Projektdokumentation im Modul Semantic Web – Zusammenhang von Terroranschlägen mit Witschaft und Politik

Hochschule für Technik, Wirtschaft und Kultur Leipzig
Julius Seiffert - julius.seiffert@stud.htwk-leipzig.de

31.08.2019

Recherchefragestellung: Wie hängen Terroranschläge in Deutschland mit dem deutschen Aktienindex und den den deutschen Wahlverhalten zusammen.

Inhaltsverzeichnis

Abkürzungsverzeichnis

DAX deutscher Aktienindex

URL Uniform Resource Locator

API application programming interface

RDF resource description framework

1 Inhaltliche Interpretation der Fragestellung

Im Rahmen dieser Forschungsarbeit ist eine konkrete Interpretation, bzw. Schlussforderung natürlich nicht möglich. Alle Zusammenhänge die zwischen den Datenquellen gefunden wurden können durch Zufall entstanden sein. Natürlich kann es bei dieser Fragestellung auch direkte Zusammenhänge geben. Terroranschläge können in der Gesellschaft einer Nation Angst und Hass schüren, was sich wiederum auf das Wahlverhalten auswirken kann. Auch können Terroranschläge bestimmte Bereiche der Wirtschaft finanziell zu Gute kommen, allerdings auch genauso Schaden. Beispielsweise könnte der Bereich der Waffen- und Militärausrüstung davon profitieren, oder bei vorhandener Angst der Einzelhandel leiden.
Mithilfe dieser Arbeit soll herausgefunden werden, ob ein direkter Zusammenhang zwischen Terroranschlägen, der Wirtschaft und der politischen Meinung in Deutschland hergestellt werden kann. Hierzu werden der deutscher Aktienindex (DAX) unmittelbar vor und nach dem Anschlag benutzt, sowie die Daten der FORSA Sonntagsfrage unmittelbar vor und nach dem Ereignis.

2 Relevante Datenquellen

Relevante Datenquellen zur Auswertung sind die Terrordatenbank der University Maryland, die Website Yahoo Finance, welche historische Daten des DAX zum Download zur Verfügung stellt und die Sonntagsfrage des FORSA Instituts, welches nahezu jeden Sonntag eine Wahlumfrage erhebt.

2.1 Terrordatenbank der University Maryland

Die Datenbank der University Maryland beinhaltet globale Terroranschläge von 1970 bis 2017. Auf der Website werden die Daten als CSV-Datei zum Download angeboten und können auch direkt über eine Tabelle auf der Website angezeigt werden.
Die Anfragen an die Datenbank können mehrfach gefiltert werden. Die Filter werden auch in dem Uniform Resource Locator (URL) der Website und in dem URL zum Download angezeigt.

Link	`https://www.start.umd.edu/gtd/about/`
Datenformat	CSV, HTML
Schnittstelle	Webscarper, manueller Download
Lizenz	`https://www.start.umd.edu/gtd/terms-of-use/`
Open Data	⋆⋆⋆

2.2 Yahoo Finance

Die Website Yahoo Finance stellt sämtliche Börsenkurse und auch historische Daten davon bereit.
Yahoo Finance hat bis zum März 2019 eine application programming interface (API)-Schnittstelle der verfügbaren Finanzdaten angeboten. Leider wurde diese Schnittstelle eingestellt und die Daten können nur durch einen Webscarper, oder einen manuellen Download heruntergeladen werden. Eine Automatisierung des Downloads gestaltet sich durch Sicherheitsmaßnahmen der Website als schwierig. Aufgrund dessen wurden die die Tageshöchstsätze des DAX manuell als CSV-Datei heruntergeladen, welche bis zum 01.01.2004 zurückgeht.

Link	`https://de.finance.yahoo.com/`
Datenformat	CSV, HTML
Schnittstelle	manueller Download
Lizenz	`https://policies.oath.com/ie/de/oath/terms/otos/index.html`
Open Data	⋆⋆

2.3 Sonntagsfrage FORSA

Das FORSA Institut stellt zu jeden Sonntag die Sonntagsfrage. Bei dieser Frage werden die Befragten gefragt, welche Partei sie wählen würden, wenn am nächsten Sonntag die Bundeswahl wäre. Diese Daten gehen bis 1998 zurück. Es gibt allerdings auch Zeiträume, in denen keine Daten vorhanden sind.
Die Daten werden auf der Website `https://www.wahlrecht.de/` in HTML-Tabellen bereitgestellt.

