Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Bachelorarbeit Institut für angewandte Informationstechnologie

Semantische Suche von juristischen Dokumenten mittels Word Embedding und Netzwerkanalyse

Autoren	Pavel Eigenmann Ravivarnen Sivasothilingam
Hauptbetreuung	Mark Cieliebak
Nebenbetreuung	Don Tuggener
Industriepartner	Weblaw AG
Externe Betreuung	Blaise Dévaud Christian Sprecher
Datum	08.06.2018

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Erklärung betreffend das selbständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:	Unterschriften:
Zürich, 08.06.2018	Pavel Eigenmann
……	……...
	Ravivarnen Sivasothilingam
	……
	……

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Bachelorarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Zusammenfassung

Das Ziel dieser Bachelorarbeit ist es, eine neue Art der Suche nach juristischen Dokumenten zu eruieren und einen lauffähigen Prototyp zu bauen. Es sollte möglich sein, einen kompletten Text, zum Beispiel einen Sachverhalt oder Gerichtsentscheid, in die Suche einzugeben. Mithilfe der Netzwerkanalyse oder des Word-Embedding-Verfahrens werden dem Benutzer der Suche ähnliche juristische Dokumente angezeigt. Die Resultate werden nach ihrer Relevanz sortiert.

Für den Prototyp wurden circa 100'000 deutsche Gerichtsentscheide vorselektiert. Die Dokumente wurden bereinigt und für das Information Retrieval mit Elasticsearch indexiert. Es wurde eine Netzwerkanalyse mit NetworkX auf den Gerichtsentscheiden durchgeführt. Mithilfe von PageRank und einem eigens für diese Bachelorarbeit implementierten Algorithmus konnten die Gerichtsentscheide nach ihrer Relevanz gewichtet werden.

Um eine Suche nach ähnlichen Dokumenten mit einem neu verfassten Sachverhalt möglich zu machen, wurden Word-Embedding-Verfahren angewendet. Für die Word Embedding wurde die Library „fastText" verwendet. Für die Auswertung der unterschiedlichen Verfahren wurden von einem Juristen mehrere Testfälle erstellt. Ein Testfall beinhaltet die Eingabe eines Sachverhalts oder Gerichtsentscheids und die Ausgabe der zehn ähnlichsten Gerichtsentscheide. Mit dem Kendall-Score konnten die generierten Ranglisten mit den Testfällen verglichen werden. Das fastText-Verfahren mit Skipgram und 300 Dimensionen erreichte das höchste Ergebnis.

Zudem wurde ein Experiment mit einem Juristen durchgeführt. Dabei wurden ihm die Resultate von acht unterschiedlichen Verfahren in zufälliger Reihenfolge präsentiert. Der Jurist musste jeden Gerichtsentscheid auf einer Skala von 1 bis 5 bewerten. Dabei erreichte fastText die beste Bewertung. Das Resultat der Netzwerkanalyse-Verfahren war niedrig.

Das Ergebnis dieser Bachelorarbeit ist ein lauffähiger Prototyp, welcher als Input einen Sachverhalt oder Gerichtsentscheid entgegennimmt und als Output die zehn ähnlichsten Gerichtsentscheide anzeigt. Diese Bachelorarbeit wurde in Zusammenarbeit dem Wirtschaftspartner Weblaw AG durchgeführt.

Abstract

The aim of this bachelor thesis is to find a new way of searching for legal documents and to build an executable prototype. It should be possible to enter a complete text such as a case or a court decision in the search. With the help of the network analysis or the Word Embedding procedure similar legal documents are displayed to the user. The results are sorted according to their relevance.

Approximately 100,000 German court decisions were pre-selected for the prototype. The documents were cleaned up and indexed for Information Retrieval with Elasticsearch. A network analysis was carried out with NetworkX on the documents. With the help of PageRank and an algorithm implemented especially for this bachelor thesis the nodes could be weighted according to their relevance.

In order to enable a search for similar documents with a newly written subject matter Word Embedding procedures were used. The "fastText" library was used for the Word Embeddings. Several test cases were created by an legal expert to evaluate the different procedures. A test case contains the input of a fact or court decision and the output of the ten most similar court decisions. With the Kendall Score the generated rankings could be compared with the test cases. The fastText procedure with Skipgram and 300 dimensions achieved the highest result.

An experiment was conducted with a legal expert. The results of eight different procedures were presented to the legal expert in random order. The legal expert had to evaluate each court decision on a scale of 1 to 5. fastText received the best rating. The result of the network analysis procedures was low. This can be attributed to the fact that the court decisions linked inside a court decision are rarely similar. Usually individual sentences from a court decision are quoted, the subject and the content of the texts are often different.

The result of this bachelor thesis is an executable prototype, which receives a case or court decision as input and displays the ten most similar court decisions as output. This bachelor thesis was carried out in cooperation the industry partner Weblaw AG.

Vorwort

Die Fragestellung für diese Arbeit entstand durch die Idee, einen intelligenten Bot zu programmieren, welcher Rechtsfragen beantworten kann. Nach intensiven Konversationen mit Karl Moritz Hermann und Ekaterina Kochmar, beides Experten in diesem Fachgebiet, erwies sich die Fragestellung als sehr ambitioniert.

Der Computer muss lange Texte verstehen und zusammenfassen können, um Rechtsfragen richtig beantworten zu können. Um dieses Problem zu lösen, müssen zuerst eine neue Speicherarchitektur und ein neuer Lese-Mechanismus entwickelt werden. In seiner Arbeit „Teaching Machines to Read and Comprehend“ beschreibt Karl Moritz Hermann die Schwierigkeiten und Limitationen bei langen Texten. Dabei ist das grösste Problem, dass meist genügend Trainings- und Testdaten fehlen [1].

Um eine grosse Menge an Rechtsdokumenten zu bekommen, kam nur eine Partnerschaft mit einem Industriepartner in Frage. Dank dem Rechtsanwalt Marc Fischer wurde uns die Firma Weblaw vermittelt. Weblaw hat eine umfassende Sammlung von Gerichtsdokumenten und ist eines der erfolgreichsten Legal-Tech-Unternehmen der Schweiz.

Mit Franz Kummer dem CEO von Weblaw AG wurden erste Ideen ausgetauscht, wie die Bachelorarbeit der Firma einen Mehrwert bieten kann. Dabei wurde schnell festgestellt, dass die firmeninterne Suche von Weblaw AG, Lawsearch, erweitert werden sollte. Gemeinsam mit Christian Sprecher und Blaise Dévaud, welche uns während der Bachelorarbeit betreut haben, wurde die Fragestellung weiter verfeinert. Wir sind sehr dankbar für ihre Bemühungen.

Ein grosser Dank geht an die Betreuer Dr. Mark Cieliebak und Dr. Don Tuggener, die uns während der gesamten Bachelorarbeit begleitet und unterstützt haben.

Inhaltsverzeichnis

1 Einleitung

Ein grosser Teil der Arbeit eines Rechtsanwalts beschränkt sich auf die Recherche zu einem Rechtsfall, um einen Mandanten möglichst gut zu beraten und zu verteidigen. Dabei kann der Rechtsanwalt auf öffentlich zugängliche Gerichtsentscheide, Gesetzesartikel und Literatur zurückgreifen. Sein Ziel ist es, möglichst ähnliche Gerichtsentscheide zu einem bestimmten Fall eines Mandanten zu finden. An dieser Lösung, möglichst ähnliche Gerichtsentscheide ausgehend von einem Sachverhalt oder Gerichtsentscheid zu finden, wurde im Rahmen der vorliegenden Bachelorarbeit, die einen Zeitraum von drei Monaten umfasste, geforscht.

Dabei wurden am Anfang die Suchmaschine Lawsearch und die Daten der Firma Weblaw AG genauer untersucht. Die Einleitung gibt einen Überblick über den aktuellen Stand von Lawsearch. Die Erweiterung von Lawsearch wird beschrieben und die Interessen von Weblaw AG an der Arbeit werden eruiert. Aus diesen Erkenntnissen folgen die Aufgabenstellung und die Abgrenzungen. Die Zielsetzung und der Aufbau der Bachelorarbeit werden dokumentiert.

1.1 Ausgangslage

Die Ausgangslage gibt einen Überblick über die Funktionsweise der Suchmaschine Lawsearch. Auch wird die Erweiterung von Lawsearch, mit der sich diese Bachelorarbeit beschäftigt, vorgestellt.

Lawsearch ist eine Stichwort-basierte Suchmaschine über Rechtsdokumente der Firma Weblaw AG. Mithilfe von Lawsearch können Dokumente über die Eingabe von Stichwörtern gesucht werden. Der User gibt die für ihn relevanten Begriffe ein und Lawsearch sucht in den indexierten Dokumenten nach einer Übereinstimmung der Wörter. Es können zudem noch weitere Attribute hinzugefügt werden, um die Suchanfrage zu präzisieren und die Anzahl der ausgegebenen Dokumente zu minimieren.

Innerhalb von Gerichtsentscheiden wird aber auf andere Gerichtsentscheide verlinkt. Dadurch ist es möglich, ein Netzwerk zwischen den Gerichtsentscheiden aufzuzeichnen. Mithilfe der Netzwerkanalyse können die Verbindungen zwischen den Texten analysiert und so relevante Dokumente angezeigt werden. Auch ist es möglich, Texte mithilfe von Word Embedding zu

vergleichen. Dabei werden Wörter als Vektoren repräsentiert. Zwischen zwei Texten werden dann die Vektoren verglichen und eine Ähnlichkeit wird definiert.

Die Zielsetzung dieser Arbeit ist es daher, Lawsearch um die Fähigkeit zu erweitern, Dokumente zu vergleichen und eine Rangliste von ähnlichen Dokumenten aufzubauen.

