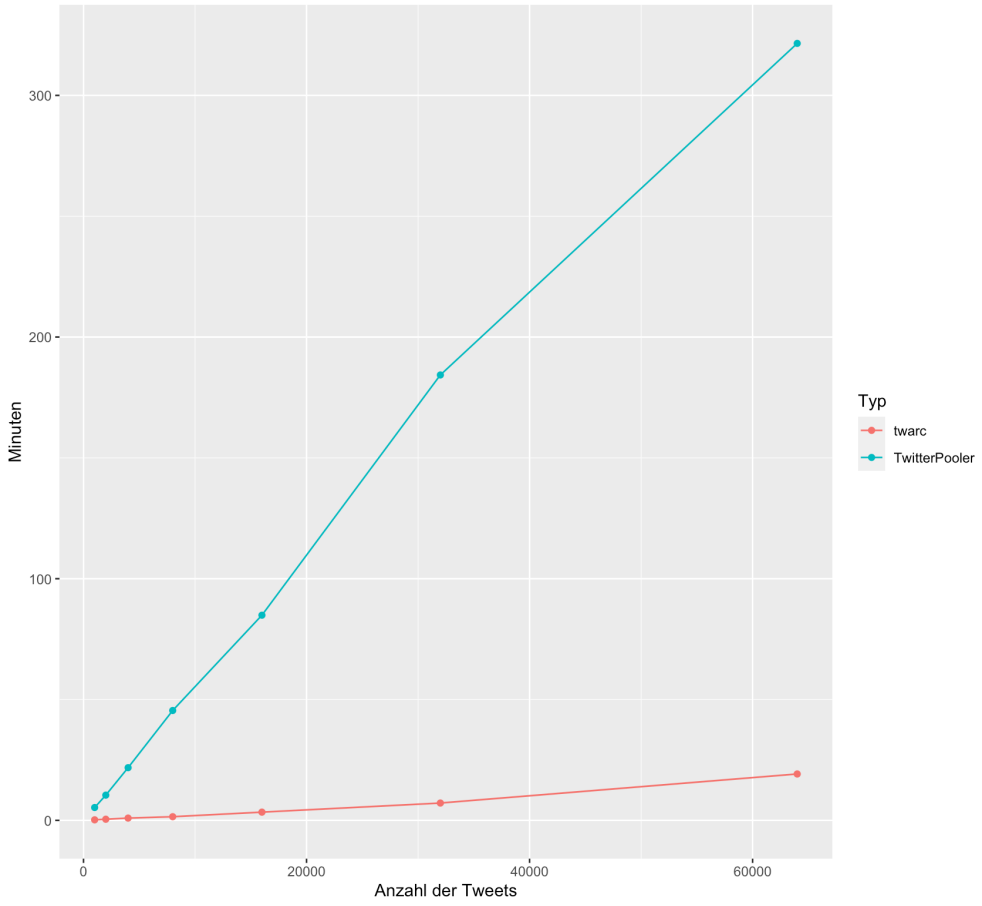

Abbildung 5.5.: Tweet-Download Vergleich mit twarc

Da die Unterschiede bei den Download-Geschwindigkeiten zwischen den vorgestellten Anwendungen deutlich sind, sollten die Ursachen dafür ermittelt werden. Der größte Faktor, der die Dauer des TwitterPooler-Downloads beeinflusst, ist die Aufbereitung der Dokumente für die NLP-Vorverarbeitung. Aus der Interpretation der Ergebnisse in Abschnitt 5.1 wurde entnommen, dass die Erstellung eines JCas-Objekts und dessen Serialisierung etwa 72 % der Gesamtzeit in Anspruch nimmt. Um einen genaueren Vergleich anstellen zu können, müssen alle mit NLP verbundenen Kosten abgezogen werden. Das liegt daran, dass twarc keine Funktionen in diesem Bereich anbietet. Nach deren Entfernung ist die TwitterPooler-Anwendung bei 4 000 Tweets nur noch 5,8-mal und bei 64 000 4-mal langsamer. Obwohl sich der Abstand erheblich verkürzt hat, ist das alternative Programm weiterhin deutlich leistungsfähiger. Dies ist vor allem auf die fehlende Zuordnung zusätzlicher Responsedaten (die in Includes ent-

