Dokumentation der Softwarearchitektur des Projekts Twitter-Dash

Hahn, Bastian Kleber, Martin Klier, Andreas Kreussel, Lukas Paris, Felix Ziegler, Andreas
b.hahn@oth-aw.de m.kleber2@oth-aw.de a.klier@oth-aw.de l.kreussel@oth-aw.de f.paris1@oth-aw.de a.ziegler1@oth-aw.de

***Zusammenfassung*—In diesem technischen Report wird die Softwarearchitektur des Projekts Twitter-Dash vorgestellt. Das Projekt wurde im Rahmen der Vorlesung Big Data and Cloud Computing (BDCC) implementiert. Ziel der Implementierung ist es, ein Dashboard zu erstellen, welches aktuelle und historische Informationen zu Twitter Trends anzeigen und analysieren kann.**

***Index Terms*—Twitter, Big Data, Database, Web UI, Data Analysis**

I. Introduction

Bei der Applikation Twitter-Dash werden von der Twitter-API alle 15 Minuten die aktuellen Trends abgegriffen und in einer lokalen Datenbank gespeichert. Die Kommunikation zwischen den einzelnen Services ist über eine gRPC-Schnittstelle realisiert. Die Interaktion des Anwenders mit der Applikation erfolgt dabei über eine Web-UI basierend auf dem React Web-Framework.

II. Architektur Allgemein

Als Architektur wird ein Ansatz verfolgt, der sich stark an Microservices orientiert. Dadurch wird eine abgeschlossene Logik von Front- und Backendkomponenten erreicht, die jeweils von einem Subteam entwickelt wird. Diese gekapselten Einheiten können jeweils als isolierte Applikation der Containervirtualisierung Docker betrieben werden. Durch eine zentrale Konfigurationsdatei des Containers können diese zwischen den Teammitgliedern ausgetauscht und ausgeführt werden. Damit wird eine hohe Portabilität gewährleistet. Die Kommunikation der Teilsysteme wird durch eine sprachunabhängige gRPC-Schnittstelle angeboten. Dadurch wird es möglich einzelne Komponenten des Gesamtsystems auszutauschen, ohne größere Veränderungen an den anderen Microservices vornehmen zu müssen [1].

III. Kommunikation

Um Konflikten zwischen den Schnittstellendefinitionen der einzelnen Dienste zu vermeiden, wurden alle gRPC-Schnittstellen in einem globalen Ordner definiert und vor dem Ausführen der Container über ein Script in die jeweiligen Source-Verzeichnisse der Dienste kopiert, bzw. wenn nötig mit dem gRPC-Tool kompiliert. Dadurch werden Änderungen an den Schnittstellen gleichzeitig für alle Dienste vorgenommen.