1.2 Zielsetzung

In der Zielsetzung wird das Ziel dieser Bachelorarbeit genauer definiert. Auch wird der generierte Nutzen für die Firma Weblaw beschrieben.

Das Ziel dieser Arbeit ist es, eine Relation zwischen Dokumenten zu finden, um für ein Ausgangsdokument die ähnlichsten Dokumente sortiert anzuzeigen. In dieser Arbeit wurden zwei Verfahren implementiert, miteinander verglichen und kombiniert.

Einerseits wurde die Netzwerkanalyse angewendet, um einen Überblick über die verlinkten Gerichtsentscheide zu erhalten. Dann wurde PageRank und ein eigenes Scoring-Verfahren auf dem Netzwerk angewendet, um für jedes Dokument im Netzwerk den Score zu berechnen. Je höher der Score ist, desto wichtiger ist das jeweilige Dokument.

Andererseits wurden sowohl Word Embedding als auch N-grams verwendet, um Wörter und Texte als Vektoren darzustellen. Damit können zwei Texte miteinander auf ihre Ähnlichkeit verglichen werden. Als Trainingsdaten für den Algorithmus dienen Gerichtsentscheide, die von der Firma Weblaw zur Verfügung gestellt wurden. Um das Ziel zu erreichen, wurden beide Verfahren einzeln und in Kombination mit den Testdaten verglichen. Die Testdaten sind von einem Juristen erstellte Ranglisten. Ein Testfall besteht aus einem Eingangsdokument und den Top 10 der ähnlichsten Dokumente.

Der Benutzer wählt mithilfe von Lawsearch, und der in dieser Arbeit beschriebenen, erweiterten Suche, einen Gerichtsentscheid aus. Vom betrachteten Dokument aus werden ihm nun die Top 10 der ähnlichsten Dokumente angezeigt. Der Benutzer kann dann ein Dokument auswählen, ohne die Dokumentenansicht verlassen zu müssen. Diese Funktionalität erweitert die Suche von Lawsearch und macht Lawsearch für den Nutzer attraktiver. Der Benutzer profitiert von der Benutzerfreundlichkeit. Die Erweiterung impliziert ein neues Benutzererlebnis,

welches man von YouTube gewohnt ist. Der Benutzer findet die für ihn relevanten Dokumente dadurch schneller und besser als mit der aktuellen Stichwortsuche von Lawsearch. Zudem kann der Benutzer nun einen ganzen Sachverhalt eingeben, woraufhin ihm Gerichtsentscheide angezeigt werden, welche ähnliche Sachverhalte beinhalten.

Aus diesem konkreten Ziel, dem Benutzer von Lawsearch einen Mehrwert zu bieten, ist die konkrete Aufgabenstellung entstanden.

1.3 Aufgabenstellung

Aus dem Ziel, möglichst ähnliche Dokumente zu einem bestimmten Sachverhalt oder Gerichtsentscheid zu finden, ergibt sich folgende Aufgabenstellung:

„Wie gut können ähnliche juristische Dokumente zu einem bestehenden Dokument mithilfe moderner Natural-Language-Processing- und Netzwerkanalyse-Verfahren gefunden werden im Vergleich zur bestehenden Lösung basierend auf der Volltextsuche?"

Um dieser Aufgabenstellung einen Rahmen zu geben, damit der Zeitplan eingehalten werden kann, wurden Abgrenzungskriterien definiert.

1.4 Abgrenzungen

Die Erstellung der Bachelorarbeit ist auf drei Monate beschränkt. Im Vorfeld wurden mit dem Auftraggeber folgende Abgrenzungskriterien definiert:

- Mit dem Unterzeichnen einer Gemeinhaltungsvereinbarung mit der Firma Weblaw AG werden die Daten und Informationen, welche zur Verfügung gestellt werden, vertraulich behandelt.
- Der in der Arbeit erstellte Prototyp und der Code müssen nicht produktiv einsatzfähig sein.
- Es bestehen keine Einschränkungen bezüglich verwendeter Technologien und Algorithmen.
- Die Implementierung der Algorithmen wird, wo möglich, aus bereits bestehenden Bibliotheken oder Frameworks übernommen.

1.5 Vorgehensweise

Die Vorgehensweise definiert die Struktur der Bachelorarbeit und das Vorgehen beim Erarbeiten der Themen. Der Data-Mining-Prozess kann beschrieben werden mit dem Cross Industry Standard Process for Data Mining (CRISP-DM). Diese Struktur wird in „Data Science for Business“ von Foster Provost und Tom Fawcett genauer beschrieben [2].

1.5.1 Geschäftsverständnis

Im ersten Schritt wird versucht, das Problem zu verstehen und zu beschreiben. Um eine passende Lösung zu finden, sind meist mehrere Iterationen notwendig. Das Business-Problem wird als ein oder mehrere Data-Science-Probleme formuliert.

1.5.2 Datenverständnis

Um die Daten zu verstehen, wurden die folgenden Schritte ausgeführt:

- Benötigte Daten aus den verfügbaren Rohdaten auswählen, die das Business-Problem lösen
- Zusammenführen unterschiedlicher Daten aus vielen unterschiedlichen Quellen
- Ermittlung möglicher Probleme mit der Datenqualität

1.5.3 Datenvorbereitung

Um die analytischen Technologien anzuwenden, müssen die Daten in eine bestimmte Form gebracht oder gewisse Vorbedingungen müssen erfüllt werden. Die Daten müssen auf einer Umgebung bereitgestellt werden, auf der man effizient auf sie zugreifen kann. Je nach verwendetem Algorithmus müssen unterschiedliche Natural-Language-Processing-Verfahren auf den Daten angewendet werden, um sie in die gewünschte Form zu bringen.

1.5.4 Modellierung

Die Modellierung ist der primäre Schritt. Hier werden Algorithmen auf den Daten angewendet, die das Business-Problem lösen. Es werden Parameter optimiert, um das bestmögliche Resultat zu erhalten.

1.5.5 Evaluation

Die Resultate werden überprüft. Es wird geschaut, ob die gewonnenen Informationen valide sind. Die Resultate des Modeling müssen das Business-Problem lösen, die Anforderungen erfüllen und dabei helfen, bestmöglich die Aufgabenstellung der Bachelorarbeit zu beantworten. Die Resultate müssen für alle Stakeholder verständlich präsentiert sein.

1.5.6 Development

Die geeigneten Techniken samt den Parametern werden in die Produktivumgebung eingebaut. Dieser Schritt fällt bei dieser Bachelorarbeit weg.

Nach jedem Schritt oder einer jeden Iteration stösst man möglicherweise auf eine neue Erkenntnis oder Idee. Das Projekt sollte flexibel genug sein, einen Schritt oder eine Iteration zurückzugehen und diese Idee auszuprobieren.

Bevor die einzelnen Schritte des CRISP-DM-Verfahrens beschrieben werden, wird eine Übersicht über die theoretischen Grundlagen gegeben. Diese werden benötigt, um die grundlegenden Konzepte in dieser Bachelorarbeit zu verstehen.

2 Theoretische Grundlagen

Als theoretische Grundlage für diese Bachelorarbeit dienten die wissenschaftliche Arbeit von Karl Moritz Hermann [1] und die Arbeit von Lawrence Page and Sergey Brin „The PageRank Citation Ranking: Bringing Order to the Web“ [3].

Das Buch „Data Science for Business” von Foster Provost und Tom Fawcett gibt einen guten nicht-technischen Überblick über die grundlegenden Prinzipien von Data-Science [2]. Das Buch „Artificial Intelligence A Modern Approach” von Stuart Russell und Peter Norvig ist ein Standardwerk zum Thema Künstliche Intelligenz [4]. Das Buch „Networks An Introduction” von M.E.J. Newman deckt das Thema der Netzwerkanalyse umfassend ab [5].

Der Online-Kurs „Text Retrieval and Search Engines” von Zhai ChengXiang [6], die Online-Dokumentation diverser Software-Bibliotheken und diverse Internetquellen waren Gegenstände der Recherche.

Für das Verständnis dieser Arbeit werden der Gerichtsentscheid, die Informations-Rückgewinnung, die Netzwerkanalyse, Word Embedding und N-grams genauer beschrieben bzw. definiert. Weitere spezifische Technologien werden in dieser Bachelorarbeit jeweils in den einzelnen Kapiteln eingeführt.

2.1 Gerichtsentscheid

Die zur Verfügung gestellten Daten beschränken sich auf öffentliche Gerichtsentscheide, Gesetzesartikel und Literatur. Der Gerichtsentscheid wird zum Trainieren der Algorithmen verwendet und ist Gegenstand der Analyse und der User Stories. Ein Beispiel für einen Gerichtsentscheid ist im Anhang ersichtlich. Alternativ können Gerichtsentscheide auf der öffentlichen Seite Lawsearch.ch eingesehen werden.

Der Gerichtsentscheid ist ein Entscheid des Gerichts über einen Streitgegenstand. Der Gerichtsentscheid ist jeweils in Urteilskopf, Regeste, Sachverhalt, Erwägungen, Dispositiv und Schlussformel unterteilt. In den zur Verfügung gestellten Daten gibt es drei verschiedene Gerichtsentscheide und dazugehörige Gerichte: Bundesverwaltungsgericht, Bundesstrafgericht und Bundesgericht.