halten sind) und deren Speicherung in der Datenbank zurückzuführen. Das Auffinden und Zuordnen von Informationen aus Includes ist ein teures Verfahren, das viele Iterationen erfordert. Zur Veranschaulichung des Zuordnungsverfahrens und der Unterschiede zwischen den Endergebnissen dienen die folgenden Abbildungen 5.6 und 5.7. Grafik 5.6 zeigt eine von der Twitter-API v2 empfangene Beispielantwort im JSON-Format (links), die aufgrund ihrer Länge so gekürzt wurde, dass sie nur eine Tweet-Nachricht darstellt. Die rechte Seite stellt die formatierte Entsprechung dar, die in einer MongoDB-Datenbank gespeichert ist. Es handelt sich um genau denselben Tweet wie auf der linken Seite. Sie enthält jedoch zusätzliche Informationen von Includes (in diesem Fall über den Autor des Beitrags).

Abbildung 5.6.: TwitterPooler-Mapping

Die twarc-Anwendung verfügt nicht über eine eingebaute Mapping-Funktion. Die abgerufenen Daten werden in einer .jsonl-Datei gespeichert, und zwar genau im gleichen Zustand wie die empfangene Rückmeldung. Bei einer großen Anzahl von Antworten (Twitter API v2 sendet maximal 100 pro Anfrage) werden diese auch nicht zusammengefasst. Die endgültige .jsonl-Datei entspricht einer Liste aller Anfragen (siehe Abbildung 5.7). Offensichtlich müssen die resultierenden Daten weiter bearbeitet werden, was den Einsatz zusätzlicher Werkzeuge erfordert.

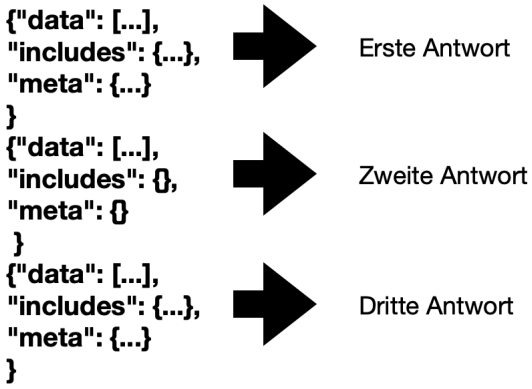

Abbildung 5.7.: Twarc: gespeicherte Daten

5.3. Diskussion

Die Aufgabe, ein System zum Herunterladen von Tweets zu entwickeln, das auch die NLP-Vorverarbeitung und Pooling anbietet, wurde erfolgreich abgeschlossen. Mit quantitativen Methoden wurden die Auswirkungen zusätzlicher Funktionen auf die Leistung der Anwendung untersucht. Die Ergebnisse der Tests zeigen einen großen Einfluss der Vorverarbeitung auf die Geschwindigkeit des Downloads von Tweets und der Vergleich mit dem Alternativprogramm twarc zeigte einen großen Unterschied in der Wartezeit auf Daten von Twitter. Bei einer genaueren Analyse der Pooling-Funktion wurde festgestellt, dass Verfahren, die mehr Iterationen (und nicht die endgültige Anzahl der gespeicherten Dokumente) erfordern, deutlich langsamer sind.

Dank der implementierten Lösungen bietet TwitterPooler der Community viele Annehmlichkeiten, die sich positiv auf die Verwaltung der von Twitter heruntergeladenen Daten auswirken. Nur sehr wenige der Anwendungen dieser Art, bieten dem Benutzer eine zusätzliche grafische Schnittstelle. Ein Beispiel wurde bereits in Kapitel 3.1 mit dem Twitter-Explorer erwähnt. Dieses Programm bietet eine Seite zum Senden von Anfragen an, die jedoch recht spärlich ist und nur die zur Konfiguration notwendigen Felder enthält (vgl. Pournaki u. a. 2020, S. 114). Das zweite Beispiel stellt das ebenfalls bereits beschriebene DMI-TCAT-Tool dar. Im Vergleich zum Twitter-Explorer ist er mit einer reichhaltigen Weboberfläche ausgestattet, die aus vielen für den Abruf und die Analyse von Daten zuständigen Modulen besteht (vgl. Borra und Rieder 2014, S. 266–274). Es fehlen jedoch die Möglichkeit, Entwicklerkonten von Twitter zu verwalten und Funktionen, um zu prüfen, ob eine bestimmte Abfrage in der Vergangenheit ausgeführt wurde. Darüber hinaus bietet keines der erwähnten oder im Zuge der Erstellung dieser Arbeit gefundenen Programme die Möglichkeit des Poolings und der NLP-Vorverarbeitung. Alle oben genannten Punkte sprechen für die hohe Attraktivität von TwitterPooler. Die Ergebnisse der Systemleistungsstudie zeigen, dass die Auswirkungen der zusätzlichen Funktionen erheblich sind. Der Einsatz von NLP-Tools führt zu doppelt so langen Wartezeiten auf Daten. Erfreulicherweise wirkte sich die Verwendung von parallelen Threads positiv aus und führte zu einer wesentlich besseren Leistung. Dies ist eine

