Realtime Data Mining mit Apache Kafka und Samza

Wackere Goldgräber

Peter Hoffmann, Paul Watzlaw

Apache Samza ist ein modernes Stream-Processing-Framework, mit dem sich komplexe Daten-Pipelines aufbauen lassen. Data-Mining-Tasks innerhalb der Verarbeitungskette fördern nebenbei manche Entdeckung zutage.

Mit dem Aufkommen verteilter Plattformen und dem damit verbundenen Bedarf an Echtzeitverarbeitung großer Datenmengen rollte in den letzten Jahren eine neue Welle von Anwendungen heran. Stream-Processing-Applikationen ermöglichen als Gegenentwurf zum traditionellen Batch-Processing das Verarbeiten von Daten, sobald diese entstehen, also nahezu in Echtzeit.

Eines der bekanntesten Stream-Processing-Systeme ist Kafka. Es wurde ursprünglich von LinkedIn entwickelt und ist seit 2011 als Apache-Projekt öffentlich verfügbar. LinkedIn nutzt Kafka intensiv zum Tracking der Benutzeraktivitäten, für den Nachrichtenaustausch und für das Sammeln von Kennzahlen. Im Oktober 2019 verzeichnete LinkedIn ein Aufkommen von sieben Billionen Kafka-Nachrichten täglich.

Kafka speichert Nachrichten (Messages) in sogenannten Topics und liest sie dort wieder aus. Anwendungen, die Nachrichten bereitstellen, nennt man Producer. Bei Anwendungen, die Nachrichten aus einem Topic lesen, spricht man von Consumern. Der Producer-Consumer-Ansatz erinnert stark an klassische Message-Broker wie RabbitMQ. Im Unterschied zu einem Message-Broker speichert Kafka jedoch alle Nachrichten im eigenen Transaktions-Log, bei Bedarf auch zeitlich unbegrenzt. Auf diese Weise lassen sich revisionssichere Applikationen erstellen und auch nachträglich neue Consumer-Anwendungen hinzufügen, die das gesamte Transaktions-Log verarbeiten, um eine neue Sicht auf die Daten zu erstellen.

iX-TRACT

- Data Mining spürt Zusammenhänge, Trends und Strukturen in großen Datenmengen auf.
- Mit Apache Samza lassen sich autonome Tasks implementieren, die Daten in Echtzeit lesen, schreiben und analysieren können.
- Samza versteht sich mit Apache Kafka, Microsoft Azure Event Hubs, Amazon Kinesis Streams und Hadoop als Input-Quellen.
- Die Assoziationsanalyse ist ein Verfahren zum Auffinden häufig vorkommender Muster, Korrelationen und kausaler Strukturen in Datenbeständen.

Samza für Stream-Processing-Tasks

Ebenso wie Kafka stammt Apache Samza ursprünglich von LinkedIn und dient speziell der Entwicklung von Stream-Processing-Anwendungen. Samza ermöglicht es, kleine Tasks zu implementieren, die über Input- und Output-Deskriptoren in Echtzeit auf einer Stream-Processing-Topologie arbeiten. Samza ist nicht von Kafka abhängig, kann aber Kafka als Input- und Output-Quelle nutzen. LinkedIn hat über 3000 Samza-Tasks im Einsatz, deren Aufgaben von Störungs- und Betrugserkennung bis zur Echtzeitanalyse reichen. Samza-Tasks lassen sich sowohl mit dem von Hadoop bekannten Resource Manager YARN (Yet Another Resource Nego-

Data Mining – wenn das Unbekannte lockt

Data Mining hat die Aufgabe, Wissen zutage zu fördern. Es ist ein Prozess zum Auffinden unbekannter und im besten Fall nicht trivialer Strukturen, Zusammenhänge und Trends in den Daten und gehört damit zu den entdeckenden Analyseverfahren. Wer meint, genug zu wissen, braucht kein Data Mining. Wer weiß, worauf er in seinem Wissensgebiet achten muss, arbeitet mit zielgerichteten Querys oder geschickt eingerichteten Reporting-Tools.

Gib mir Regeln

Eine der wichtigsten Untergruppen der vielfältigen Data-Mining-Methoden stellen die Verfahren zur Regelgenerierung dar. Sie dienen dazu, auf Strukturen und Zusammenhänge und die möglichen dahinterliegenden Kausalitäten in den Daten hinzuweisen. Bei der Regelgewinnung entstehen explizite Wenn-dann-Regeln, Ergebnisse also, die gut kommunizierbar sind und zum weiteren Nachforschen und Aufdecken verdeckter Kausalitäten im entsprechenden Wissensbereich einladen. Typische Regelverfahren sind bayessche Netze, Entscheidungsbäume und Fuzzy-Regeln.

Das Verfahren, das in diesem Artikel für die Analyse von Kurslisten zur Anwendung kommt, nennt sich Association-Rule Mining. Gerne wird auch der Begriff des Frequent Item-Set Mining verwendet. Noch allgemeiner gefasst spricht man schlicht vom Auffinden von Koinzidenzen.

Bekanntheit in der Allgemeinheit erlangte das Verfahren unter dem Begriff der Warenkorbanalyse. Fast jeder dürfte dieses Beispiel kennen: „Wer freitagabends Windeln kauft, der kauft auch Bier!“ Bei der Warenkorbanalyse gilt es herauszufinden, welche Produkte mit welcher Wahrscheinlichkeit wie oft gemeinsam eingekauft werden. Will man dieses Konzept allgemeiner fassen, spricht man nicht von Warenkörben und Produkten, sondern von Transaktionen, Item-Sets und Events.