Das Bundesverwaltungsgericht entscheidet über Beschwerden gegenüber der Verfügung von Bundesbehörden [7]. Wenn das Bundesverwaltungsgericht (BVGer) nicht als letzte Instanz entscheidet, dann können seine Urteile vor dem Bundesgericht angefochten werden. Das Bundesgericht (BGer) ist das oberste Gericht der Schweiz und gehört zu den drei Staatsgewalten des politischen Systems [8]. Das Bundesstrafgericht (BStGer) entscheidet über Beschwerden, die dem Gesetz der Gerichtsbarkeit der Schweiz unterstellt sind, zum Beispiel Beschwerden gegenüber Verfahrenshandlungen der Polizei [9]. Im Anhang 12.2 wird ein Gerichtsentscheid als Beispiel zu finden sein.

2.1.1 Urteilskopf und Regeste

Urteilskopf und Regeste enthalten eine Angabe der entscheidenden Instanz. Dann wird das Entscheid-Datum angegeben, wann der Entscheid erfolgt ist. Es folgt die Angabe der Abteilung und die Auflistung der Teilnehmer. Die Besetzung setzt sich zusammen aus Richter und Gerichtsschreiber. Die Verfahrensbeteiligten sind der Kläger und der Beklagte, welche durch ihre Rechtsanwälte vertreten sind. Zudem wird in diesem Abschnitt der Streitgegenstand beschrieben.

2.1.2 Sachverhalt

Der Sachverhalt beschreibt im ersten Schritt die unbestrittenen Fakten zum Rechtsfall. Danach werden die Tatsachenbehauptungen des Beklagten und des Klägers aufgelistet. Dann wird der komplette Verlauf des Prozesses beschrieben mit den möglichen Beweisaufnahmen. Auch beinhaltet der Sachverhalt die Prozessgeschichte wie Schriftenwechsel oder Details zur Hauptverhandlung. Das Rechtsbegehren der Parteien wird erläutert.

Das Ziel einer Rechtsschrift ist, das tatsächliche Geschehen aus der Sicht der einzelnen Parteien darzustellen. Dabei werden die rechtlichen Schlussfolgerungen beschrieben, welche für den Rechtsfall ausschlaggebend sind und aus dem Tatbestand erfolgen.

2.1.3 Erwägungen

Die Erwägungen beschreiben das Geschehen und die Überlegungen des Gerichts. Die Prozessvoraussetzungen, wie die Zuständigkeit des Gerichts, müssen erfüllt werden. Es muss festgehalten werden, dass die Voraussetzungen erfüllt sind. Es wird jeweils zuerst von den behaupteten und rechtlich relevanten Tatsachen der einzelnen Parteien ausgegangen. Die

von den Parteien aufgeführten Beweismittel werden analysiert und relevante Rechtssätze hierzu eruiert. Wichtig ist es, sich, zum einen, ausgiebig mit den einzelnen rechtlichen Standpunkten der Parteien auseinanderzusetzen und, zum anderen, diverse Rechtsprechungen und Literaturen zu begutachten, um sich eine Meinung zu bilden und sich dann zu entscheiden. In einzelnen Fällen kann es vorkommen, dass die Parteien nicht die nötigen Aussagen vorbringen. Dann können auch andere Argumentationen hervorgebracht werden, die überzeugender sind. Am Schluss im Rahmen der Erwägungen wird das Ergebnis zusammengefasst. Der Entscheid wird beschrieben und es wird erklärt, wer die Kosten für die Verhandlung tragen muss.

2.1.4 Dispositiv

Das Dispositiv fasst den Entscheid noch einmal zusammen in vier nummerierten Elementen. Zum einen wird der Sachentscheid beschrieben, ob die Beschwerde gutgeheissen oder abgelehnt wird. Der Kostenentscheid wird noch einmal erwähnt. Die Rechtsmittelbelehrung wird aufgeführt und der Entscheid über die Art der Bekanntmachung wird dargestellt.

2.1.5 Schlussformel

In der Schlussformel werden die Teilnehmer mit ihren Unterschriften aufgelistet.

Um auf die von der Firma Weblaw zur Verfügung gestellten Gesetzesartikel effizient zuzugreifen, sollten diese für eine schnelle Abfrage indexiert werden. Auch sollte eine Suche implementiert werden, mit der Datenabfragen möglich sind. Information Retrieval gibt eine Einführung in dieses Thema.

2.2 Information Retrieval

Bei der Information Retrieval geht es darum, bestehende Informationen, welche in einer grossen Datenbank gespeichert sind, abrufbar zu machen. Auch möchte man aus den zur Verfügung gestellten Daten nützliches Wissen extrahieren. Systeme, die entworfen wurden, um Informationen auf Computersystemen zu lokalisieren, werden Suchmaschinen genannt. Das Ziel einer Suchmaschine ist es, so schnell wie möglich Informationen aus einer Datenquelle zu extrahieren und eine Schnittstelle bereitzustellen. Der Benutzer einer Suchmaschine kann eine Abfrage machen und die Suchmaschine versucht dann, die relevantesten Ergebnisse in Bezug auf die Abfrage zu finden.

2.2.1 Apache Lucene

Apache Lucene ist eine Open Source Information-Retrieval-Bibliothek, welche von der Firma Weblaw verwendet wird. In der Bachelorarbeit wird Elasticsearch verwendet, welches auf Apache Lucene aufgebaut ist. Die Lucene-Bibliothek wurde in Java programmiert und bietet eine Volltextindexierung und eine Textsuche. Sie ist sehr performant und wird unter anderem für Wikipedia verwendet. Auch werden boolesche Operatoren, Wildcard-, Feld- und Bereichs-Suchvorgänge unterstützt. Apache Lucene hat eine sehr gute Dokumentation, Unterstützung für allgemeine Suchmaschinen-Funktionen und Skalierbarkeit [10].

Die Suchanfragen, die von einem Benutzer eingegeben werden, werden vom Lucene-Query-Parser verarbeitet. Der Query-Parser verarbeitet und interpretiert die Eingabe der Benutzer, sodass diese von Lucene verstanden wird. Die Abfragen sind auf zwei Hauptkomponenten aufgebaut, nämlich auf Begriffe und Operatoren. Die Begriffe können auch Attribute enthalten. Der Benutzer kann nach Attributen, zum Beispiel dem Autor, suchen [10].

Apache Lucene speichert alle benötigten Informationen, auf denen es sucht, in Dateien, die Indizes genannt werden. Diese Indizes enthalten mehrere Dokumente und diese wiederum Felder. Indizes speichern Term-Vektoren, um die Sucheffizienz zu steigern. Apache-Lucene-Indizes können in Segmente unterteilt werden, wobei jedes Segment als Index fungiert. Es kann jeweils nach einem oder mehreren Segmenten gesucht werden [10].

2.2.2 Thesaurus

Der Thesaurus ist eine Baumstruktur an Wörtern, welche ein Themengebiet möglichst gut beschreibt. Lawsearch verwendet den Thesaurus Jurivoc. Jurivoc ist ein mehrsprachiger Thesaurus des Schweizerischen Bundesgerichts.

Beim Thesaurus handelt es sich um ein Vokabular, worin die Wörter durch Relationen miteinander verbunden sind. Jurivoc verwaltet Ober- und Unterbegriffe und Synonyme.

Sobald die Daten indexiert und suchbar sind, können effiziente Abfragen auf den Daten gemacht werden. Zum Beispiel kann nun nach Gerichtsentscheiden gesucht werden. Auch können Gerichtsentscheide gesucht werden, welche innerhalb der Gerichtsentscheide erwähnt

werden. Diese Gerichtsentscheide und ihre Verknüpfungen untereinander bilden ein Netzwerk. Dieses Netzwerk kann mithilfe der Netzwerkanalyse analysiert werden.

2.3 Netzwerkanalyse

Ein Netzwerk ist eine Sammlung von Knoten bzw. einzelnen Dokumenten, die jeweils mit Kanten, Verbindungen untereinander, verbunden sind. In dieser Bachelorarbeit wird das Netzwerk zwischen Gerichtsentscheiden aufgebaut. Die einzelnen Gerichtsentscheide sind die Knoten. Die Referenzen auf andere Gerichtsentscheide innerhalb eines Gerichtsentscheids bilden die Kanten.

Das Ziel der Netzwerkanalyse ist es, eine Struktur in einer grossen Menge an Daten zu finden. Das Netzwerk kann mithilfe der Graphenanalyse analysiert werden. Dort werden die Beziehungen zwischen den Knoten analysiert. Es wird untersucht, ob es sich um einen gerichteten oder ungerichteten Graphen handelt. Viele Themen teilen sich die Netzwerkanalyse und die Graphenanalyse [5].

2.3.1 Gerichteter Graph

Verlinkungen auf andere Gerichtsentscheide gehen jeweils von einem Gerichtsentscheid aus. Das heisst, zwischen zwei Knoten sind die Kanten gerichtet; sie werden durch Pfeile symbolisiert. Es handelt sich dabei um einen gerichteten Graph. Der Vorteil des gerichteten Graphen liegt darin, dass an den Kanten jeweils die Richtung zu erkennen ist und somit deutlich wird, welcher Gerichtsentscheid welchen anderen Gerichtsentscheid verlinkt.

2.3.2 Multigraph

Bei einem Multigraphen handelt es sich um einen Graphen, welcher zwischen Knoten mehrere Verlinkungen aufweisen kann. Das heisst, ein Gerichtsentscheid kann im Text auf mehrere gleiche Gerichtsentscheide verlinken. Auch kann es sein, dass ein Gerichtsentscheid im Text auf sich selber verlinkt. Es kann der Fall sein, dass sich zwei Gerichtsentscheide gegenseitig verlinken [11].

2.3.3 Gewichtete Kanten

Es ist möglich, den Kanten jeweils weitere Attribute hinzuzufügen und somit eine Gewichtung zu geben. Zum Beispiel kann die Entfernung zwischen zwei Knoten angegeben werden. Zudem kann, wie in unserem Beispiel, die Ähnlichkeit von zwei Gerichtsentscheiden jeweils mithilfe von Word Embedding und Kosinus-Ähnlichkeit angegeben werden. Die gewichtete Kantenzahl kann genutzt werden, um einen Gerichtsentscheid mit einer hohen Ähnlichkeitszahl zu priorisieren.