gute Nachricht, denn sie zeigt, dass die Geschwindigkeit der NLP-Verarbeitung durch den Einsatz von mehr Prozessorkernen erhöht werden kann. Der Vergleich der Anwendung mit der twarc ergab keine positiven Ergebnisse. Tests, die mit sieben verschiedenen Abfragen durchgeführt wurden, zeigten die Schwäche von TwitterPooler. Die Wartezeit für Tweets (ohne NLP-Analyse) war im schlimmsten Fall 25-mal höher und im besten Fall 16-mal höher als bei der Verwendung des Twarc-Systems. Die Gründe für diesen großen Unterschied liegen in der Vorbereitung der Dokumente für das NLP, dem Auftreten von Informationsmapping und der Erfassung in der Datenbank. Dies ist ein weiterer Beweis dafür, wie stark zusätzliche Funktionen die Leistung des Programms beeinträchtigen. Trotz der höheren Anzahl an Features ist die Diskrepanz jedoch zu groß, und es sollten weitere Arbeiten zu diesem Thema in Angriff genommen werden. Die in Kapitel 5 durchgeführten Untersuchungen umfassten auch einen Vergleich der verschiedenen Pooling-Methoden. Die Erstellung von Pools erwies sich als weniger zeitaufwändig als das Herunterladen von Tweets. Dies wird in den Abbildungen 5.1 und 5.2 deutlich, wo auf den ersten Blick ein signifikanter Unterschied zu erkennen ist (die Zeit für das Herunterladen von Tweets wird in Minuten gezählt, während das Pooling in Sekunden erfolgt). Tests haben auch gezeigt, dass Hashtag und Burst-Score-wise-Pooling zu den anspruchsvollsten Verfahren mit den längsten Entwicklungszeiten gehören. Pooling ist aufgrund seiner Geschwindigkeit und seines breiten Anwendungsspektrums (siehe Abschnitt 2.2.2) eine interessante Ergänzung, die die Attraktivität des Programms erheblich steigert.

Die in diesem Kapitel durchgeführten Analysen haben sich nicht mit Multithreading befasst. Alle Prozesse (sowohl Pooling als auch Tweet-Abruf) wurden nacheinander durchgeführt. Das in dieser Arbeit entwickelte System ermöglicht jedoch die gleichzeitige Ausführung mehrerer Aufträge, so dass die Untersuchung ihrer Leistung sicherlich interessante Ergebnisse liefern und eine Weiterentwicklung des Programms ermöglichen wird.

6. Zusammenfassung

6.1. Fazit

Das Ziel dieser Arbeit besteht darin, ein System zu entwickeln, das die Sammlung von Daten aus Twitter erleichtert. Außerdem sollte es über innovative Funktionalitäten wie NLP-Vorverarbeitung und Pooling verfügen. Einen wichtigen Aspekt stellte auch die Überprüfung der Auswirkungen der implementierten Funktionalitäten auf die Leistung der Anwendung dar. Im ersten Kapitel wurde die Motivation für die Wahl des Themas und die zu erfüllenden Aufgaben beschrieben. Eine Darstellung der wichtigsten Grundlagen, die notwendig sind, um zu verstehen, wie TwitterPooler funktioniert erfolgte im zweiten Kapitel. In den Kapiteln 3 und 4 wurden das Konzept und die Implementierung des Systems vorgestellt.