WKN	Änderung [%]
A1EWWW	2.16
A0D9PT	1.15
843002	1.08
840400	1.02
A2DSYC	0.98
747206	0.97
604843	0.87
...	...
514000	0.02
623100	0.00
555750	-0.03
555750	-1.20
ENAG99	-1.54

Event
A1EWWW-TOP5
A0D9PT-TOP5
843002-TOP5
840400-TOP5
A2DSYC-TOP5
514000-FLOP5
623100-FLOP5
555750-FLOP5
555750-FLOP5
ENAG99-FLOP5

Zahlenwerte (Aktienkurse) in Events überführen (Abb. 1)

WKN	WKN	Coverage	Confidence	Lift
710000	543900	7.25 %	80%	12.75

Beispiel für eine Assoziationsregel (Abb. 2)

Interessant wird das Assoziation-Rule Mining in den weiteren Analyseschritten, wenn es um die Interpretation der so gefundenen Regeln geht. Koinzidenzen als Zusammentreffen zweier oder mehrerer Ereignisse stellen ein starkes Indiz für das Bestehen möglicher offener oder verborgener kausaler (ursächlicher) Zusammenhänge dar. Man darf allerdings auch nicht verschweigen, dass ein Ergebnis eines solchen Prozesses auch sein kann, dass es keinen Sinn macht, hier nach bestimmten Zusammenhängen zu fahnden, oder dass sich schlicht nichts finden lässt.

Bei der Generierung von Assoziationsregeln geht es nicht um die Suche nach linearen Zusammenhängen, sondern um das Auffinden von Verbundwahrscheinlichkeiten von Ereignissen. Deshalb muss man im ersten Schritt des Prozesses die Kurse, die ja als reine Zahlenwerte vorliegen, in Events überführen. Anschließend werden die Häufigkeiten ihres gemeinsamen Auftretens durchgezählt. Die Bildung von Gütemaßen im letzten Schritt hilft bei der Beurteilung der gefundenen Regeln.

Schritt 1: Diskretisierung – oder: Der Datenanalyst tobt sich aus

Bei der Überführung metrisch skalierter Daten in eine Nominalskala hat ein Datenanalyst große Freiheiten, was man begrüßen, aber natürlich auch bedauern kann. Er muss Vorentscheidungen über Anzahl der Intervalle, Intervallbreiten und Kriterien für Intervallgrenzen fällen.

Für die Fragestellung „Welche Kurse entwickeln sich gemeinsam besonders gut mit oder gerade gegen den Markt und vice versa?“ wird im Beispiel die Kursliste nach Kursänderung absteigend sortiert und aus den oberen und unteren Werten werden Quantile gebildet. Das sind umgangssprachlich die Tops und Flops; sie bilden im Beispiel die nötigen Ausprägungen für den Data-Mining-Prozess. Vorentscheidungen im Rahmen der Datenvorverarbeitung können entscheidenden Einfluss auf die Ergebnisse haben. Um den gewählten Ansatz für die Diskretisierung besser beurteilen zu können, werden die Daten hier in drei Varianten unterschieden in ihrer Intervallbreite parallel durchgerechnet:

- Fall 1: Top 5 versus Flop 5 (obere und untere Sextile)
- Fall 2: Top 10 versus Flop 10 (obere und untere Terzile)
- Fall 3: Kurse besser als Marktdurchschnitt versus Kurse schlechter als der Marktdurchschnitt (Median)

Fall 1 transformiert beispielsweise die Kursliste aus Abbildung 1 auf die folgende Eventliste:

```
[A1EWWW-TOP5, A0D9PT-TOP5, 843002-TOP5,
840400-TOP5, A2DSYC-TOP5, 514000-FLOP5,
623100-FLOP5, 555750-FLOP5, 823212-FLOP5,
ENAG99-FLOP5]
```

Schritt 2: Häufigkeiten auszählen – oder: Wer mit wem und wie oft?

Sind die Transaktionen gebildet, lassen sich die Häufigkeiten auszählen und die Regeln bilden. Zu dem Zweck ermittelt der Algorithmus aussichtsreiche Kandidatenpaare und zählt ihr Vorkommen über alle Transaktionen durch.

Schritt 3: Gütemaße bilden – oder: Welche Regeln sind es wert, genauer begutachtet zu werden?

Im Rahmen dieses Data-Mining-Prozesses entsteht eine Vielzahl von Regeln. Für die Bewertung der Interessantheit der Regeln stehen Gütemaße oder auch Kombinationen davon zur Verfügung.

- Die Abdeckung (Coverage) gibt an, wie häufig die betrachteten Attribute gemeinsam innerhalb des Datenbestandes vorkommen.
- Die Konfidenz (Confidence) gibt die Wahrscheinlichkeit an, wie oft beim Zutreffen von Ereignis A auch Ereignis B tatsächlich eingetroffen ist (bedingte Wahrscheinlichkeit).
- Der Lift ist ein Maß für die Hebelwirkung, das heißt, um wie viel häufiger sich bei Anwendung der linken Regelhälfte die rechte Regelhälfte in den untersuchten Daten einstellt.

Abbildung 2 zeigt exemplarisch eine mögliche Assoziationsregel aus dem Data-Mining-Prozess. In 7,25 Prozent aller Transaktionen befinden sich die Aktien von Daimler und Continental gemeinsam unter den Top-5-Aktien. Die Wahrscheinlichkeit, dass, wenn Daimler unter den Top 5 ist, das Gleiche für Continental gilt, liegt bei 80 Prozent. Die Wahrscheinlichkeit, dass Continental unter den Top 5 ist, ist 12,75 Mal höher, wenn auch Daimler unter den Top 5 ist.