2.3.4 Nachbarschaft

Von einem Ausgangsknoten aus sind alle mit Kanten verbundenen Knoten Nachbarn. Knoten, die mit Kanten auf den Ausgangsknoten zeigen, sind jeweils Vorgänger und Knoten, auf welche der Ausgangsknoten zeigt, sind Nachfolger. Für diese Bachelorarbeit wurde noch eine weitere Variante als Nachbar definiert. Knoten, welche jeweils über zwei Nachbarn mit einem anderen Knoten verbunden sind, werden ebenfalls als Nachbarn bezeichnet. Das wird jeweils in beide Richtungen berücksichtigt. In dieser Bachelorarbeit nennen wir diese Knoten „across-neighbors-Knoten".

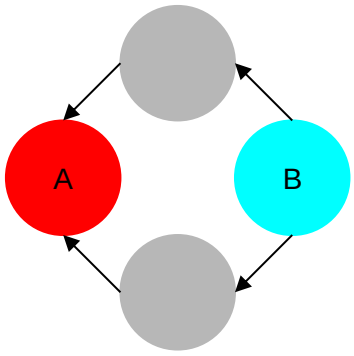


Abbildung 1 "across-neighbors-Knoten" Konstellation 1

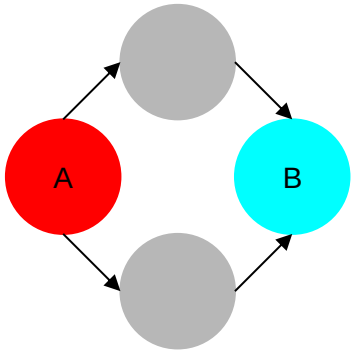


Abbildung 2 "across-neighbors-Knoten" Konstellation 2

2.3.5 PageRank

Der PageRank-Algorithmus ist ein mathematisches Verfahren, um die Popularität eines Knotens im Graphen zu berechnen. Dieses Verfahren wurde von Google in der Vergangenheit verwendet, um die Popularität einer Website zu berechnen und um die Websites nach Popularität in den Suchergebnissen aufzulisten.

Bei den Gerichtsentscheiden untersucht der PageRank-Algorithmus, wie viele Referenzen es auf einem gegebenen Gerichtsentscheid gibt. Neben der Anzahl der Referenzen ist auch das Gewicht des gegebenen Gerichtsentscheides ausschlaggebend, von dem aus auf weitere Gerichtsentscheide referenziert wird [3] [12].

2.4 Word Embedding

Word Embedding ist ein Oberbegriff für eine Anzahl an Natural-Language-Processing-Techniken. Es wird versucht, Wörter und Wortgruppen als Vektoren darzustellen, um auf ihnen mathematische Operationen durchführen zu können [13]. Wörter mit der gleichen Bedeutung sollten die gleiche Repräsentation im Vektorraum haben. Jedes Wort wird als ein Vektor dargestellt.

2.4.1 Wörter-Repräsentation

Bei der Wörter-Repräsentation geht es darum, Wörter als Vektoren darzustellen. Je nachdem, in welchem Zusammenhang einzelne Wörter gebraucht werden, können für das gleiche Wort unterschiedliche Vektoren entstehen. Deshalb beinhalten die einzelnen Vektoren Informationen zu Sprache, Wortanalogien und Semantik. Um sinnvolle Vektoren zu den einzelnen Wörtern zu erstellen, braucht es einen grossen Korpus an Texten, und je nachdem, welche Dokumente im Korpus sind, bekommen die Vektoren andere Informationen. Dieses Verfahren wird in dieser Bachelorarbeit für fastText verwendet. fastText ist eine Library für Textklassifikation das Erlernen von Word Embedding [14]. Sobald die Wörter als Vektoren dargestellt sind, können mathematische Verfahren auf den Vektoren ausgeführt werden, zum Beispiel kann die Kosinus-Ähnlichkeit zwischen zwei Vektoren berechnet werden.

2.4.2 Kosinus-Ähnlichkeit

Die Kosinus-Ähnlichkeit ist ein Mass, welches die Ähnlichkeit zwischen zwei Vektoren definiert. Die Ähnlichkeit wird bestimmt, indem der Kosinus des Winkels zwischen den beiden Vektoren berechnet wird. Je kleiner der Winkel zwischen zwei Vektoren ist, desto ähnlicher sind die Vektoren [15]. Da dieses Mass für diese Arbeit von zentraler Bedeutung ist, wird es hier zusätzlich mathematisch erklärt.

$$\text{Kosinus-Ähnlichkeit} = \cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2} = \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} (a_i)^2} \cdot \sqrt{\sum_{i=1}^{n} (b_i)^2}}$$

Abbildung 3 Formel für Kosinus-Ähnlichkeit [15]

a und b sind Vektoren. Das Skalarprodukt beider Vektoren geteilt durch das Produkt von a in Betrag und b in Betrag ergibt den Kosinus.

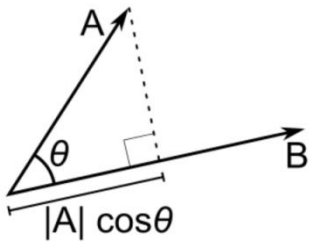


Abbildung 4 Illustration Abstand zwischen zwei Vektoren [16]

Die Kosinus-Ähnlichkeit kann jeweils einen Wert von -1 bis 1 annehmen. Da die Semantik eines Wort-Vektors oder Text-Vektors nicht negativ dargestellt werden kann, beschränkt sich der Wertebereich auf 0 bis 1. 0 bedeutet, dass zwei Wörter oder zwei Texte gar nicht ähnlich sind. Ein Wert von 1 bedeutet, dass zwei Wörter oder zwei Texte die gleiche Bedeutung haben.

2.4.3 N-Gramm

Zwei Wörter werden jeweils mithilfe von Word Embedding und Kosinus auf ihre Ähnlichkeit geprüft. Was passiert aber mit unbekannten Wörtern, welche nicht mithilfe von Wörter-

Repräsentation gelernt wurden? Um trotzdem ein Ergebnis zu berechnen, können N-Gramms verwendet werden. Dabei werden Wörter aufgeteilt in Teilwörter. Zum Beispiel wird das Wort Bachelorarbeit in Bachelor und Arbeit aufgeteilt. Wenn es Vektoren für Bachelor und Arbeit gibt, dann wird die Ähnlichkeit zu diesen berechnet [16].

In dieser Bachelorarbeit wird für das Trainieren von Word Embedding fastTest verwendet.

2.4.4 fastText

fastText ist eine vom Facebook-Forschungsteam erstellte Bibliothek für das effiziente Erlernen von Word Embedding und Text-Klassifizierungen. In dieser Arbeit wurde diese Library ausschliesslich für das Erlernen von Word Embedding aus Gerichtsentscheiden verwendet.
Beim Erlernen der Word Embedding wird jedes Wort von fastText als Zusammensetzung aus N-Gramms behandelt. fastText berechnet für jedes N-Gramm einen Vektor und am Ende werden alle N-Gramm-Vektoren des Wortes zusammengesetzt, welcher der Vektor des Wortes ist und das Wort in semantischer Form repräsentiert [14].

2.4.5 Skipgram vs. CBOW

fastText bietet zwei verschiedene Modell-Architekturen an: Einerseits das „Skipgram" und andererseits das „CBOW".
Die Skipgram-Architektur lernt, durch ein naheliegendes Wort im Text ein Zielwort vorherzusagen. Die CBOW-Architektur lernt, anhand vom Kontext das Zielwort vorherzusagen. Der Kontext wird als „Bag of words" dargestellt, die in einem Fenster mit fester Größe um das Zielwort herum enthalten sind [14]. In Abbildung 16 werden die beiden Architekturen illustriert.

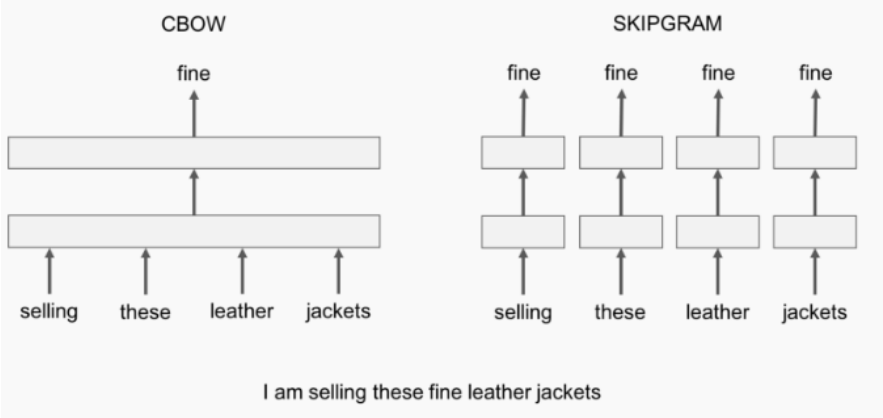


Abbildung 5 Skipgram und CBOW [14]

Dieses Verfahren dient als Grundlage für die Library „fastText". Um die Ähnlichkeit zwischen zwei Texten zu berechnen gibt es noch ein zweites Verfahren, das TF-IDF-Verfahren.