Link	https://www.wahlrecht.de/umfragen/forsa.htm
Datenformat	HTML
Schnittstelle	Webscarper
Lizenz	
Open Data	⋆⋆

3 Vokabular

Für den Aufbau der Datenbasis wurde das resource description framework (RDF)-Vokabular verwendet. Der Präfix zu dem RDF-Vokabular ist *http://www.w3.org/1999/02/22-rdf-syntax-ns*

Für die spezifischen Daten wurde ein neues Vokabular erstellt. Dieses wurde unter dem Präfix *https://github.com/juuls28/semanticWEB* verwendet.

Listing 1: Präfixe

```
1       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
2       xmlns:j.0="https://github.com/juuls28/semanticWEB#"
```

3.1 Terroranschlag

Die Daten der Terroranschläge sind durch Ihre ID gekennzeichnet.
Aus der Terrordatenbank lassen sich viele Daten zu jedem Terroranschlag auslesen. Hier wurden nur die wichtigsten extrahiert.

happenedOn beschreibt das Datum im Format yyyy-MM-dd an dem das Ereignis geschehen ist.

inCountry beschreibt das Land in dem das Ereignis passiert ist als Zeichenkette.

inCity beschreibt die Stadt in dem das Ereignis passiert ist als Zeichenkette.

hasFatalitites beschreibt die Anzahl der getöteten Personen als ganze Zahl.

hasInjuries beschreibt die Anzahl der verletzten Personen als ganze Zahl.

Listing 2: Beispiel RDF Terroranschlag

```
1   <rdf:Description rdf:about="https://github.com/juuls28/semanticWEB#201608040029">
2       <j.0:hasInjuries>0</j.0:hasInjuries>
3       <j.0:hasFatalities>0</j.0:hasFatalities>
4       <j.0:inCity>Berlin</j.0:inCity>
5       <j.0:inCountry>Germany</j.0:inCountry>
6       <j.0:happenedOn>2016-08-04</j.0:happenedOn>
7    </rdf:Description>
```

3.2 DAX-Kurs

Die Daten des Aktienkurses sind durch das Wort DAX und das Datum eindeutig identifizierbar. Die Ressourcen haben daher die Form DAXyyyy-MM-dd.
Bei den historischen Daten des DAX-Kurses wird das Datum und der zugehörige Eröffnungskurs beachtet.

happenedOn beschreibt das Datum im Format yyyy-MM-dd an dem das Ereignis geschehen ist.

value beschreibt den Wer des Eröffnungskurses am Tag der Ressource als Fließkommazahl.

Listing 3: Beispiel RDF DAX

```
<rdf:Description rdf:about="https://github.com/juuls28/semanticWEB#DAX2009-11-02">
    <j.0:value>5410.609863</j.0:value>
    <j.0:happenedOn>2009-11-02</j.0:happenedOn>
</rdf:Description>
```

3.3 Wahlumfrage

Die Wahlumfrage wird durch das Datum im Format yyyy-MM-dd gekennzeichnet. Sie beinhaltet das Datum und eine Liste der Parteien mit den zugehörigen Prozentsatz, mit dem diese gewählt worden wären.

happendOn beschreibt das Datum im Format yyyy-MM-dd an dem das Ereignis geschehen ist.

outcomes beinhaltet eine Liste der Ressource Partei.

3.4 Partei

Die Ressource Partei wir durch das Datum im Format yyyy-MM-dd und den Parteinamen gekennzeichnet. Sie repräsentiert das Wahlergebnis zu jeder Partei an dem Datum der Wahl.

partyName beschreibt den Namen der Partei als Zeichenkette.

partyPercent beschreibt das Wahlergebnis (Prozentsatz) der Partei als Fließkommazahl.