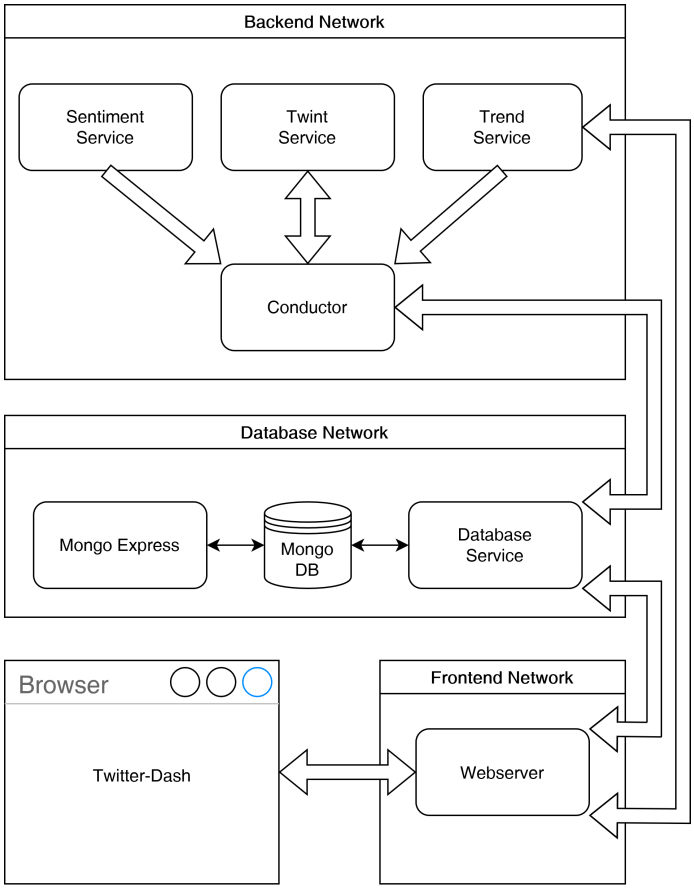

Abbildung 1. Übersicht Gesamtsystem

IV. Datenakquise

Um Daten für das Dashboard zu erhalten, werden zwei unterschiedliche Vorgehensweisen verwendet. Der meist verwendete Ansatz ist die Abfrage über die offizielle Twitter-API [2]. Über diese werden die aktuellen Hashtags periodisch abgerufen. Außerdem wurde Twint, ein fortschrittliches Scraping-Tool verwendet. Dieses Tool liefert vollständige Tweets, inklusive zugehöriger Meta-Daten, ohne direkten Zugriff auf die Twitter-API zu benötigen [3].

Trend-Service

Für den Zugriff auf die Twitter-API wurde die Python-Bibliothek Tweepy [4] verwendet. Diese Bibliothek bietet einen Client für die Verbindung zur Twitter-API. Über diesen Client werden alle 15 Minuten die aktuellen Trends pro Land abgefragt und nach Hashtags gefiltert. Diese werden über den Conductor zur Speicherung an die Datenbank gesendet.

Der Trend-Service bietet folgende Funktionen:

- OnNewTrends:
  Löst alle 15 Minuten ein Event aus, welches die aktuellen Hashtags in einer Liste enthält. Die Listenelemente umfassen:
  - Hashtag
  - Platzierung nach Aktualität im Land
  - Anzahl der Tweets zu diesem Hashtag
  - Zeitstempel der Abfrage
  - Land
- GetRecentTweetCounts:
  Liefert die Anzahl der Tweets in einem vorgegebenen Zeitraum zu einem Hashtag zurück.

Twint-Service

Da die Nutzung der Twitter-API eingeschränkt ist, wurde Twint verwendet, um viele Tweets zu laden. Der Twint-Service wird nur auf Anfrage des Conductors ausgeführt. Dieser gibt einen Hashtag, eine Start- und Endzeitpunktsangabe und eine Sprache vor. Zu dieser Anfrage werden Tweets mit Twint gesammelt und nach Bereinigung von unwichtigen Metadaten zurück an den Conductor gegeben.

Die zurückgegebenen Elemente enthalten die folgenden Informationen:

- Tweet-ID
- Conversation-ID
- Zeitstempel
- Textinhalt des Tweets
- Anzahl der Likes
- Anzahl der Retweets
- Hashtags in dem Tweet
- Sprache

V. Datenverarbeitung

Aus den gesammelten Daten können weitergehende Informationen extrahiert werden. Hier wurde ein Service implementiert, der das Sentiment eines Tweets analysiert.

Sentiment-Analyse

Bei der Sentimentanalyse werden die in einem Text ausgedrückten Meinungen mithilfe von NLP analysiert, um festzustellen, wie die Einstellung des Verfassers zu einem bestimmten Thema ist. Zur Analyse der Stimmungen wurden zwei verschiedene Modelle eingebunden, ein bereits vortrainiertes neuronales Sprachmodell (BERT [5]) und ein stochastisches Modell (TextBlob [6]). Aufgrund des hohen Ressourcenverbrauchs des Transformers (BERT) wurde das TextBlob-Modell für die Ausführung gewählt.

Die Ausführung der Sentiment-Analyse erfolgt nach einem Aufruf durch den Conductor. Bei dieser Anfrage wird ein Text übergeben, der analysiert werden soll, sowie eine Sprache, die zur Wahl des richtigen TextBlob-Modells benötigt wird. Bevor der Text analysiert werden kann, muss er bereinigt werden. Dabei werden zum Beispiel unerwünschte Zeichen entfernt und die Zeichenkette in Kleinbuchstaben umgewandelt. Als Ausgabe liefert der Service einen Zahlenwert zwischen -1 und 1. Dabei wird der Wert -1 für negative und 1 für positive Stimmungen angenommen.