2.5 TF-IDF

TF-IDF steht für *term-frequency* (TF) und *inverse-document-frequency* (IDF). In Deutsch steht TF für „Vorkommenshäufigkeit" und IDF für „Inverse Dokumenthäufigkeit". TF-IDF ist ein Mass für die Relevanz eines Wortes in einem Dokument eines Korpus [17]. TF gibt an, wie häufig ein Wort in einem Dokument vorkommt. IDF senkt das Gewicht von Wörtern, die häufig im Korpus vorkommen, und erhöht das Gewicht von Wörtern, die im Korpus selten erscheinen. Die Relevanz eines Wortes steigt also proportional mit der Anzahl der Auftritte eines Wortes (TF) im Dokument oder im Korpus, wird aber durch die Häufigkeit des Wortes im Korpus ausgeglichen (IDF) [18]. Die Multiplikation von TF-Zahl und IDF-Zahl ergibt die Zahl, welche aussagt, wie relevant das Wort ist.
Das Mass wird nun in einem Beispiel illustriert. Man nehme an, dass ein (nicht-existent) ZHAW-Jahresbericht der Korpus ist und dass der Beispielsatz in einem Kapitel des Jahresberichts vorkommt, in dem über die Frackwoche berichtet wird.

Beispielsatz: „Die Frackwoche war hervorragend."

Wörter wie „*Die*" oder „*war*", welche in deutschen Texten üblicherweise häufig vorkommen und welche man in Natural-Language-Processing als Stoppwörter bezeichnet, werden hohe TF-Zahlen erhalten. Aber wegen ihrer Häufigkeit im gesamten Text(Korpus) werden sie tiefere IDF-Zahlen haben. Das Wort „*hervorragend*" wird vermutlich eine höhere TF-IDF-Zahl haben, da es in deutschen Texten weniger vorkommt als die Wörter „*die"* und „*war"*. Das Wort „Frackwoche" ist ein seltenes Wort im ZHAW-Jahresbericht, kommt aber dennoch einige Male vor, da die Frackwoche, über das gesamte Jahr gesehen, in der ZHAW als eines der grössten Geschehnisse betrachtet wird. Dementsprechend wird das Wort „*Frackwoche"* in diesem Satz die höchste TF-IDF-Zahl haben.
Dieses Verfahren wird in der Suchmaschine Elasticsearch verwendet. Diese Suchmaschine ist für diese Bachelorarbeit wesentlich und wird im Kapitel „Datenvorbereitung" erklärt. Es gibt einen wesentlichen Unterschied zwischen TF-IDF und Word Embedding, welcher im nächsten Kapitel erläutert wird.

2.6 TF-IDF vs. Word Embedding

Der Unterschied zwischen TF-IDF und Word Embedding ist, dass Word Embedding jedes Wort gleichbehandelt. TF-IDF gewichtet jedes Wort anders, je nachdem, wie oft das Wort im Dokument und im gesamten Dokument-Korpus vorkommt. Der Vorteil von Word Embedding ist, dass dieses erlaubt, den gesamten Kontext zu analysieren, indem es jedes Wort und seine Nachbarwörter in Betracht zieht. TF-IDF filtert anhand der Gewichtung bei den einzelnen Wörtern gewissermassen die Stoppwörter heraus und daher kann der Kontext weniger gut erfasst werden. TF-IDF ist hingegen gut geeignet, um das Topik oder das Schlagwort eines Textes zu bestimmen [19].

Sobald man die Ähnlichkeit zwischen dem Ausgangsdokument und den anderen Dokumenten im Korpus berechnet hat, kann diese in einer Rangordnung aufgelistet und nach ihrer Ähnlichkeit sortiert werden.

2.7 Rangordnung

Eine Rangordnung ist eine Reihenfolge von Objekten. Die Sortierung zwischen den Objekten spielt eine grosse Rolle und legt eine Bewertung fest. In dieser Bachelorarbeit sind die Objekte einzelne Gerichtsentscheide. Der Gerichtsentscheid auf dem ersten Platz ist der Gerichtsentscheid, dessen Sachverhalt am Ähnlichsten zum Sachverhalt des Ausgangsdokuments ist. Die Rangordnung wird jeweils aus mehreren Kriterien gebildet, diese werden zu einem Gesamtwert zusammengerechnet. In dieser Bachelorarbeit wurden jeweils mehrere Ranglisten erstellt. Auch wurden von der Firma Weblaw AG mehrere Ranglisten mit den gewünschten Resultaten erstellt. Die generierten Ranglisten sollen nun mit den manuell erstellten Ranglisten verglichen werden. Für den Vergleich wird das Rangkorrelationsverfahren verwendet, welches nun näher beschrieben wird.

2.7.1 Rangkorrelation

Eines der meist verwendeten Verfahren, um die Korrelation zwischen Objekten in einer Rangliste zu messen, ist das Kendall-Tau-Verfahren. Dieses Verfahren misst die Wahrscheinlichkeit, dass zwei Objekte in zwei unterschiedlichen Ranglisten in der gleichen Reihenfolge sind [20].

$$\tau = P(C) - P(D) = \frac{C}{N} - \frac{D}{N} = \frac{C - D}{N}$$

Dies ist die Formel für die Berechnung des Kendall-Tau-Scores, welche nun erläutert wird. N ist die Gesamtzahl der Paare. Als Paare werden zwei Objekte in einer Rangliste bezeichnet. Unten wird N mathematisch beschrieben.

$$N = \frac{1}{2} n(n-1)$$

n ist die Anzahl der Objekte in einer Rangliste.

C ist die Anzahl der übereinstimmenden Paare. Die Anzahl der überstimmenden Paare ist die Anzahl der Paare, für die die Reihenfolge in den beiden Listen beibehalten wird.
Die Wahrscheinlichkeit, dass es übereinstimmende Paare gibt, wird folgendermassen beschrieben.

$$P(C) = \frac{C}{N}$$

D ist die Anzahl der nicht übereinstimmenden Paare. Die Anzahl der nicht übereinstimmenden Paare ist die Anzahl der Paare, bei denen die Reihenfolge umgekehrt ist.
Die Wahrscheinlichkeit, dass es nicht übereinstimmende Paare gibt, wird folgendermassen beschrieben.

$$P(D) = \frac{D}{N}$$

Es müssen mehrere Aspekte beim Verwenden dieses Verfahrens beachtet werden. Es kann vorkommen, dass Objekte jeweils nur in einer Rangliste aufgelistet werden. Man muss Annahmen für fehlende Objekte treffen. Eine Variante ist es, anzunehmen, dass die fehlenden Objekte zuunterst in der Liste sind.
Ein weiteres Problem ist, dass die Reihenfolge der Objekte innerhalb einer Rangliste keine Auswirkung auf den Kendall-Tau-Score hat. Als Beispiel nehmen wir drei Listen an.

A: [a, b, c, d, e]

B: [b, a, c, d, e]

C: [a, b, c, e, d]

Die Elemente (Objekte) a und b sind in den Listen A und B vertauscht. In der Liste B und C sind die Elemente d und e vertauscht. Vergleicht man nun die Listen B und C zu Liste A, indem man den Kendall-Tau-Score für die Liste B und C anhand obiger Formeln berechnet, erzielen Liste B und C trotz Vertauschungen zweier Elementpaare den gleichen Kendall-Tau-Score, nämlich 0.8 [20].

Für diese Bachelorarbeit ist die Reihenfolge in der Rangliste von immenser Wichtigkeit. Was nun versucht wird, ist, die vorderen Objekte höher zu gewichten als die hinteren Objekte. Das kann mit einem Average-Overlap-Score erreicht werden. Eine Änderung weiter hinten in der Rangliste ergibt einen höheren Average-Overlap-Score als weiter vorn. Je höher der Score ist, desto ähnlicher sind die Ranglisten.

Der Average-Overlap-Score kann aber in grossen Ranglisten sehr gross werden und wird somit gar nicht mehr aussagekräftig. Um dieses Problem zu bewältigen, können geometrische Reihen verwendet werden, um den Score konvergieren zu lassen. Das Rank-Biased-Overlap-Verfahren basiert auf dem Average-Overlap-Score und verwendet geometrische Reihen, um den Score bei grossen Ranglisten konvergieren zu lassen [21].

Der Vergleich unterschiedlicher Ranglisten wird in der Evaluation dieser Bachelorarbeit angewendet, um die vom Algorithmus generierten Ranglisten mit den manuell von einem Menschen erstellten Ranglisten zu vergleichen und zu bewerten. Bevor die Implementierung der Algorithmen und die Auswertung der Resultate durchgeführt werden können, muss das Problem des Kunden verstanden und die Umgebung, wo die Daten abgespeichert werden, aufgesetzt werden.

3 Geschäftsverständnis

Beim Business-Understanding werden die Firma Weblaw und ihr Produkt „Lawsearch" genauer vorgestellt. Das Problem des Kunden wird genauer untersucht, um aus den Erkenntnissen funktionale und nichtfunktionale Anforderungen abzuleiten. Die Systemumgebung beschreibt das System im Betrieb und die Systemarchitektur beschreibt die Funktion der Applikation im System.

3.1 Weblaw AG

Die Weblaw AG bietet auf ihrer Webseite unterschiedliche Dienste rund um das Thema Recht an. Diese umfassen Online-Zeitschriften, E-Publikationen, Magisterarbeiten sowie Campuslizenzen. Zusätzlich bietet Weblaw juristische Fachliteratur und Beratung zum Thema Recht und Weiterbildungen an. Eine weitere Hauptfunktion der Firma Weblaw ist das Zurverfügungstellung einer Onlinesuche über Rechtsdokumente via Lawsearch.ch.