Das erstellte System erfüllte alle im ersten Teil der Arbeit gestellten Anforderungen. Das Produkt bietet eine grafische Oberfläche und eine Reihe von Funktionen zur Erleichterung von Prozessmanagement (siehe Kapitel 3.1), Tweet-Retrieval, Pooling und NLP-Vorverarbeitung. Die Ergebnisse der Untersuchungen in Kapitel 5 trugen auch zur Beantwortung der Frage bei, wie sich die neuen Funktionalitäten auf die Leistungsfähigkeit auswirken. Die Analyse hat gezeigt, dass die Vorbereitung der Dokumente und der NLP-Prozesse selbst wesentliche Faktoren sind, die die Arbeit verlangsamen. Positiv ist jedoch, dass durch den Einsatz von parallelen Funktionen die Dauer von NLP-Analysen verkürzt werden kann. Die Konfrontation mit dem alternativen Programm twarc zeigte die Schwäche des Systems, nämlich die Wartezeit für das Herunterladen von Tweets. Trotz des erwähnten Mappings, der Erstellung eines JCas-Objekts und der Speicherung in der Datenbank sind die Zeitunterschiede zu groß. Daher sollte in Zukunft daran gearbeitet werden, um diese Punkte zu optimieren.

Aufgrund des Mangels an Applikationen, die Pooling und NLP-Vorverarbeitung anbieten, bezog sich die im Rahmen dieser Arbeit durchgeführte vergleichende Studie nur auf die Daten-Download-Prozesse. Es sei auch darauf hingewiesen, dass die Auswahl einer alternativen Anwendung für die oben genannten Zwecke ebenfalls eingeschränkt war. Denn das erstellte Programm ist nur mit der neuesten Version der Twitter API v2 kompatibel.

TwitterPooler ist ein System, das Frische und Innovation in die Twitter-Community bringt. Trotz der Mängel dürfte seine Attraktivität immer noch als hoch einzuschätzen sein. Deswegen sollte in Zukunft weiterentwickelt werden.

6.2. Weiterführende Arbeiten

In diesem Abschnitt werden Ideen für die weitere Entwicklung der Applikation vorgestellt:

- Der wichtigste Punkt ist die Verbesserung der Funktionsweise und insbesondere die Verkürzung der Dauer des Tweet-Abrufs. Dafür könnte die Serialisierung und Deserialisierung des JCas-Objekts optimiert werden. Es ist auch überlegenswert, ob die

Erstellung dieser Objekte während des Tweet-Retrievals oder nur während der NLP-Vorverarbeitung erfolgen soll.

- Eine interessante Ergänzung wäre die Implementierung von rekursivem Pooling, was die Möglichkeit bieten würde, Pools aus bereits bestehenden Pools zu erstellen.
- Während der Entstehung dieser Arbeit erschien ein interessanter Artikel, in dem der VBEST-Algorithmus vorgestellt wurde. Das Verfahren dient der Entnahme von Stichproben aus Twitter API. VBEST liefert eine Liste, anhand derer sich feststellen lässt, zu welchen Tageszeiten es die meisten Tweets gibt, die die gesuchten Schlüssel enthalten (vgl. Buskirk u. a. 2022, S. 4–5). Eine Erweiterung von TwitterPooler mit diesem Ansatz könnte das Projekt noch attraktiver machen.
- Die Twitter API v2 bietet eine große Anzahl von Endpunkten, die erfolgreich in eine entwickelte Anwendung implementiert werden können.
- Die Möglichkeit, die Daten zu visualisieren, wäre ebenfalls eine interessante Funktion. So könnte sich der Nutzer ein besseres Bild über den Inhalt der heruntergeladenen Tweets machen.

Literatur

Ahmed, Wasim, Peter A Bath und Gianluca Demartini (2017). „Using Twitter as a data source: An overview of ethical, legal, and methodological challenges“. In: *The ethics of online research*.

AlAgha, Iyad (2021). „Topic Modeling and Sentiment Analysis of Twitter Discussions on COVID-19 from Spatial and Temporal Perspectives“. In: *Journal of Information Science Theory and Practice* 9.1, S. 35–53.

Borra, Erik und Bernhard Rieder (2014). „Programmed method: Developing a toolset for capturing and analyzing tweets“. In: *Aslib journal of information management*.

Boyd, Danah, Scott Golder und Gilad Lotan (2010). „Tweet, tweet, retweet: Conversational aspects of retweeting on twitter“. In: *2010 43rd Hawaii international conference on system sciences*. IEEE, S. 1–10.