Der Lift ist eine sehr wichtige Größe für die Beurteilung einer Regel, denn er gibt die Hebelwirkung an, um wie viel diese Regel die Wahrscheinlichkeitsdichteverteilung der Untergruppe in Bezug auf die für die Regel geltende Grundgesamtheit verändert. Oft wird der Lift mit der Interessantheit für eine Regel gleichgesetzt; manchmal ist aber auch eine Abwägung aller drei Gütemaße oder Kombinationen davon sinnvoll, zum Beispiel das Produkt aus Abdeckung und Lift.

Listing 1: Das StreamTask-Interface von Apache Samza

```
public interface StreamTask {
          public void process(
                    IncomingMessageEnvelope envelope,
                    MessageCollector collector,
                    TaskCoordinator coordinator) throws Exception;
}
```

Listing 2: Das InitableTask-Interface

```
public interface InitableTask {
  void init(Context context) throws Exception;
}
```

Listing 3: Das WindowableTask-Interface

```
public interface WindowableTask {
  void window(MessageCollector collector,
TaskCoordinator coordinator) throws Exception;
}
```

tiator) als auch als unabhängige Anwendungen betreiben.

Teile und herrsche durch Isolation und Koppelung

Wer in der Unix-Welt zu Hause und auf der Konsole unterwegs ist, dem ist das Prinzip wohlvertraut. Dank genormter Interfaces für Ein- und Ausgabe (stdin/stdout/stderror) lassen sich viele kleine und für sich genommen recht einfache Tools mithilfe des Pipe-Mechanismus mit wenigen Handgriffen zu mächtigen Pipelines verketten. Der nachfolgende Kommandozeilenbefehl zeigt beispielhaft die Analyse einer CSV-Datei durch Verkettung von fünf Unix-Kommandos:

```
awk -F ';' '{print $6}' data.csv | sort | uniq ⤸
                    -c | sort -rn | head -n 3
```

Der Hinweis auf die Unix-Welt schärft den Blick für die Designphilosophie hinter Samza. Eine Komposition von Samza-Stream-Processing-Tasks unter Verwendung von zum Beispiel Kafka als Ablage- und Transportsystem ähnelt in vielerlei Hinsicht den mit Unix-Tools gebildeten Pipelines. Einzelne Samza-Tasks übernehmen jeweils genau eine Teilaufgabe des Gesamtprozesses und lassen sich so bauen, dass ihr Output zum Input für die nächste Task dient. Abgesehen von diesem Input-Output-Mechanismus werden weitere Abhängigkeiten oder eine Kommunikation zwischen den Tasks konsequent vermieden.

Dieses Konzept macht es möglich, die einzelnen Arbeitsschritte voneinander zu isolieren und sie lose zu koppeln. Ein solches Vorgehen bringt viele Vorteile: Tasks lassen sich unabhängig voneinander entwickeln, modifizieren oder austauschen. Auch kann man so einzelne Prozessschritte überwachen, auf Fehler überprüfen oder je nach Bedarf skalieren; zudem lassen sich Zwischenergebnisse leicht persistieren.

Die Kriterien für Isolation und Koppelung sind vielfältig. Sie können technischer Natur sein, sich aber auch aus der Fachlichkeit der Aufgabenstellung ergeben. Die Zwischenergebnisse kann man begutachten oder je nach gewähltem Data-Mining-Verfahren parallel auf verschiedene Arten und Weisen weiterverarbeiten. Weil alle Informationen in der Data-Pipeline gespeichert sind, lassen sich die Ergebnisse der einzelnen Prozessknoten für weitere Nutzungsmöglichkeiten oder zukünftige – zum Zeitpunkt der Implementierung noch nicht spezifizierte – Consumer offen halten.

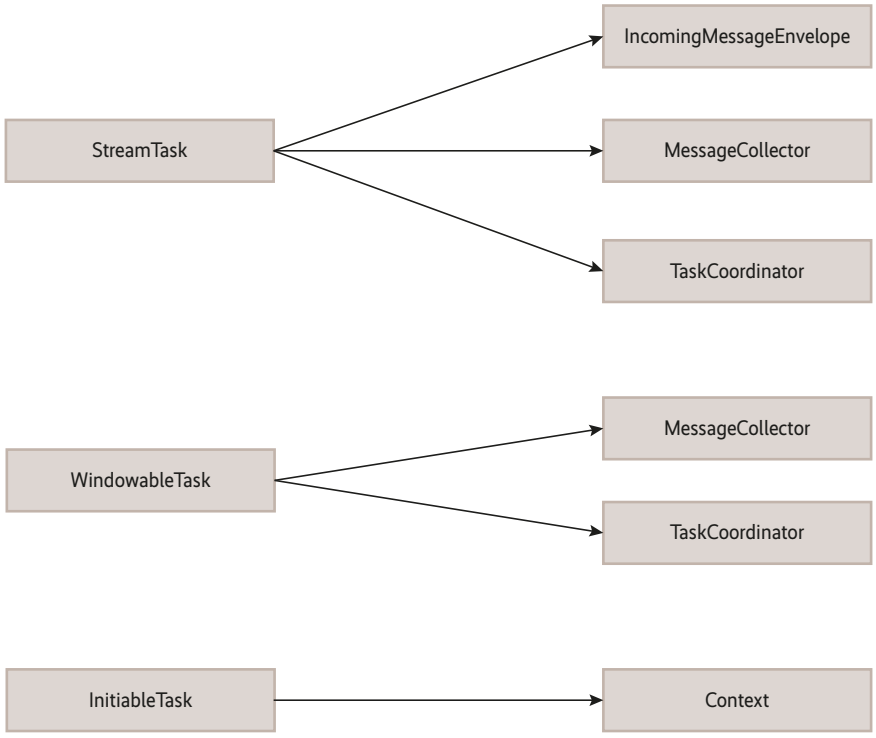
Der in diesem Artikel vorgestellte Anwendungsfall untersucht mithilfe eines Kafka-Samza-Set-ups Kurslisten und prüft, ob es Investments oder Gruppen von Investments gibt, die sich auffallend häufig in ähnlicher oder in entgegengesetzter Weise zum Markt verhalten. Dabei ist es für den nachfolgend dargestellten Data-Mining-Prozess sinnvoll, einzelne Prozessschritte wie zum Beispiel die Datenvorverarbeitung und die eigentliche Datenanalyse voneinander zu trennen.