3.2 Lawsearch

Lawsearch gehört zur Produktpalette der Firma Weblaw. Lawsearch ist eine Suchmaschine. Sie verwendet für die Indexierung der Rechtsdokumente das Jurivoc und speichert die Daten anschliessend auf einem mit Apache Lucene aufgesetzten Server. Jurivoc ist ein Thesaurus bzw. ein Wörterbuch. Die Daten werden verschlagwortet, übersetzt, mit externen Referenzen vernetzt und nach relevanten Stichworten durchsuchbar gemacht.
Anhand einem Beispiel-Screenshot wird der Aufbau der Lawsearch Webseite im nächsten Kapitel 3.3 illustriert und beschrieben.

3.3 Aufbau der Lawsearch-Webseite

Der Aufbau der Lawsearch-Webseite ist ganz simpel gelöst. Abbildung 5 veranschaulicht das. Ganz oben bei der Nr. 1 ist das Suchfeld zu sehen, wo einzelne Suchwörter eingegeben werden können. In diesem Beispiel ist es das Wort „Auto". Zusätzlich kann man aus der linken Spalte Attribute wählen und diese in die untere Zeile Nr. 2 einfügen. Die Anzahl der Suchresultate wird dadurch verkleinert. Nun wird geschaut, wie oft „Auto" im Dokument erscheint. Je öfter es auftaucht, desto höher ist der Score und desto höher wird das Resultat in der Liste bei Nr. 3 aufgeführt.

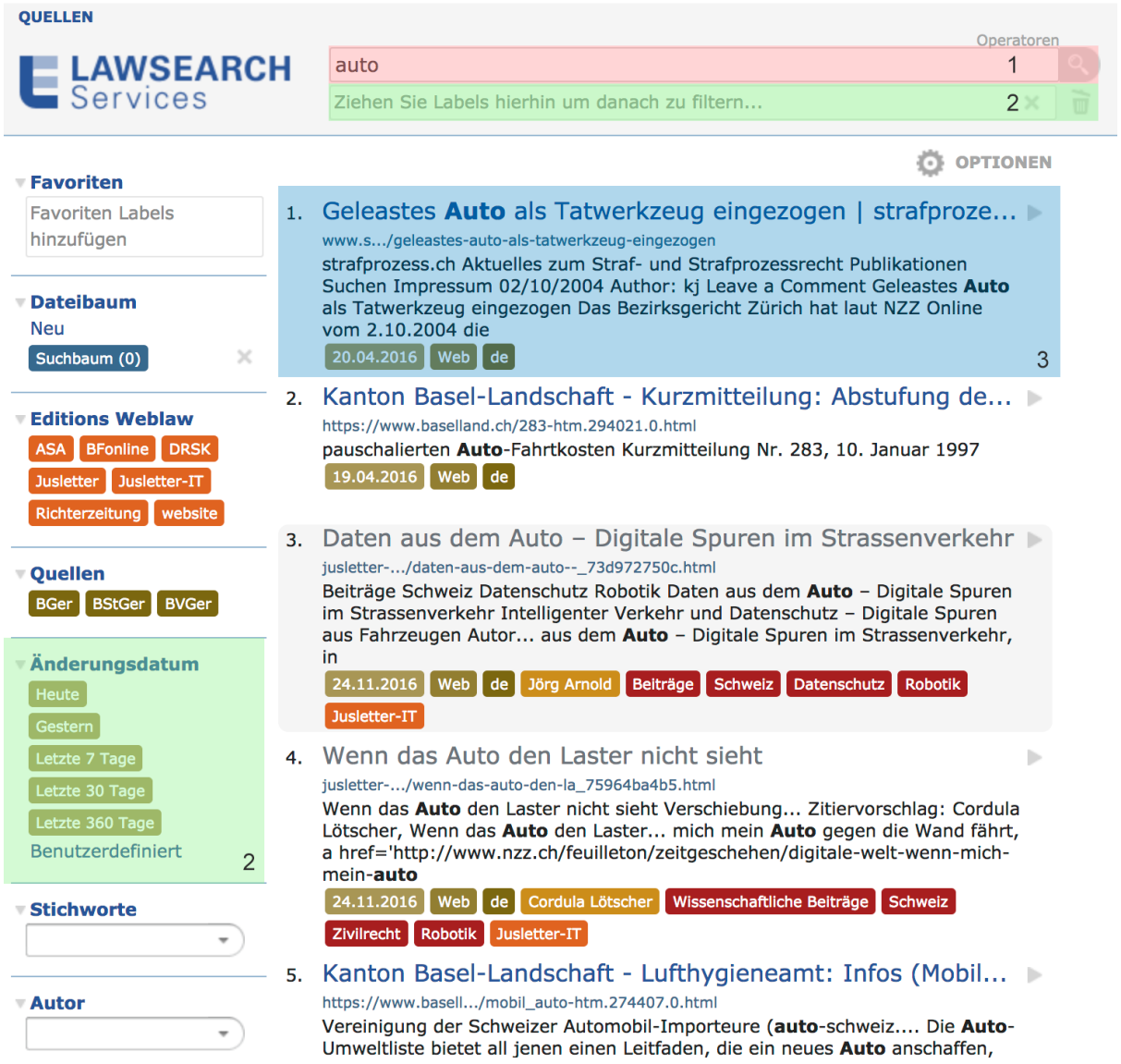


Abbildung 6 Lawsearch

Sobald man einen Gerichtsentscheid ausgewählt hat, kommt man auf die Detailansicht. Bei diesem Beispiel wurden die Suchwörter gelb markiert, Nr. 1 BGE. Dann folgt der Sachverhalt, welcher den Rechtsfall beschreibt. Auf der rechten Seite sind die Schlagwörter, Nr. 2, zu sehen. Diese Wörter wurden mithilfe des Thesaurus aus dem Text extrahiert. Es konnten auch die einzelnen Gerichtsentscheide erkannt und aufgelistet werden, Nr. 3. Anhand dieser wird dann später im Kapitel Netzwerkanalyse das Netzwerk gebildet.

Abbildung 7 Gerichtsentscheid

3.4 Ziele des Kunden

Das Ziel von Weblaw ist es, gewünschte Dokumente schneller und präziser von Lawsearch finden zu lassen. Was es bisher in Lawsearch nicht gab, ist ein Vergleich zwischen zwei Dokumenten auf ihre Ähnlichkeit. Das heisst, wenn man jeweils ein Dokument betrachtet, kann man nicht von diesem Dokument auf das nächste ähnliche Dokument springen. Auch werden bei der Indexierung der einzelnen Wörter nur Wörter innerhalb eines Dokuments betrachtet.

In der aktuellen Implementierung kann man nur mithilfe von einzelnen Wörtern nach Dokumenten suchen und nicht mithilfe eines ganzen Textes.
Weblaw möchte eine Erweiterung von Lawsearch implementieren, die es erlaubt, ausgehend von einem Text möglichst ähnliche Texte zu finden. Diese Texte werden nach ihrer Relevanz sortiert und angezeigt. Dadurch entsteht für den Benutzer von Lawsearch eine neue Benutzererfahrung. Der Benutzer kann zum Beispiel einen Sachverhalt selber verfassen und, mithilfe der Suche, nach Gerichtsentscheiden mit ähnlichen Sachverhalten suchen. Auch kann der Benutzer ausgehend von einem Text direkt zum nächsten relevanten Text springen.

Als Beispiele können Google und YouTube betrachtet werden. Bei Google kann, wie bei Lawsearch, ein Dokument aus einer Liste ausgewählt und angeschaut werden. Danach muss dann aber wieder zurück auf die Suche gesprungen werden, um ein neues Dokument auszuwählen. Als zusätzliches Feature für Lawsearch wird in dieser Bachelorarbeit die YouTube-Funktionalität betrachtet. Wenn man bei YouTube ein Video anschaut, werden auf der rechten Seite weitere dazu relevante Videos angezeigt. In dieser Bachelorarbeit geht es darum, eine solche Funktion für Lawsearch und für Rechtsdokumente zu implementieren.

In dieser Bachelorarbeit wird diese neue Funktionalität theoretisch erarbeitet und ein Prototyp dafür erstellt. Für diese erweiterte Suchfunktionalität können funktionale und nichtfunktionale Anforderungen an die Implementierung abgeleitet werden.

3.5 Abgeleitete Anforderungen

Aus den Zielen des Kunden Weblaw für die neue Funktionalität, die erweiterte Suche und die Probleme mit der aktuellen Implementierung wurden Anforderungen abgeleitet, gemäss Requirements Engineering, wobei sich die Anforderungen auf deutsche Gerichtsentscheide beschränken (mehr dazu im Kapitel 4 „Daten-Verständnis“).

- Wenn ein Gerichtsentscheid als Ausgangsdokument ausgewählt wird, sollten die ähnlichsten Gerichtsentscheide angezeigt werden.
- Wenn ein neuer Sachverhalt als Ausgangsdokument ausgewählt wird, sollten Gerichtsentscheide mit den ähnlichsten Sachverhalten angezeigt werden.
- Es sollten jeweils die fünf besten Resultate angezeigt werden.

- Die Resultate sollten nach einer Rangordnung sortiert werden. Zuoberst werden die ähnlichsten Dokumente genannt.

Alle Anforderungen, welche nicht direkt mit einer Funktionalität in Verbindung stehen, werden als nicht-funktionale Anforderung erfasst.

3.6 Nicht-Funktionale Anforderungen

Die nicht-funktionalen Anforderungen wurden aus dem ISO 25010 Standard abgeleitet [22].
Funktionale Software:

- Alle Software-Funktionen sollten implementiert sein, um die Anforderungen zu erfüllen.
- Die Funktionen sollten korrekt sein und das bestmögliche Resultat sollte angezeigt werden.

Effiziente Performanz:

- Bei bekannten Gerichtsentscheiden sollte die Anzeige des Resultats nie länger als einige Sekunden betragen.
- Die Suche mit neu verfassten Texten kann lange dauern.