Brett, Molina (26. Okt. 2017). *Twitter overcounted active users since 2014, shares surge on profit hopes*. URL: https://eu.usatoday.com/story/tech/news/2017/10/26/twitter-overcounted-active-users-since-2014-shares-surge/801968001/ (besucht am 09. 02. 2022).

Bruns, Axel, Dirk Lewandowski und Katrin Weller (2014). *Twitter data analytics*. Emerald Group Publ.

Bueno, Alex und Jason Porter (2020). *Quarkus Cookbook*. O'Reilly Media, Inc.

Buskirk, Trent D u. a. (2022). „Sweet tweets! Evaluating a new approach for probability-based sampling of Twitter“. In: *EPJ Data Science* 11.1, S. 9.

Chopra, Rohan u. a. (2020). „The Natural Language Processing Workshop“. In: *Packt: Birmingham*.

Dayter, Daria (2015). „Small stories and extended narratives on Twitter“. In: *Discourse, Context & Media* 10, S. 19–26.

Freemarker (2022). URL: https://freemarker.sourceforge.io/index.html (besucht am 11. 02. 2022).

Hemati, Wahed, Tolga Uslu und Alexander Mehler (2016). „TextImager: a Distributed UIMA-based System for NLP“. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, S. 59–63.

Hong, Liangjie und Brian D Davison (2010). „Empirical study of topic modeling in twitter“. In: *Proceedings of the first workshop on social media analytics*, S. 80–88.

Khan, Muhammad Haseeb UR Rehman, Kei Wakabayashi und Satoshi Fukuyama (2019). „Events Insights Extraction from Twitter Using LDA and Day-Hashtag Pooling“. In: *Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services*, S. 240–244.

Mehrotra, Rishabh, Scott Sanner, Wray Buntine und Lexing Xie (2013). „Improving lda topic models for microblogs via tweet pooling and automatic labeling“. In: *Proceedings of the 36th*

international ACM SIGIR conference on Research and development in information retrieval, S. 889–892.

Pournaki, Armin, Felix Gaisbauer, Sven Banisch und Eckehard Olbrich (2020). „The twitter explorer: a framework for observing twitter through interactive networks“. In: *arXiv preprint arXiv:2003.03599*.

Schildt, Herbert (2021). *Java: The Complete Reference, Twelfth Edition, 12th Edition*. McGraw-Hill.

Späth, Peter (2021). *Beginning Java MVC 1.0: Model View Controller Development to Build Web, Cloud, and Microservices Applications*. Springer.

Trelle, Tobias (2014). *MongoDB: der praktische Einstieg*. dpunkt. verlag.

Tweepy Documentation (2022). URL: `https://docs.tweepy.org/en/stable/` (besucht am 10. 02. 2022).

Twitter API (2022). URL: `https://developer.twitter.com/en/docs/twitter-api` (besucht am 09. 02. 2022).

Twitter API v2 support (2022). URL: `https://developer.twitter.com/en/support/twitter-api/v2` (besucht am 09. 02. 2022).

Varanasi, Balaji (2019). *Introducing Maven: A Build Tool for Today's Java Developers*. Apress.

Vayansky, Ike und Sathish AP Kumar (2020). „A review of topic modeling methods“. In: *Information Systems* 94, S. 101582.

Verspoor, Karin und William A. Baumgartner (2013). „Unstructured Information Management Architecture (UIMA)“. In: *Encyclopedia of Systems Biology*. Hrsg. von Werner Dubitzky, Olaf Wolkenhauer, Kwang-Hyun Cho und Hiroki Yokota. New York, NY: Springer New York, S. 2320–2324. ISBN: 978-1-4419-9863-7. DOI: `10.1007/978-1-4419-9863-7_183`. URL: `https://doi.org/10.1007/978-1-4419-9863-7_183`.

Weng, Jianshu, Ee-Peng Lim, Jing Jiang und Qi He (2010). „Twitterrank: finding topic-sensitive influential twitterers“. In: *Proceedings of the third ACM international conference on Web search and data mining*, S. 261–270.

Wilton, Paul (2003). *Beginning JavaScript*. Indianapolis, Ind. : Wiley. ISBN: 9780764558559.