Ein Dashboard visualisiert die Analyseergebnisse und ermöglicht den Vergleich zwischen den berechneten Korrelationen der historischen Kurse für die letzten 90 und 365 Tage sowie der Echtzeitkurse des aktuellen Tages. Bevor es jedoch um die Details des Anwendungsfalls geht, ist eine kurze Übersicht der Samza-Interfaces sinnvoll.

Eine Samza-Task implementieren

Das `StreamTask`-Interface nimmt den Input aus dem übergeordneten System entgegen und bildet damit die notwendige Basis für einen Samza-Job (siehe Listing 1). Die Methode `process` ist ein Callback und wird bei Zustellung einer neuen Message im übergeordneten Streaming-System ausgelöst. Die Entitäten `IncomingMessageEnvelope` und `MessageCollector` abstrahieren den Input und Output des Jobs. Man weiß an dieser Stelle nicht (und muss es auch nicht wissen), woher die Daten kommen oder wohin sie gehen.

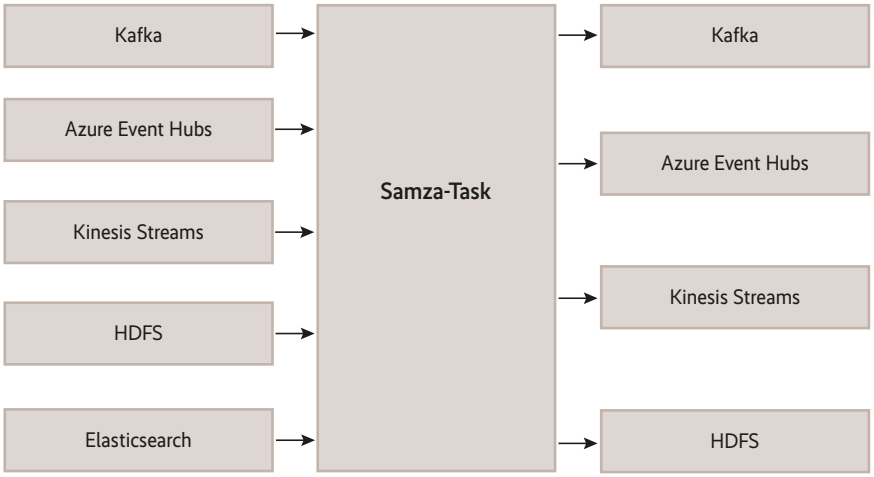
Der Input in Form des `IncomingMessageEnvelope` enthält neben dem eigentlichen Inhalt der Nachricht als Key-Value-Paar alles, um eine Message eindeutig inner-



Die bei Apache Samza am Processing beteiligten Objekte (Abb. 3)

halb des Systems zu identifizieren. Er gibt Informationen zum zugrunde liegenden Streaming-System, zum Namen des Streams, der Partition und dem Offset innerhalb der Partition. Der MessageCollector stellt das Interface zur Verfügung, um den Output zu erzeugen. Über den TaskCoordinator ist es möglich, eine eigene Checkpointing-Strategie zu implementieren. Abbildung 3 zeigt die gängigen am Processing beteiligten Objekte.

Von Haus aus unterstützt Samza als Input-Quellen neben Apache Kafka auch Microsoft Azure Event Hubs, Amazon Kinesis Streams und das Hadoop-Dateisystem. Output kann Samza in Apache Kafka, Microsoft Azure Event Hubs, Hadoop und Elasticsearch schreiben (siehe Abbil-



Input und Output für Samza-Tasks (Abb. 4)