3.7 Risiken und Massnahmen

Hier werden die Risiken, an welcher die Bachelorarbeit scheitern könnte, beschrieben. Auch werden die Massnahmen dargestellt, die angewendet werden können, damit es nicht zum Scheitern kommt.

- Die Zeit für die Implementierung ist stark begrenzt. Je nach der Anzahl der ausgewerteten Dokumente und dem Zustand der Dokumente kann die Entwicklungszeit stark variieren.
- Wegen begrenztem Vorwissen und begrenzten allgemeinen Kenntnissen in diesem Fachgebiet kann es sein, dass nicht die optimalste Lösung implementiert wird.
- Eine gute Implementation ist auf die Zusammenarbeit mit Weblaw angewiesen. Es müssen Rechtsdokumente zur Verfügung gestellt werden. Auch müssen die Resultate durch Weblaw bewertet werden.

3.8 Systemumgebung für den Prototyp

Für den Prototyp wurde eine virtuelle Maschine verwendet, auf welcher das Elasticsearch und ein Python-Environment installiert wurden.
Elasticsearch ist, wie in Kapitel 2.2.1 erwähnt, über Lucene aufgebaut und bietet eine JSON-basierte REST-API, um auf Lucene-Features zu verweisen. Diese JSON-basierte REST-API wird vom Python-Environment verwendet, um Abfragen über die Gerichtsentscheide zu senden und um Suchergebnisse im JSON-Format zu erhalten. Die Suchergebnisse werden danach im Python-Environment verarbeitet und in den Algorithmen verwertet. Näheres zur Verarbeitung der Suchergebnisse und zu den Algorithmen wird im Kapitel 7 „Umsetzung" beschrieben.
Zudem wurde fastText installiert, um die Ähnlichkeit zwischen Gerichtsentscheiden zu berechnen. Weitere Informationen zu fastText sind in Kapitel 6 „Modellierung" zu finden.

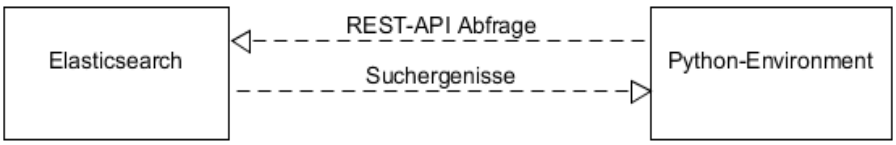


Abbildung 8 Systemumgebung für den Prototyp

3.9 User Stories

Eine User Story ist eine Anforderung an das System. In dieser Bachelorarbeit gibt es zwei Epic User Stories, da das Programm nur genau zwei Funktionen zur Verfügung stellt. Beide Funktionen generieren eine Rangliste von zehn Gerichtsentscheiden sortiert nach ihrer Ähnlichkeit. Bei der ersten User Story wird ein bestehender Gerichtsentscheid als Input genommen und bei der zweiten User Story wird ein neu verfasster Gerichtsentscheid als Input verwendet.

3.9.1 US 1: Ähnliche Gerichtsentscheide mit Netzwerkanalyse

- Als System möchte man dem Betrachter eines Gerichtsentscheids eine Rangliste mit ähnlichen Gerichtsentscheiden präsentieren, damit der Betrachter direkt einen ähnlichen Gerichtsentscheid auswählen kann.

3.9.2 US 2: Ähnliche Gerichtsentscheide mit Word Embedding

- Als System möchte man dem Verfasser eines neuen Sachverhalts eine Rangliste mit Gerichtsentscheiden präsentieren, welche ähnliche Sachverhalte aufweisen, damit der Verfasser direkt einen ähnlichen Gerichtsentscheid auswählen kann.

Die Sequenzdiagramme sollen diese beiden User Stories noch weiter verdeutlichen.

3.10 Sequenzdiagramm

Ein Sequenzdiagramm zeigt eine Interaktion des Systems mit den unterschiedlichen Software-Komponenten. Das erste Sequenzdiagramm US1 zeigt die Interaktion der ersten User Story auf: Ähnliche Gerichtsentscheide mit Netzwerkanalyse.

Der Benutzer führt eine Suche durch und wählt einen Gerichtsentscheid. Sobald der Inhalt des Gerichtsentscheids angezeigt wird, macht das System einen Aufruf an die Scoring-Komponente und übergibt den Titel des aktuellen Gerichtsentscheids.

Vorrangig wurde bereits ein Netzwerk über alle Gerichtsentscheide gebildet. Nun kann die Scoring-Komponente mit einem Aufruf an NetworkX alle Nachbarknoten und deren Kanten holen. Jetzt kann man mit einem Scoring-Algorithmus, zum Beispiel PageRank, die einzelnen Knoten bewerten. Schliesslich wird eine Rangliste erstellt, in der ersichtlich ist, welche Nachbarknoten am relevantesten sind.

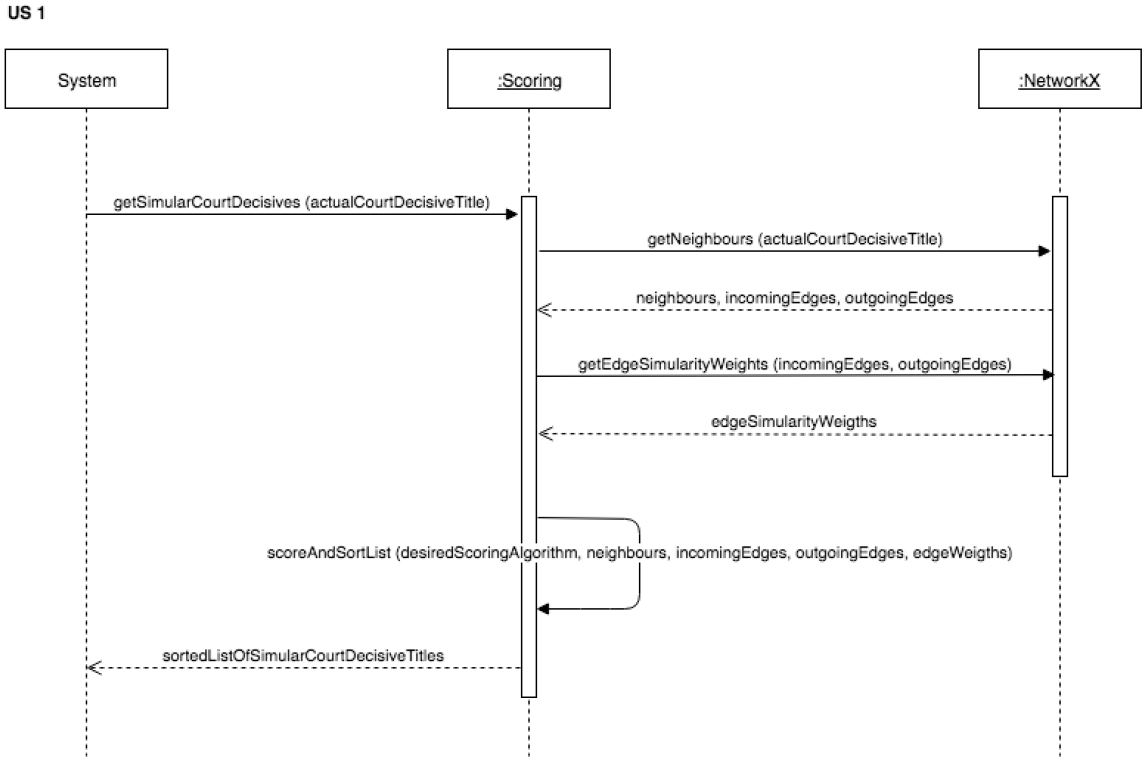


Abbildung 9 Sequenzdiagramm US 1

Das zweite Sequenzdiagramm US2 zeigt die Interaktion der zweiten User Story auf: Ähnliche Gerichtsentscheide mit Word Embedding. Vorrangig wurde das Model in fastText jeweils mit allen Gerichtsentscheiden trainiert.

Der Benutzer erstellt einen Sachverhalt und gibt diesen als Input ein. Das System macht einen Aufruf an die Scoring-Komponente und übergibt den Text. Die Scoring-Komponente macht einen Aufruf an das fastText-Modul. Dieses lädt sein Model und erstellt für den neuen Sachverhalt einen Vektor. Sobald die Vektoren der bereits verfügbaren Sachverhalte und der Vektor des neuen Sachverhalts vorliegen, kann die Kosinus-Ähnlichkeit zwischen dem neuen Text und den anderen Sachverhalten berechnet werden. Nach der Berechnung der Ähnlichkeiten zwischen dem neuen Sachverhalt und allen anderen Sachverhalten wird eine Rangliste erstellt, in der ersichtlich ist, welche die ähnlichsten Gerichtsentscheide sind.