Listing 4: Beispiel RDF Wahlumfrage

```
<rdf:Description rdf:about="https://github.com/juuls28/semanticWEB#2001-06-21">
    <j.0:outcomes>
      <rdf:Description rdf:about="https://github.com/juuls28/semanticWEB#2001-06-21Sonstige">
        <j.0:partyPercent>4.0</j.0:partyPercent>
        <j.0:partyName>Sonstige</j.0:partyName>
      </rdf:Description>
    </j.0:outcomes>
    <j.0:outcomes>
      <rdf:Description rdf:about="https://github.com/juuls28/semanticWEB#2001-06-21PDS">
        <j.0:partyPercent>5.0</j.0:partyPercent>
        <j.0:partyName>PDS</j.0:partyName>
      </rdf:Description>
    </j.0:outcomes>
    <j.0:outcomes>
      <rdf:Description rdf:about="https://github.com/juuls28/semanticWEB#2001-06-21FDP">
        <j.0:partyPercent>9.0</j.0:partyPercent>
        <j.0:partyName>FDP</j.0:partyName>
      </rdf:Description>
    </j.0:outcomes>
    <j.0:outcomes>
      <rdf:Description rdf:about="https://github.com/juuls28/semanticWEB#2001-06-21GRÜNE">
        <j.0:partyPercent>7.0</j.0:partyPercent>
        <j.0:partyName>GRÜNE</j.0:partyName>
      </rdf:Description>
    </j.0:outcomes>
    <j.0:outcomes>
      <rdf:Description rdf:about="https://github.com/juuls28/semanticWEB#2001-06-21SPD">
        <j.0:partyPercent>39.0</j.0:partyPercent>
        <j.0:partyName>SPD</j.0:partyName>
      </rdf:Description>
    </j.0:outcomes>
    <j.0:outcomes>
      <rdf:Description rdf:about="https://github.com/juuls28/semanticWEB#2001-06-21CDU/CSU">
        <j.0:partyPercent>36.0</j.0:partyPercent>
        <j.0:partyName>CDU/CSU</j.0:partyName>
      </rdf:Description>
    </j.0:outcomes>
    <j.0:happenedOn>2001-06-21</j.0:happenedOn>
</rdf:Description>
```

4 Extraktion relevanter Daten und Import in einen Triplestore

Die Datensätze wurde teilweise automatisch und teilweise manuell heruntergeladen und weiterverarbeitet. Als Basis dient hierzu ein Java-Projekt welches unter dem Github Repository `https://github.com/juuls28/semanticWEB` einsehbar ist.
Aus den einzelnen Daten wurde mit Hilfe des Apache Jena Frameworks ein Tripelstore gebildet und SPARQL-Abfragen ausgeführt.

4.1 Extraktion Terrordatenbank

Die CSV-Datei mit den Daten zu den Terroranschlägen kann direkt durch einem URL in das Programm geladen werden. Mit in der URL können Filterparameter wie Startdatum, Enddatum, Land, Stadt etc. angegeben werden.
Hier war nur das Start- und Enddatum relevant, welche dynamisch anhand des Suchzeitraums hinzugefügt werden. Das Land bleibt statisch auf Deutschland eingestellt. Eine Abfrage vom 01.01.2010 bis zum 01.01.2011 würde wie folgt aussehen: `https://www.start.umd.edu/gtd/search/ResultsCSV.aspx?csv=1&casualties_type=b&casualties_max=&start_year=2010&start_month=1&start_day=1&end_year=2011&end_month=1&end_day=1&dtp2=all&country=75`
Aus den CSV-Dateien konnte dann die Daten zu den Anschlägen ausgelesen werden. Hierbei wurden fehlerhafte Daten bereinigt. Wenn die Daten nicht bereinigt werden konnten, wurden sie nicht übernommen.

4.2 Extraktion DAX-Kurs

Der DAX-Kurs wurde manuell als CSV-Datei heruntergeladen und in das Projekt importiert. Es wurden alle Tagessätze vom 28.07.2004 bis zum 27.07.2018 heruntergeladen. Bei bedarf kann auch eine neue CSV-Datei in das Programm geladen werden. Die Daten konnten dann wie bei der Terrordatenbank in das Programm geladen und bereinigt werden.

4.3 Extraktion Wahlumfrage

Die Daten zur Wahlumfrage mussten direkt von der HTML-Tabelle auf der Website ausgelesen werden. Hierfür wurde die Bibliothek jsoup[1] verwendet. jsoup ist ein HTML-Parser mit dem Websiten nach bestimmten Merkmalen untersucht werden können. Mithilfe dieser Bibliothek konnten die Daten ausgelesen und in aufbereitet werden.
Leider werden auch auf dieser Website inkonsistente Daten bereitgestellt, weswegen bestimmte Datensätze aufbereitet werden mussten, oder nicht verwertbar waren.