Listing 4: Realtime-Daten als XML-Datensatz

```
<?xml version="1.0" encoding="UTF-8" ?>

<quotes count="30">
    <quote>
        <ag>318</ag>
        <ask>98.11</ask>
        <ask_pieces>650.0</ask_pieces>
        <bid>98.1</bid>
        <bid_pieces>650.0</bid_pieces>
        <change_abs>1.16</change_abs>
        <change_ask_abs>1.12</change_ask_abs>
        <change_ask_rel>1.15</change_ask_rel>
        <change_bid_abs>1.36</change_bid_abs>
        <change_bid_rel>1.41</change_bid_rel>
        <change_rel>1.2</change_rel>
        <close>96.99</close>
        <close_ask>96.99</close_ask>
        <close_bid>96.74</close_bid>
        <close_eur>96.99</close_eur>
        <close_t>2019-09-27 00:00:00</close_t>
        <curfactor_2>1</curfactor_2>
        <curfactor_close_2>1</curfactor_close_2>
        <currency_id>2</currency_id>
        <currency_name>EUR</currency_name>
        <currency_symbol>&amp;euro;</currency_symbol>
        <delay>0</delay>
        <high>98.15</high>
        <high_52w>111</high_52w>
        <isin>DE0007236101</isin>
        <last>98.15</last>
        <last_eur>98.15</last_eur>
        <last_pieces>32.0</last_pieces>
        <last_turnover>-1.0</last_turnover>
        <last_update>2019-09-30 16:01:02</last_update>
        <low>97.0</low>
        <low_52w>84.45</low_52w>
        <market_phase>C</market_phase>
        <name>Siemens</name>
        <open>97.39</open>
        <open_52w>110.22</open_52w>
        <pfx>250</pfx>
        <pfx_notax>250</pfx_notax>
        <pieces>32318</pieces>
        <previous_change_abs>0.99</previous_change_abs>
        <previous_change_rel>1.03</previous_change_rel>
        <previous_close>96.0</previous_close>
        <previous_close_t>2019-09-26 00:00:00</previous_close_t>
        <previous_pieces>48173</previous_pieces>
        <previous_volume>4688736.5</previous_volume>
        <quotesource_id>21</quotesource_id>
        <time>2019-09-30 15:58:12</time>
        <typ_name>Aktie</typ_name>
        <volume>3158091.5</volume>
        <wkn>723610</wkn>
    </quote>
        ...
    <quote>
        <ag>717816</ag>
        <ask>244.3</ask>
        <ask_pieces>150.0</ask_pieces>
        <bid>244.2</bid>
        <bid_pieces>150.0</bid_pieces>
        <change_abs>1.2</change_abs>
        <change_ask_abs>0.9</change_ask_abs>
        <change_ask_rel>0.37</change_ask_rel>
        <change_bid_abs>1.2</change_bid_abs>
        <change_bid_rel>0.49</change_bid_rel>
        <change_rel>0.49</change_rel>
        <close>243.2</close>
        <close_ask>243.4</close_ask>
        <close_bid>243.0</close_bid>
        <close_eur>243.2</close_eur>
        <close_t>2019-09-27 00:00:00</close_t>
        <curfactor_2>1</curfactor_2>
        <curfactor_close_2>1</curfactor_close_2>
        <currency_id>2</currency_id>
        <currency_name>EUR</currency_name>
        <currency_symbol>&amp;euro;</currency_symbol>
        <delay>0</delay>
        <high>245.5</high>
        <high_52w>260</high_52w>
        <isin>DE000A0D9PT0</isin>
        <last>244.4</last>
        <last_eur>244.4</last_eur>
        <last_pieces>20.0</last_pieces>
        <last_turnover>-1.0</last_turnover>
        <last_update>2019-09-30 16:01:02</last_update>
        <low>242.2</low>
        <low_52w>155.6</low_52w>
        <market_phase>C</market_phase>
        <name>MTU Aero Engines</name>
        <open>244.2</open>
        <open_52w>196.9</open_52w>
        <pfx>91</pfx>
        <pfx_notax>91</pfx_notax>
        <pieces>3892</pieces>
        <previous_change_abs>0</previous_change_abs>
        <previous_change_rel>0</previous_change_rel>
        <previous_close>243.2</previous_close>
        <previous_close_t>2019-09-26 00:00:00</previous_close_t>
        <previous_pieces>5068</previous_pieces>
        <previous_volume>1230966.25</previous_volume>
        <quotesource_id>21</quotesource_id>
        <time>2019-09-30 15:58:39</time>
        <typ_name>Aktie</typ_name>
        <volume>947526.31</volume>
        <wkn>A0D9PT</wkn>
    </quote>
</quotes>
```

Listing 5: Diskretisierte Top-Flop-5 als JSON-Objekt

```
{
  "2019-12-09T15:15:41.554": {
    "events":["659990-TOP-5","723610-TOP-5","747206-FLOP-5","578580-TOP-5","514000-TOP-5","519000-FLOP-5","710000-FLOP-5","A2DSYC-TOP-5","543900-FLOP-5","716460-FLOP-5"]
  },
  "2019-12-09T15:14:06.626": {
    "events":["578580-FLOP-5","747206-TOP-5","823212-FLOP-5","A1ML7J-TOP-5","543900-FLOP-5","716460-TOP-5","555750-TOP-5","606214-FLOP-5","843002-TOP-5","703712-FLOP-5"]
  },
  "2019-12-09T14:15:33.122": {
    "events":["823212-FLOP-5","604700-FLOP-5","578580-TOP-5","723610-FLOP-5","555200-TOP-5","A0D9PT-TOP-5","ENAG99-FLOP-5","606214-FLOP-5","840400-TOP-5","703712-TOP-5"]
  },
  "2019-12-09T15:12:32.628": {
    "events":["623100-FLOP-5","520000-TOP-5","514000-FLOP-5","581005-TOP-5","823212-FLOP-5","578580-TOP-5","ENAG99-FLOP-5","A2DSYC-FLOP-5","840400-TOP-5","703712-TOP-5"]
  },
  ...
  "2019-12-09T15:17:46.485": {
    "events":["581005-TOP-5","747206-FLOP-5","A0D9PT-FLOP-5","723610-FLOP-5","ENAG99-TOP-5","A2DSYC-TOP-5","543900-FLOP-5","606214-FLOP-5","840400-TOP-5","703712-TOP-5"]
  }
}
```

dung 4). Input und Output sind in Samza ebenfalls über Interfaces abstrahiert. Somit ist es möglich, individuelle Implementierungen für andere Quell- und Zielsysteme bereitzustellen.

