Ziel war es einen Datenstrom von News Artikel unbekannter Sprache und Quelle nach einer im Vorne herein nicht definierten Anfrage zu filtern. Folgende Schritte waren dazu notwendig.

Zu den Daten:

- Die News Artikel sind in JSON Dateien verpackt und umfassen bereits mehrere Annotationen wie NER(Named Entity Recognition), POS(Part of Speech), DEP (Dependency Parse), Lemmatization etc..
- Größe: 400 GB, 650758 News Artikel größtenteils englischer Sprache aus verschiedenen Quellen

Vorgehensweise:

1) Filtern und laden der annotierten Daten in Elasticsearch
2) Implementierung eines Document Similarity Algorithmus der anhand von Wortvektoren Ähnlichkeiten zwischen der Suchanfrage und einem News Artikel feststellt.
    a) Erstellen der Wortvektoren
        1) News Artikel(650758) aus Elasticsearch einlesen.
        2) Mit dem Standford Parser die Artikel in Sätze trennen.
        3) Sätze transformieren:
            - Lemmatization (Wortstämme bilden)
            - Named Entity Recognition (Eigennamen erkennen und als einzelnes Wort abbilden e.g. New York → New_York)
            - alle Wörter werden in Kleinbuchstaben umgewandelt
            - Links werden heraus gefiltert e.g. www.google.de
            - Andere sprachliche Konzepte wie Distanzen, Dauer, Geld, Nummern, Prozent und Uhrzeit werden konsolidiert e.g. 5 € → money
        4) Wortvektoren erstellen: Ein window mit einer Größe von 9 über die Sätze laufen lassen. Das bedeutet, dass gemäß dem CBOW(Continuous Bag of Words) Algorithmus alle Wörter die vier Positionen nach oder vor dem aktuellen Wort im Satz stehen, in einer Matrix vermerkt werden. Diese Matrix heißt Co-Occurrence Matrix und gibt an, welche Wörter zusammen in einem Text aufgetreten sind. Das ist nützlich um Rückschlüsse auf den Kontext des Wortes führen zu können. Beispielsweise wird "Arzt" häufig zusammen mit dem Wort "Krankenhaus" auftauchen, sie besitzen also einen gemeinsamen Kontext. Beipiel für einen Wortvektor: *metallica: {"fiamma":1," ago":1," concert":1, "billy_eichner":1, "fan":1, "band":1 …*
        5) Wortvektoren filtern: Es werden nur die Wörter und deren Vektoren behalten welche folgenden Wortarten entsprechen: `JJ JJR JJS NN NNS NNP NNPS PDT RB RBR RBS RP VB VBD VBG VBN VBP VBZ VBG.` Dies sind hauptsächlich Verben, Adjektive und Nomen.

Die Intention ist, dass Wörter welche nicht in diese Kategorien fallen wie "to" oder "and" wenig zum Kontext des Wortes beitragen und somit vernachlässigt werden können. Somit wird auch die Matrix kleiner gehalten, was die Performance steigert.

6) Die fertige Co-Occurrence Matrix mit 463467 Zeilen (entspricht der Anzahl an Wortvektoren) wird nun in einer Avro Datei gespeichert, damit sie bei einem erneuten Programmstart wiederverwendet werden kann.

b) Erstellen von Dokumentvektoren
   1) Um nun die Ähnlichkeit zwischen einer Suchanfrage wie "Car crashes in New York" und einem anderen beliebigen Satz heraus zu finden werden Dokumentvektoren gebildet. Hierzu werden die Anfrage als auch die Sätze abermals transformiert (siehe 2 a 3). Unsere Suchanfrage sieht nach der Transformation so aus: "car crash in new_york".
   2) Anschließend wird der transformierte Satz in Token zerlegt. Die einzelnen Token(Wörter) werden per lookup in der Avro Datei in Wortvektoren überführt. Diese werden dann aufsummiert und das Ergebnis ist ein Vektor welcher einen Satz repräsentiert.

c) Erreichen der Ähnlichkeit zweier Dokumente
   1) Da Dokumentvektoren eine numerische Repräsentation darstellen, können mathematische Methoden anwendet werden, welche die Ähnlichkeit zwischen diesen feststellt.
   2) Hierzu wird die Cosine similarity implementiert, welche den Kosinus des eingeschlossenen Winkels der Vektoren errechnet. $\frac{A \cdot B}{\|A\|\|B\|}$
   3) Da $\|B\|$ die Länge des zweiten Vektors repräsentiert und der Term im Nenner steht, werden lange Sätze tendenziell mit einer geringeren Ähnlichkeit bewertet. Dies ist einerseits sinnvoll, da der Zähler ebenfalls von der Länge des Satzes profitiert(soll an dieser Stelle nicht weiter erklärt werden). Andererseits wird die Kernaussage nicht weniger relevant, nur weil viele zusätzliche Informationen in dem Satz stecken. Deshalb haben wir uns dazu entschieden, die Formel folgend abzuändern. $\frac{A \cdot B}{\|A\|\|B\|^{0,7}}$. Somit wurde erreicht, dass die Relevanz des Satzes nicht mehr linear sondern unter-proportional von der Länge abhängig ist. Damit das Ergebnis wieder zwischen 0 und 1 ist, wurde außerdem normiert.

3) Anschließend wurde eine Routine entwickelt, welche generisch Anfragen vom Nutzer einliest und zu Testzwecken die Ähnlichkeit zwischen der Anfrage und News Artikeln in Elasticsearch einliest.