5 Verlinkung von Ressourcen

Die Verschiedenen Ressourcen werden über das Datum, also über die Eigenschaft https://github.com/juuls28/semanticWEB#happenedOn/ verlinkt. Dazu wurde mit Hilfe des Frameworks Apache Jena ein RDF Model erstellt. Dieses Model fungiert als Tripel Store. Das Implementierung ist wurde in der Datei ModelBuilder.java vorgenommen und kann im Anhang Erstellung des Model eingesehen werde. Das ganze Model ist in dem Projekt unter dem Pfad /data/model.rdf exportiert worden, damit es auch in anderen Services wie den SPARQL Server Apache Jena Fuseki verwendet werden kann.

6 Anfrage an die Forschungswissensbasis

Im Rahmen der geschriebenen Software gibt es drei SPARQL-Anfragen. Da der Benutzer zuerst nach einem Zeitraum gefragt wird, in dem er Terroranschläge in Deutschland angezeigt bekommen soll, müssen diese Anschläge aus dem Tripelstore geladen werden.
Danach wählt der Benutzer einen Anschlag aus und bekommt die Daten zu diesem Anschläge mit der zweiten SPARQL Anfrage zurück.
Zum Schluss erfolgt die eigentliche und wichtigste SPARQL Anfrage, welche die Ressourcen verknüpft und Korrelationen aufzeigt.

6.1 SPARQL-Anfragen

SPARQL dient zur Abfrage von RDF-Daten. Diese Daten werden in Graphen gespeichert. Mithilfe von SPARQL können die Daten verbunden, Inhalte getestet und die Ergebnisse als Sets angezeigt werden.
Alle Anfragen werden anhand eines Beispiels aufgezeigt, um die Funktion zu verdeutlichen.

[1] jsoup ist eine OpenSource Bibliothek und ist unter der Verwendung der MIT-Lizenz benutzbar. Informationen hierzu gibt es unter `https://jsoup.org/`

6.1.1 Terroranschläge in einem bestimmten Zeitraum

Der Benutzer kann zu Beginn der Programmausführung einen Zeitraum auswählen, in dem die Daten zu den Terroranschlägen heruntergeladen werden.
Bei der ersten Anfrage werden alle Terroranschläge mit den zugehörigen Daten ausgegeben, die heruntergeladen wurden.(vgl. 8)

Listing 5: SPARQL 1

```
Prefix ns: <https://github.com/juuls28/semanticWEB#>
SELECT ?s ?c ?d ?o
WHERE { ?s ns:hasInjuries ?o .
  ?s ns:inCity ?c .
  ?s ns:happenedOn ?d .
}
ORDER BY ASC(?d)
```

6.1.2 Bestimmter Terroranschlag

Vor der zweiten Abfrage wird der User nach einen Terroranschlag gefragt, zu dem er die Verbindung zu DAX und Wahlumfrage erhalten will.
Es wird das Datum des Anschlags abgefragt und zurückgegeben.(vgl. 9)
Im Beispiel wurde der Anschlag des 29.09.2014 in Berlin genommen.

Listing 6: SPARQL 2

```
Prefix ns: <https://github.com/juuls28/semanticWEB#>
SELECT ?d
WHERE { <https://github.com/juuls28/semanticWEB#201409290078> ns:happenedOn ?d .
}
```

6.1.3 Verknüpfung mit dem DAX-Kurs und der Wahlumfrage

Die letzte SPARQL Anfrage liefert die Werte des DAX am ersten verfügbaren Tag, vor und nach des Anschlags, sowie die ersten Wahlumfragen vor und nach dem Anschlag.
Hierbei wird dynamisch das Datum des Anschlags in die Query mit einbezogen, um so die zugehörigen Ressourcen herauszufinden. Die Ressource der Wahlumfragen und der DAX-Kurse werden mithilfe vier Subqueries abgefragt. Aus den Ressourcen der Subqueries werden dann die Eigenschaften abgefragt und angezeigt.(vgl. 10)