VI. Workflow

Um die Zusammenarbeit zwischen den verschiedenen Datenakquise Diensten zu ermöglichen wurde ein Conductor-Service erstellt. Der Dienst wurde in .NET 6 mithilfe der ELSA [7] Open Source Workflow Library entwickelt. Der Conductor Dienst enthält Client Implementierungen für alle benötigen Dienste. Für die Datenakquise wurde ein Standard-Workflow implementiert, welcher die benötigen Daten von den anderen Diensten abruft und für weitere Verarbeitungsschritte aufbereitet. Dieser Workflow wird durch das Erhalten von Trends des Trend-Services initialisiert und gestartet. Über die ELSA-Bibliothek wird ein Web-Interface bereitgestellt, welches den aktuellen Status eines laufenden Workflows visualisiert, sowie Informationen und Ergebnisse von bereits ausgeführten Durchläufen enthält.

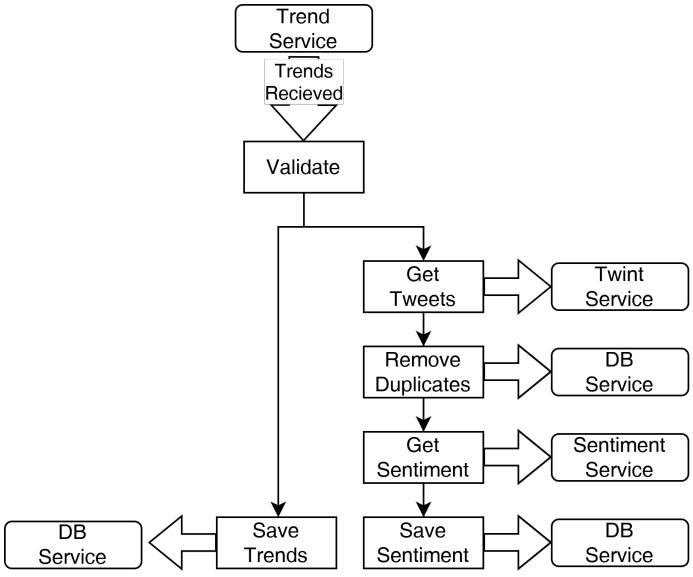

Abbildung 2. Standard Workflow

VII. Datenbank

Alle Datenbankanfragen werden durch einen Datenbank-Service abstrahiert, der die Funktionen als gRPC-Schnittstelle anbietet. Dabei laufen die Datenbank und der Datenbank-Service in jeweils einem eigenen Container. Durch die gRPC-Schnittstelle können die Daten typsicher und plattform- sowie programmiersprachenunabhängig an den Datenbank-Service

übergeben und von diesem zurückgeliefert werden.
Vom Datenbank-Service werden folgende Funktionen angeboten:
Zur Speicherung in die Datenbank:

Trends speichern (StoreTrends)

- Beschreibung: Speichert Trends in die Datenbank
- Parameter:
  - Zeitstempel
  - Liste von Trends mit folgenden Attributen:
    - trendtype
    - name
    - country
    - placement
    - tweetVolume24
- Rückgabe: Keine

Sentiment speichern (StoreSentiment)

- Beschreibung: Speichert Sentiment in die Datenbank
- Parameter:
  - Liste von Sentiments mit folgenden Attributen:
    - Tweet
    - Sentiment
    - Topic
- Rückgabe: Keine

Zum Laden aus der Datenbank:

Aktuelle Trends abrufen (GetCurrentTrends)

- Beschreibung: Lädt aktuelle Trends aus der Datenbank
- Parameter:
  - Land
  - Limit
- Rückgabe:
  - Zeitstempel
  - Liste von Trends mit folgenden Attributen:
    - Trendtyp
    - Name
    - Land
    - Platzierung
    - 24Stunden-Volumen

Vergangene Trends abrufen (GetRecentTrends)