Bei einfachen Jobs, die den Input über eine einzelne Funktion auf den Output transformieren, reicht es in der Regel, das `StreamTask`-Interface zu implementieren. Komplexere Jobs, beispielsweise die Bildung von Aggregationen über bestimmte Zeiträume, zeitgesteuerte Aufgaben oder Anforderungen, die sich aus der Fachlichkeit der Daten ergeben, lassen sich mit Add-on-Interfaces umsetzen.

Das Interface `InitableTask` stellt die `init`-Methode bereit (siehe Listing 2). Diese Methode erhält als Parameter einen Context, der zum Beispiel die Abfrage der Einzelheiten einer Task-Konfiguration oder den Zugriff auf State oder Table Stores ermöglicht.

Listing 6: Die process-Methode speichert eine Nachricht im lokalen State Store

```
@Override
public void process(IncomingMessageEnvelope envelope,
                    MessageCollector collector,
                    TaskCoordinator coordinator) throws Exception {
  String key = (String)envelope.getKey();
  String message = (String)envelope.getMessage();

  try {
    LocalDateTime localDateTime = LocalDateTime.parse(key, formatter);
    Quotes quotes = unmarshal(message);
    stateStore.put(localDateTime, quotes);
  } catch (Exception e) {
    ...
  }}
```

Listing 7: Diskretisierung durch die window-Methode

```
@Override
public void window(MessageCollector collector, TaskCoordinator coordinator) throws Exception {
  HashMap<String, Transaction> transactions = new HashMap<>();
  List<LocalDateTime> localDateTimes = new ArrayList<>();

  KeyValueIterator<LocalDateTime, Quotes> iterator = stateStore.all();
  iterator.forEachRemaining(item -> {
    localDateTimes.add(item.getKey());
  });

  final AtomicReference<Quotes> predecessorQuotes = new AtomicReference<>();

  Collections.sort(localDateTimes);
  localDateTimes.forEach(localDateTime -> {
    Quotes quotes = stateStore.get(localDateTime);
    if (predecessorQuotes.get() != null) {
      Transaction transaction = discretize(quotes, predecessorQuotes.get(), mappingStrategy);
      transactions.put(localDateTime.format(formatter), transaction);
    }
    predecessorQuotes.set(quotes);
  });

  collector.send(new OutgoingMessageEnvelope(OUTPUT_STREAM, new Date().toString(),
transactions));
  coordinator.commit(TaskCoordinator.RequestScope.CURRENT_TASK);
}
```

Flexibel für viele Szenarien

Das Interface `WindowableTask` definiert die Methode `window` (siehe Listing 3). Anders als die Methode `process` aus dem `StreamTask`-Interface wird diese Methode kontinuierlich und unabhängig von der Zustellung einer neuen Message getriggert. Auf diese Weise kann man Daten zu bestimmten Zeitpunkten verarbeiten. Die Frequenz des Callbacks lässt sich über die Task-Konfiguration einstellen.

Durch die Kombination der drei Interfaces `StreamTask`, `InitiableTask` und `WindowableTask` ergeben sich viele Szenarien. In der Regel sammelt man über die Implementierung der `StreamTask`-Schnittstelle die relevanten Messages und speichert diese in einem mit der Task verknüpften State Store. Die Implementierung des `WindowableTask`-Interface übernimmt die notwendigen inhaltlichen Prüfungen, bestimmt den richtigen Zeitpunkt der Ausführung des Jobs und schreibt via MessageCollector den Output.

Mit dem Zugriff auf den TaskCoordinator, der sowohl in der `process`- als auch in der `window`-Methode zur Verfügung steht, ist ein individuelles, anwendungsbezogenes Checkpointing möglich. Die Verwendung eines lokalen State Store in Kombination mit dem Checkpointing erlaubt im Fehlerfall das problemlose Neuaufsetzen des Jobs.

Her mit den Daten

Nach dieser Kurzeinführung in Samza geht es nun ans Eingemachte. Auf der obersten Ebene arbeiten zwei Kafka-Pro-

Listing 8: Auszug aus der Nachricht mit den Assoziationsregeln für die Top-Flop-5 90 Tage

```
[
  {
    "a": "747206-TOP-5",
    "b": "716460-TOP-5",
    "descriptionA": "Wirecard",
    "descriptionB": "SAP",
    "support": 0.1,
    "confidence": 0.25,
    "lift": 1.875
  },
  {
    "a": "514000-TOP-5",
    "b": "723610-FLOP-5",
    "descriptionA": "Deutsche Bank",
    "descriptionB": "Siemens",
    "support": 0.06666666666666667,
    "confidence": 0.16666666666666666,
    "lift":1.0
  },
  {
    "a": "555200-TOP-5",
    "b": "543900-TOP-5",
    "descriptionA": "Deutsche Post DHL",
    "descriptionB": "Continental",
    "support": 0.06666666666666667,
    "confidence": 0.3333333333333333,
    "lift": 2.5
  },
  ...
  {
    "a": "A0D9PT-TOP-5",
    "b": "A1ML7J-FLOP-5",
    "descriptionA": "MTU Aero Engines",
    "descriptionB": "Vonovia",
    "support": 0.06666666666666667,
    "confidence": 0.4,
    "lift": 2.0
  }
]
```