Listing 7: SPARQL 3

```
Prefix ns: <https://github.com/juuls28/semanticWEB#>

SELECT ?firstStock ?secondStock ?vF ?vS ?firstPoll ?secondPoll ?p1Name ?p1Percent ?p2Name ?p2Percent
WHERE {
  {Select ?firstStock
    WHERE{
      ?firstStock ns:happenedOn ?date .
      ?firstStock ns:value ?o .
      Filter(?date < "2014-09-29")
    } Order By DESC (?date)
    Limit 1
  }
  {Select ?secondStock
    WHERE{
      ?secondStock ns:happenedOn ?date .
      ?secondStock ns:value ?o .
      Filter(?date > "2014-09-29")
    } Order By ASC (?date)
    Limit 1
  }
  ?firstStock ns:value ?vF .
  ?secondStock ns:value ?vS .
  {Select ?firstPoll
    WHERE{
      ?firstPoll ns:happenedOn ?date .
      ?firstPoll ns:outcomes ?o .
      Filter(?date < "2014-09-29")
    } Order By DESC (?date)
    Limit 1
  }
```

```
    {Select ?secondPoll
      WHERE{
        ?secondPoll ns:happenedOn ?date .
        ?secondPoll ns:outcomes ?o .
        Filter(?date > "2014-09-29")
      } Order By ASC (?date)
      Limit 1
    }
    ?firstPoll ns:outcomes ?outF .
    ?outF ns:partyName ?p1Name .
    ?outF ns:partyPercent ?p1Percent .
    ?secondPoll ns:outcomes ?outS .
    ?outS ns:partyName ?p2Name .
    ?outS ns:partyPercent ?p2Percent .
    Filter(?p1Name = ?p2Name)
  }
```

6.2 Ergebnis der Anfragen

Die Ergebnisse der SPARQL-Anfragen wurden aus Platzgründen modifiziert. Hierdurch leidet leider die Darstellung und Übersichtlichkeit.

6.2.1 Ergebnis Abfrage 1

Das Ergebnis aus SPARQL 1 mit dem Beispiel vom 01.01.2011 bis zum 01.01.2015.

Listing 8: Ergebnis SPARQL 1

```
Number: 0
Id: https://github.com/juuls28/semanticWEB#201103020018 Date: 2011-03-02  City: Frankfurt
Number: 1
Id: https://github.com/juuls28/semanticWEB#201110100012 Date: 2011-10-10  City: Berlin
Number: 2
Id: https://github.com/juuls28/semanticWEB#201110100013 Date: 2011-10-10  City: Brieselang
Number: 3
Id: https://github.com/juuls28/semanticWEB#201110110004 Date: 2011-10-11  City: Berlin
Number: 4
Id: https://github.com/juuls28/semanticWEB#201110120010 Date: 2011-10-12  City: Staaken
Number: 5
Id: https://github.com/juuls28/semanticWEB#201110130007 Date: 2011-10-13  City: Staaken
Number: 6
Id: https://github.com/juuls28/semanticWEB#201112070001 Date: 2011-12-07  City: Frankfurt
Number: 7
Id: https://github.com/juuls28/semanticWEB#201201230007 Date: 2012-01-23  City: Magdeburg
Number: 8
Id: https://github.com/juuls28/semanticWEB#201205010017 Date: 2012-05-01  City: Berlin
Number: 9
Id: https://github.com/juuls28/semanticWEB#201205140051 Date: 2012-05-14  City: Potsdam
Number: 10
Id: https://github.com/juuls28/semanticWEB#201210150015 Date: 2012-10-15  City: Berlin
Number: 11
Id: https://github.com/juuls28/semanticWEB#201212100015 Date: 2012-12-10  City: Bonn
Number: 12
Id: https://github.com/juuls28/semanticWEB#201401020034 Date: 2014-01-02  City: Berlin
Number: 13
Id: https://github.com/juuls28/semanticWEB#201407020046 Date: 2014-07-02  City: Berlin
Number: 14
Id: https://github.com/juuls28/semanticWEB#201407290016 Date: 2014-07-29  City: Wuppertal
Number: 15
Id: https://github.com/juuls28/semanticWEB#201408100063 Date: 2014-08-10  City: Bielefeld
Number: 16
Id: https://github.com/juuls28/semanticWEB#201408110052 Date: 2014-08-11  City: Berlin
Number: 17
Id: https://github.com/juuls28/semanticWEB#201408250087 Date: 2014-08-25  City: Berlin
Number: 18
Id: https://github.com/juuls28/semanticWEB#201408280032 Date: 2014-08-28  City: Berlin
Number: 19
Id: https://github.com/juuls28/semanticWEB#201408300060 Date: 2014-08-30  City: Oldenburg
Number: 20
Id: https://github.com/juuls28/semanticWEB#201408300061 Date: 2014-09-04  City: Moelln
Number: 21
Id: https://github.com/juuls28/semanticWEB#201409290078 Date: 2014-09-29  City: Berlin
Number: 22
Id: https://github.com/juuls28/semanticWEB#201410110069 Date: 2014-10-11  City: Bad Salzuflen
Number: 23
Id: https://github.com/juuls28/semanticWEB#201410120059 Date: 2014-10-12  City: Gross Lusewitz
Number: 24
Id: https://github.com/juuls28/semanticWEB#201412120084 Date: 2014-12-12  City: Vorra
Number: 25
Id: https://github.com/juuls28/semanticWEB#201501110002 Date: 2015-01-11  City: Hamburg
```