- Beschreibung: Lädt vergangene Trends aus der Datenbank
- Parameter:
  - Hashtag
  - Land
  - Startzeitpunkt (optional)
  - Endzeitpunkt (optional)
- Rückgabe:
  - Liste an Trends mit folgenden Attributen:
    - Zeitstempel
    - Liste von Trends mit folgenden Attributen:
      - Trendtyp
      - Name
      - Land
      - Platzierung
      - 24Stunden-Volumen

Vorhandene Länder abrufen (GetAvailableCountries)

- Beschreibung: Lädt vorhandene Länder aus der Datenbank
- Parameter: Keine
- Rückgabe: Liste mit allen Ländern, die in der Datenbank vorhanden sind

Trends mit vorhandenen Sentiment abrufen (GetTrendsWithAvailableSentiment)

- Beschreibung: Lädt Trends mit vorhandenen Sentiment aus der Datenbank, welche den Query String enthalten.
- Parameter:
  - Query
  - Limit
- Rückgabe: Liste von Trendnamen

Aktuelles Sentiment abrufen (GetCurrentSentiment)

- Beschreibung: Lädt aktuelles Sentiment (Mittel der letzten Stunde) aus der Datenbank
- Parameter:
  - Trendname
- Rückgabe: Sentiment

Vergangene Sentiments abrufen (GetRecentSentiments)

- Beschreibung: Lädt vergangene Sentiments aus der Datenbank
- Parameter:
  - Trendname
  - Startzeitpunkt (optional)
  - Endzeitpunkt (optional)
  - Auflösung
- Rückgabe: Liste der vergangenen Sentiments in Zeitabschnitt

DB auf Tweet prüfen (GetUniqueTweets)

- Beschreibung: Gibt zurück, welche Tweets noch nicht in der Datenbank sind
- Parameter: Liste an Tweet IDs
- Rückgabe: Liste an Tweet IDs

Vom Backend erzeugte Daten werden in einer MongoDB Datenbank gespeichert. Dabei werden Trends zu bestimmten Zeitpunkten mit folgenden Feldern gespeichert:

- Trendtyp
- Name
- Land
- Platzierung
- 24Stunden-Volumen

Für alle erfassten Tweets werden folgende Felder gespeichert:

- Tweet-ID
- Sentiment
- Topic
- Zeitstempel

VIII. Frontend

IX. Testing

Um die Funktionalität des Backends zu garantieren wurden alle Backend-Dienste mit einem zur verwendeten Programmiersprache passenden Unit-Test Framework getestet. Die Code-Converage liegt dabei bei jeden Dienst über 90%. Der Trend-,Twint- und Sentiment-Dienst wurden mit dem Python Unittest [8] Framework getestet.
Der Conductor- und Datenbank-Dienst wurden mit dem NUnit [9] Framework in einem Arrange-Act-Assert-Pattern getestet. Das Frontend wurde noch nicht mit Unittests abgedeckt.

Literatur

[1] Microservice-Frontend-Architekturen [Online] https://www.sigs-datacom.de/uploads/tx_dmjournals/attermeyer_OTS_Microservices_Docker_16.pdf (visited on Jun. 21, 2022)

[2] Twitter-API [Online] https://developer.twitter.com/en/docs/twitter-api (visited on Jun. 23, 2022)

[3] TWINT - Twitter Intelligence Tool [Online] https://github.com/twintproject/twint (visited on Jun. 23, 2022)

[4] Tweepy: Twitter for Python! (2020) [Online] https://github.com/tweepy/tweepy (visited on Jun. 23, 2022)

[5] bert-base-multilingual-uncased-sentiment [Online] https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment (visited on Jun. 23, 2022)

[6] TextBlob: Simplified Text Processing [Online] https://textblob.readthedocs.io/en/dev/ (visited on Jun. 23, 2022)

[7] ELSA - An open source .NET Standard workflow library [Online] https://elsa-workflows.github.io/elsa-core/ (visited on Jun. 25, 2022)

[8] Unittest - Unit testing framework [Online] https://docs.python.org/3/library/unittest.html (visited on Jun. 25, 2022)

[9] NUnit is a unit-testing framework for all .Net languages. [Online] https://nunit.org/ (visited on Jun. 25, 2022)