Listing 9: Ein Micronaut-Service als Kafka-Consumer

```
@KafkaListener(offsetReset = OffsetReset.EARLIEST, offsetStrategy = OffsetStrategy.DISABLED)
@Singleton
public class Listener {
  private Map<String, ConsumerRecord<String, String>> cache;

  @Topic(value = {
          "close-strategy-top-flop-5-w-90-d",
          "close-strategy-top-flop-10-w-90-d",
          "close-strategy-median-w-90-d",
          "close-strategy-top-flop-5-w-365-d",
          "close-strategy-top-flop-10-w-365-d",
          "close-strategy-median-w-365-d",
          "last-strategy-top-flop-5-w-1-d",
          "last-strategy-top-flop-10-w-1-d",
          "last-strategy-median-w-1-d"})
  public void receive(ConsumerRecord<String, String> record, Acknowledgement acknowledgement) {
    if (record.value() != null && record.value().trim().length() > 0) {
      cache().put(record.topic(), record);
    }
    acknowledgement.ack();
  }

  ...
}
```

ducer. Ihre Aufgabe ist es, die Aktienkurse per HTTP-API abzurufen und als Raw-Daten in zwei getrennte Topics zu schreiben. Die historischen Daten sind als Kursverläufe je Aktie verfügbar. Sie landen im Topic `stockmarket-raw-data`. Diese Kursverläufe basieren auf Tagesschlusskursen; aus diesem Grund reicht ein tägliches Update dieser Daten.

Die Realtime-Kurse werden als Kurslisten je Index, in diesem Fall für den DAX-Index, zur Verfügung gestellt. Diese Daten fließen in das Topic `stockmarket-realtime-raw-data`. Da es sich hierbei um Echtzeitdaten handelt, findet das Update kontinuierlich in sehr kurzen Zeitintervallen statt. Auf dieser Ebene erfolgt allerdings noch keine echte Verarbeitung. Die Samza-Task schreibt die Daten lediglich in ihrem Originalformat als XML-Datensätze in die beiden Topics (siehe Listing 4).

Das Wunder der Diskretisierung

Auf der zweiten Ebene findet die Datenvorverarbeitung statt. In den Raw-Topics liegen die Kursdaten im XML-Format als metrisch skalierte Werte vor. Das Data-Mining-Verfahren, das hier zum Einsatz kommt, benötigt jedoch nominal skalierte Daten. Dieser Schritt des Prozesses konvertiert deshalb die XML-Daten in Java-Objekte und führt die Aktienkurse in diskrete Werte über. Die Ergebnisse landen anschließend als JSON-Objekte in einem Kafka-Topic (siehe Listing 5).

Um ein Gefühl für die Qualität der vorgenommenen Diskretisierung zu bekommen, werden drei Strategien definiert und im weiteren Verlauf parallel abgearbeitet: die Top/Flop 5, die Top/Flop 10 sowie die Top/Flop 15 (Median). Bei historischen Werten beziehen sich die Tops und Flops auf die Veränderung zu den Schlusskursen des vorherigen Tages. Bei den Realtime-Kursen geht es um die Veränderung zu den Werten des vorherigen Abrufs.

Parallelisierung erreicht man hier durch die Verteilung der Arbeit auf sechs Samza-Tasks. Letztendlich kommt immer dieselbe Implementierung zum Einsatz, allerdings mit jeweils einer anderen Konfiguration, die die entsprechende Diskretisierungsstrategie beschreibt. Die Verarbeitung der Daten ist auf dieser Ebene stateful. Sobald eine neue Nachricht in einem der Raw-Topics verfügbar ist, werden die Tasks aktiv. Innerhalb der `process`-Methode findet zuerst die Konvertierung der XML-Daten in Java-Objekte und die Speicherung in einem lokalen State Store statt (siehe Listing 6). Ein Commit über den TaskCoordinator ermöglicht es, einen aktuellen Checkpoint zu setzen.

Sollte jetzt oder später ein Fehler auftreten, kann sich beim erneuten Hochfahren der Task der State Store reorganisieren und die Task an der richtigen Stelle im Topic aufsetzen. Die eigentliche Diskretisierung der Daten und die Generierung der Transaktionslisten für den nächsten Verarbeitungsschritt erfolgt zeitlich entkoppelt

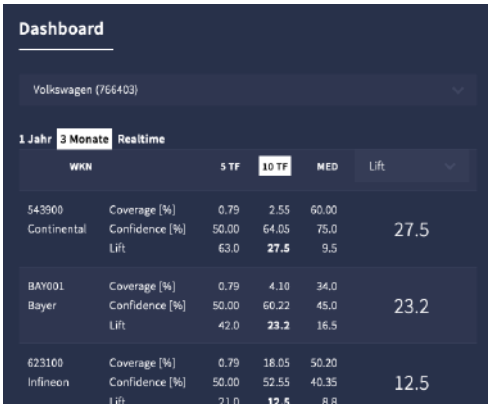
über die `window`-Methode des `WindowableTask`-Interface (siehe Listing 7).

Die Verarbeitung der historischen Kurse findet einmal am Tag statt, die für die Echtzeitdaten des aktuellen Tages in Abständen weniger Sekunden. Die Ergebnisse dieser Verarbeitungsebene landen in sechs Topics: `close-strategy-top-flop-5`, `close-strategy-top-flop-10` und `close-strategy-top-flop-median` für die historischen Werte sowie `last-strategy-top-flop-5`, `last-strategy-top-flop-10` und `last-strategy-top-flop-median` für die Realtime-Werte.

Graf Zahl lässt grüßen

Auf der dritten Ebene findet die eigentliche Datenanalyse statt. Die Assoziationsregeln generiert ein auf diesen Anwendungsfall optimierter Apriori-Algorithmus. Da jede Nachricht alle zur Berechnung nötigen Werte enthält, ist dieser Verarbeitungsschritt stateless. Die historischen Kursdaten werden jeweils in zwei Varianten, die sich nur in der Größe des Betrachtungszeitraums von entweder 90 oder 365 Tagen unterscheiden, durchgerechnet. Bei den Echtzeitkursen entfällt diese Unterscheidung, da es hier nur um die Betrachtung des aktuellen Tages geht.