6.2.2 Ergebnis Abfrage 2

Das Ergebnis der Abfrage nach einem Ereignis. Als Beispiel wurde der 29.09.2014 genommen.

Listing 9: Ergebnis SPARQL 2

```
Date: 2014-09-29
```

6.2.3 Ergebnis Abfrage 3

Das Ergebnis der Abfrage nach den Werten der der DAX-Kurse und Wahlumfragen.
Im Ergebnis werden werden pro Nummer immer der DAX-Kurs vor und nach dem Anschlag angezeigt, sowie das Umfrageergebnis einer Partei vor und nach dem Anschlag. Der DAX-Kurs und die Namen der Ressourcen bleiben also immer gleich, nur die Parteien ändern sich mit Ihrem Umfrageergebnis.

Listing 10: Ergebnis SPARQL 3

```
Number: 0
First Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-26
Value: 9500.549805
Second Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-30
Value: 9446.80957
First Poll: https://github.com/juuls28/semanticWEB#2014-09-23
Name: AfD
Value: 10.0
Second Poll: https://github.com/juuls28/semanticWEB#2014-10-01
Name: AfD
Value: 9.0
Number: 1
First Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-26
Value: 9500.549805
Second Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-30
Value: 9446.80957
First Poll: https://github.com/juuls28/semanticWEB#2014-09-23
Name: LINKE
Value: 9.0
Second Poll: https://github.com/juuls28/semanticWEB#2014-10-01
Name: LINKE
Value: 8.0
Number: 2
First Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-26
Value: 9500.549805
Second Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-30
Value: 9446.80957
First Poll: https://github.com/juuls28/semanticWEB#2014-09-23
Name: FDP
Value: 2.0
Second Poll: https://github.com/juuls28/semanticWEB#2014-10-01
Name: FDP
Value: 2.0
Number: 3
First Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-26
Value: 9500.549805
Second Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-30
Value: 9446.80957
First Poll: https://github.com/juuls28/semanticWEB#2014-09-23
Name: GRÜNE
Value: 8.0
Second Poll: https://github.com/juuls28/semanticWEB#2014-10-01
Name: GRÜNE
Value: 9.0
Number: 4
First Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-26
Value: 9500.549805
Second Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-30
Value: 9446.80957
First Poll: https://github.com/juuls28/semanticWEB#2014-09-23
Name: SPD
Value: 22.0
Second Poll: https://github.com/juuls28/semanticWEB#2014-10-01
Name: SPD
Value: 23.0
Number: 5
First Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-26
Value: 9500.549805
Second Stock: https://github.com/juuls28/semanticWEB#DAX2014-09-30
Value: 9446.80957
First Poll: https://github.com/juuls28/semanticWEB#2014-09-23
Name: CDU/CSU
Value: 42.0
```

```
64  Second Poll: https://github.com/juuls28/semanticWEB#2014-10-01
65  Name: CDU/CSU
66  Value: 42.0
```

Es lässt sich in dieser Abfrage nun sagen, dass der DAX um 54 Punkte verloren hat und das die Wahlumfrage wie folgt aussieht:

Partei	Ergbnis 1	Ergebnis 2
AfD	10.0	9.0
Linke	9.0	8.0
Grüne	8.0	9.0
SPD	22.0	23.0
CDU/CSU	42.0	42.0