A. Softwaredokumentation

A.1. Deployment

Um die TwitterPooler-Anwendung erfolgreich zu installieren, ist wie folgt vorzugehen:

- Installation von MongoDB auf dem ausgewählten System[1].
- Erstellung von drei Datenbanken: Administration, Pools, Tweets.
- Die Administrations-Datenbank muss drei Kollektionen beinhalten: Accounts, Pools, Query.
- Einfügen von Entwickler-Kontodaten in die Kollektion Accounts. A.1 zeigt ein Beispielkonto im json-Format. Die notwendigen Daten sind im Twitter-Account Dashboard zu finden.

Listing A.1: Beispiel Account

```
{
  "_id": {
    "$oid": "618723deab131de401c15331"
  },
  "bearerToken": "DAS_IST_EIN␣BEISPIEL",
  "name": "Grzegorz",
  "reset": {
    "$date": "2022-03-28T23:59:01Z"
  },
  "twitterCap": 180,
  "twitterLimit": 2000000,
  "type": "DEV"
}
```

- Klonen des Projektes aus Gitlab [2]
- Nach dem Öffnen des Projekts sollten die Datenbankadressen auf Korrektheit überprüft werden. Die Eingaben sind zu finden unter application.properties (src/main/resources/application.properties)
- Es gibt zwei Möglichkeiten, .jar-Dateien zu erstellen und die Anwendung deployen. Durch Ausführung des Befehls ./mvnw package -Dquarkus.package.type=uber-jar oder

[1] https://www.mongodb.com/try/download/community

[2] http://gitlab.texttechnologylab.org/Grz/twittermalstrom.git

./mvnw package -Pnative -Dquarkus.native.container-build=true. Der zweite Befehl ist für die Generierung der nativen Version zuständig[3]. Die ausführbare Datei (in beiden Fällen) ist target/*-runner.jar.

- Der Inhalt der Datei types.txt, die sich in der erzeugten .jar-Datei (target/*-runner.jar/META-INF/org.apache.uima/types.txt) befindet, muss durch den in A.2 dargestellten Text ersetzt werden. Dies kann manuell (direkte Änderung in der .jar-Datei) oder mit dem Befehl: **jar uf _absolut_path__-runner.jar META-INF/org.apache.uima.fit/types.txt** geschehen. Der Befehl sollte von folgendem Ort aus ausgeführt werden: **twitter-pooler /src/main/resources/META-INF/resources/deploy**.

Listing A.2: types.txt

```
classpath*:desc/type/*.xml
classpath*:org/apache/uima/ruta/engine/BasicTypeSystem.xml
classpath*:org/apache/uima/ruta/engine/InternalTypeSystem.xml
classpath*:org/apache/uima/ruta/engine/HtmlTypeSystem.xml
classpath*:org/apache/uima/ruta/engine/PlainTextTypeSystem.xml
```

A.2. Bekannte Probleme

Bei der Serialisierung/Deserialisierung eines JCas-Objekts erscheint die folgende Warnmeldung:

Listing A.3: Warnung

```
WARN [org.apa.uim.res.met.TypeSystemDescription] (pool-8-thread-2)
[jar:file:...target/twitter-pooler-1.0.0-SNAPSHOT-runner.jar!
/desc/type/HeidelTime_TypeSystemStyleMap.xml] is not a type file.
```

A.3. Anmerkungen

Während der Entwicklung von TwitterPooler wurde ein schwerwiegender Fehler im Zusammenhang mit der Log4J-Bibliothek entdeckt. Infolgedessen wurde die Textimager-API deaktiviert. Da es nicht möglich war, die Tests im Zusammenhang mit der NLP-Vorverarbeitung durchzuführen, wurde eine alternative Lösung implementiert. Um den ursprünglichen Zustand der Anwendung wiederherzustellen, sollten die folgenden Änderungen an der Klasse TextimagerRunnable.java vorgenommen werden:

- Kommentiere die Zeilen 58 bis 62 und 82 ein.
- Kommentiere Zeile 81 aus.

Die Anzahl der während des NLP-Prozesses verwendeten Threads wird in der Klasse TextimagerRunnable.java initialisiert.

[3] https://quarkus.io/guides/maven-tooling.html

Listing A.4: TextimagerRunnable.java

```
ExecutorService manager = Executors.newFixedThreadPool(4);
```