Wie schon auf zweiter Ebene laufen auch hier mehrere Samza-Tasks, die sich lediglich in der Konfiguration unterscheiden. Die Assoziationsregeln, die sich aus der Analyse der historischen Kurse der letzten 90 oder 365 Tage sowie der Realtime-Kurse ergeben, werden in neun weiteren Topics zur Verfügung gestellt: `close-strategy-top-flop-5-w-90-d`, `close-strategy-top-flop-5-w-365-d`, `close-strategy-top-flop-10-w-90-d`, `close-strategy-top-flop-10-w-365-d`, `close-strategy-median-w-90-d`, `close-strategy-median-w-365-d` für die historischen Werte sowie `last-strategy-top-flop-5-w-1d`, `last-strategy-top-flop-10-w-1d`, `last-strategy-median-w-1d` für die Realtime-Werte (siehe Listing 8).



Ein Dashboard visualisiert die Analyse-Ergebnisse (Abb. 4).

Das Auge isst mit

Am Ende der „Nahrungskette" steht ein Micronaut-Microservice. Bei Micronaut handelt es sich um ein Framework, das, anders als beispielsweise Spring, von Anfang an für den Einsatz in serverseitigen Funktionen und Microservices entwickelt wurde.

Micronaut bietet sich aber nicht nur aufgrund seines modernen Aufbaus und geringen Speicherverbrauchs zur Implementierung von Microservices auf Java-Basis an. Es ermöglicht auch eine sehr einfache Anbindung an Kafka. Im konkreten Fall arbeitet der Service als Kafka-Consumer. Er lauscht somit auf den Topics, cacht jeweils den letzten aktuellen Stand und stellt die generierten Assoziationsregeln per REST-API der Außenwelt zur Verfügung (siehe Listing 9). Neben der REST-API liefert der Service auch noch ein einfaches Dashboard auf Grundlage von vue.js aus. Damit lassen sich die Regeln für die unterschiedlichen Zeiträume visualisieren (siehe Abbildung 4).

Fazit

Selbst bei der Verwendung von Samzas Low-Level-API lässt sich ein Anwendungsfall wie der oben beschriebene mit überschaubarem Entwicklungsaufwand umsetzen. Durch die von Samza vorgegebenen Restriktionen und die Fokussierung auf die Umsetzung wohldefinierter Stream-Tasks entsteht gut strukturierter und einfacher Code. Die Verwendung lokaler State Stores in Kombination mit dem Checkpointing-Mechanismus erfüllt die Anforderungen an Fehlertoleranz und Exactly-once-Verarbeitung. Wer jetzt auch noch an den Ergebnissen der hier durchgeführten Aktienanalyse interessiert ist, kann sich diese online im frei zugänglichen Dashboard ansehen (siehe ix.de/zunw). (akl@ix.de)

Quellen

Weiterführende Links unter https://ix.de/zunw

Peter Hoffmann

ist Geschäftsführer der PublicRabbit GmbH. Er berät im Bereich Softwarearchitektur und Data Mining.

Paul Watzlaw

ist zusammen mit Peter Hoffmann Geschäftsführer der PublicRabbit GmbH. Er unterstützt Unternehmen beim Design und der Entwicklung skalierbarer, verteilter Systeme.

Korrelationskoeffizient nach Bravais

Das Risiko im Fall von Kursverlusten lässt sich minimieren, indem man bei der Zusammensetzung eines optimalen Portfolios darauf achtet, Anlagen zu kombinieren, die eine möglichst niedrige Korrelation aufweisen. In der Regel wird hierzu ein Korrelationskoeffizient nach dem Bravais-Pearson-Verfahren berechnet.

Im Gegensatz zu dem in diesem Artikel vorgestellten Data-Mining-Verfahren, das Kurslisten auf Verbundwahrscheinlichkeiten hin untersucht, ermittelt man bei dem Bravais-Pearson-Verfahren einen Koeffizienten, der Aufschluss über die lineare Korrelation von Kursverläufen gibt. Der Koeffizient ist ein Maß für den Zusammenhang zwischen den Kursverläufen zweier Investments und kann Werte von -1 bis +1 annehmen. Liegt der Wert bei -1, besteht eine gegenläufige Entwicklung, ein Korrelationskoeffizient von plus +1 zeigt einen ähnlichen Kursverlauf an. Bei Werten von 0 ist kein besonderer Zusammenhang vorhanden. Abbildung 5 zeigt beispielsweise eine starke Korrelation zwischen Daimler und Continental und eine schwache Korrelation zwischen Daimler und Commerzbank für die Jahre 2014 und 2015.

WKN	WKN	Koeffizient
710000	543900	0.94
710000	CBK100	0.03

Korrelationskoeffizienten zwischen Daimler und Continental sowie zwischen Daimler und Commerzbank (Abb. 5)

Die Anwendung des Bravais-Pearson-Verfahrens liegt auf der Hand, da hier metrisch skalierte Daten vorliegen und, falls überhaupt ein Zusammenhang erkennbar ist, ein linearer oder nahezu linearer Zusammenhang angenommen werden darf. Eine Schwäche dieses Verfahrens ist, dass Bravais-Pearson andere Formen des Zusammenhangs, wie etwa einen quadratischen oder logarithmischen, nicht oder zumindest nicht vollständig aufdeckt. Ein niedriger Wert deutet also nicht notwendigerweise auf keinerlei Zusammenhang zwischen den untersuchten Variablen hin.