7 Interpretation und Zusammenfassung

Abschließend zu dem Projekt lässt sich sagen, dass eine Verbindung zwischen den Ressourcen möglich, aber vermutlich nicht sinnvoll, oder besonders aussagekräftig ist.
Im Rahmen dieses Projektes war es nicht möglich alle Anschläge zu überprüfen und Schlussfolgerungen zu ziehen. Jedoch lässt sich im Beispiel sehen, dass es beim DAX ein leichtes Minus gab, dies jedoch auch von ganz anderen Faktoren kommen kann.
Die Parteien hatte in den Umfragen nur leichte Änderungen, woraus schließen lässt, dass ein Anschlag nicht die politische Meinung zu stark beeinflusst. Dies kann vermutlich nur über einen größeren Zeitraum geschehen. Bei dem Beispiel ist sogar teilweise das Gegenteil ersichtlich. Die rechten Parteien, von denen man meinen könnte, sie könnten von Terroranschlägen profitieren, verlieren an Prozentpunkte, genauso wie linken Parteien, während Volksparteien an Prozentpunkte gewinnen.
Es wurden die Ergebnisse größerer Anschläge angesehen, wie der Terroranschlag in Berlin vom 19.12.2016. Hier hat der DAX-Kurs zugelegt und die politische Meinung, repräsentiert durch die Sonntagsfrage hat sich kaum verändert.
Um eine wirkliche Verbindung zwischen den Ereignissen herstellen zu können benötigt man mehr Daten, wie der Allgemeine Verlauf des DAX in den letzten Jahren, oder einen Wählertrend der vergangenen Jahre.

8 Ausblick

In dem Projekt können noch einige Verbesserungen implementiert werden.
Beispielsweise wird aktuell nur mit der Konsole gearbeitet. Einen Webserver verfügt das Projekt aktuell bereits, allerdings muss noch das zugehörige Web Frontend gebaut werden. Hier kann man die Benutzereingaben besser abfragen und die Ergebnisse übersichtlicher darstellen.
Wie im Kapitel Interpretation und Zusammenfassung bereits beschrieben können auch weitere Auswertungsdaten in das Projekt mit einfließen, damit ein engerer Zusammenhang zwischen den Ressourcen beschrieben werden kann.
Auch Auswertungen auf kleineren Regionen wäre denkbar. Hier gibt es allerdings Probleme mit der Repräsentation der Wirtschaft in Bundesländern, oder Landkreisen. Wahlumfragen werden in diesen Bereichen leider auch nicht regelmäßig erhoben.

A Erstellung des Model

A.1 Terrormodel

Listing 11: Terrormodel

```
1  private void createTerrorModel(){
2
3          Property date = model.createProperty(ns,"happenedOn");
4          Property city = model.createProperty(ns,"inCity");
5          Property country = model.createProperty(ns,"inCountry");
6          Property fatalities = model.createProperty(ns,"hasFatalities");
7          Property injuries = model.createProperty(ns, "hasInjuries");
8
9          for (Terror t : attacks) {
10             Resource res = model.createResource(ns + t.getId());
11
12             res.addProperty(date,t.getDate().toString());
13             res.addProperty(country,t.getCountry());
14             res.addProperty(city,t.getCity());
15             res.addProperty(fatalities, String.valueOf(t.getFatalities()));
16             res.addProperty(injuries, String.valueOf(t.getInjured()));
17
18         }
19 }
```

A.2 Stockmodel

Listing 12: Stockmodel

```
1  private void createStockModel(){
2
3          Property date = model.createProperty(ns,"happenedOn");
4          Property value = model.createProperty(ns,"value");
5
6          for (Stock s : stocks) {
7              Resource res = model.createResource(ns + s.getId());
8
9              res.addProperty(date, s.getDate().toString());
10             res.addProperty(value, String.valueOf(s.getValue()));
11
12         }
13 }
```

A.3 Pollmodel

Listing 13: Pollmodel

```
1  private void createPollModel(HashMap<LocalDate, Poll> map){
2
3          Property date = model.createProperty(ns,"happenedOn");
4          Property partyName = model.createProperty(ns,"partyName");
5          Property partyPercent = model.createProperty(ns,"partyPercent");
6          Property outcomes = model.createProperty(ns,"outcomes");
7
8
9          for (Map.Entry<LocalDate, Poll> entry: map.entrySet()) {
10             Resource poll = model.createResource(ns + entry.getKey().toString());
11             poll.addProperty(date, String.valueOf(entry.getKey()));
12
13             for(Party p : entry.getValue().getOutcomes()){
14                 Resource party = model.createResource(ns + entry.getKey().toString()+p.getName());
15
16                 party.addProperty(partyName, p.getName());
17                 party.addProperty(partyPercent, String.valueOf(p.getPercent()));
18
19                 poll.addProperty(outcomes, party);
20
21             }
22
23         }
24
25 }
```

Listings