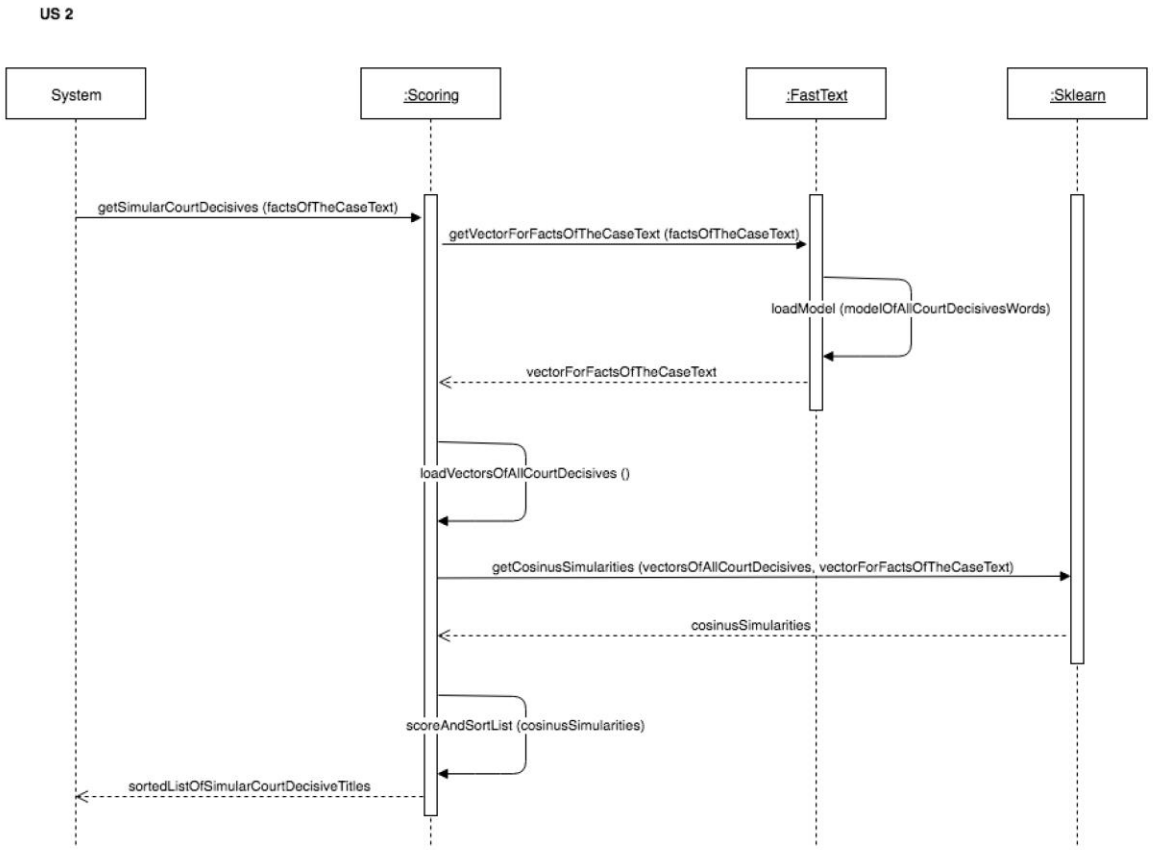


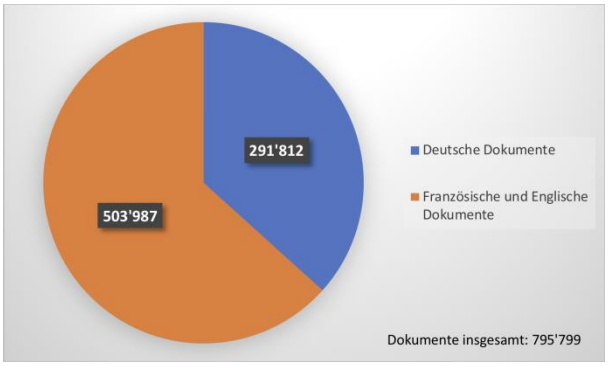
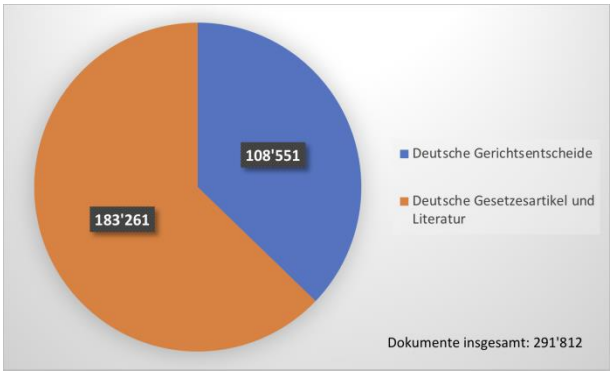
Abbildung 10 Sequenzdiagramm US 2

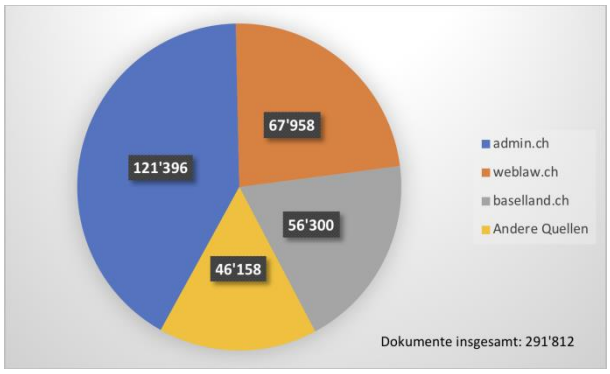
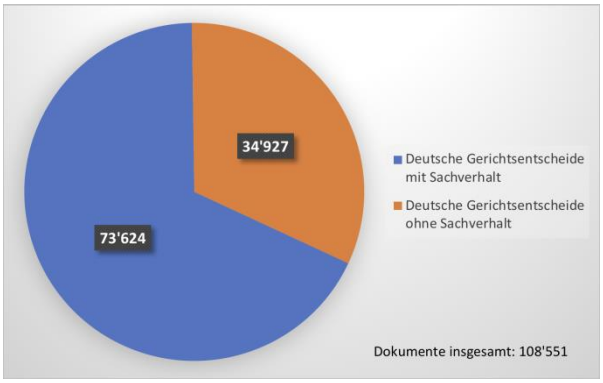
4 Daten-Verständnis

Das Daten-Verständnis dient dazu, einen Überblick über die von der Firma Weblaw zur Verfügung gestellten Daten zu erhalten. Dabei wurden jeweils unterschiedliche Statistiken erstellt, um die Daten besser zu analysieren. Nach einer ausführlichen Analyse wurde eine Auswahl an Daten vorselektiert. Auch beziehen sich die Evaluation und die Resultate auf einen spezifischen Muster-Testfall. Jedes von der Firma Weblaw erstellte Dokument folgt einem bestimmten Aufbau, welcher in diesem Kapitel beschrieben wird.

4.1 Analyse und Vorselektion der Daten

Bei der Analyse der Daten wurden mehrere Diagramme erstellt. Diese helfen dabei, ein Verständnis für die Daten zu bekommen, und sie werden hier aufgelistet.

Analyse Resultat	Beschreibung
 *Abbildung 11 Kreisdiagramm Sprachen*	Es wurden insgesamt 796k Dokumente von der Firma Weblaw zur Verfügung gestellt. Diese beinhalten Dokumente, in denen drei Landessprachen, nämlich Deutsch, Italienisch und Französisch, vorkommen. Aufgrund des höheren Mehraufwandes und der Komplexität wurde die Bachelorarbeit nur auf deutsche Dokumente beschränkt.
 *Abbildung 12 Kreisdiagramm Arten von Dokumenten*	Es gibt drei Arten von Dokumenten: Gerichtsentscheide, Gesetzesartikel und wissenschaftliche Literatur. Um die Komplexität und die Dauer bei den Berechnungen auf der Rechenmaschine zur verringern, wurde entschieden, nur deutsche Gerichtsentscheide zu berücksichtigen.

Analyse Resultat	Beschreibung
 *Abbildung 13 Kreisdiagramm Quellen*	Die Dokumente, welche in dieser Bachelorarbeit zur Verfügung gestellt wurden, sind alle öffentlich. Weblaw hat seine Dokumente mit einem Webcrawler von unterschiedlichen Seiten bezogen. Die meisten Dokumente stammen aber von admin.ch, weblaw.ch selber und baselland.ch (insgesamt 246k Dokumente). Die restlichen 46k Dokumente entstammen anderen Quellen.
 *Abbildung 14 Kreisdiagramm Sachverhalt*	Es gibt viele Dokumente, welche keinen Sachverhalt aufweisen (35k Dokumente). Nur Gerichtsentscheide mit Sachverhalt (74k Dokumente) können für fastText gebraucht werden, weil die Vektoren auf Sachverhalten trainiert werden.
Durchschnittliche Anzahl Wörter im Body über allen Dokumenten: **1752**	Die durchschnittliche Anzahl der Wörter im Body der Dokumente beträgt 1752. Das sind etwa vier A4-Seiten Text, wobei in dieser Angabe auch wissenschaftliche Literatur gezählt wurde. Gerichtsentscheide sind ein bis zwei A4-Seiten lang.

Analyse	Beschreibung
Durchschnittliche Anzahl erkannter Thesaurus-Wörter über allen Dokumenten: **84**	Erkannte Thesaurus-Wörter zählt die Anzahl der Wörter im Text, für welche eine ID erstellt wurde. Das sind Wörter, welche im Thesaurus vorkommen. Diese Wörter werden in der Webansicht von Lawsearch als Schlagwörter gekennzeichnet.
Durchschnittliche Anzahl erkannter Thesaurus-Wörter mit Übersetzungen: **63**	Wenn Thesaurus-Wörter im Text vorkommen, kann es sein, dass im Thesaurus Übersetzungen davon existieren. Durchschnittlich muss es deshalb für weniger Wörter Übersetzungen geben, als es erkannte Thesaurus-Wörter gibt.
Durchschnittliche Anzahl erkannter Thesaurus-Wörter relativ zur Dokumentlänge: **7.6**	Relativ zur Textlänge werden jeweils pro Text 7.6 Wörter als Thesaurus-Wörter markiert.
Durchschnittliche Anzahl erkannter Thesaurus-Wörter mit Übersetzungen relativ zur Dokumentlänge: **6.3**	Relativ zur Textlänge werden jeweils pro Text 6.3 Wörter als Thesaurus-Wörter, die Übersetzungen haben, markiert.

Tabelle 1 Statistiken

Damit diese Bachelorarbeit übersichtlicher und nachvollziehbarer ist, wurde mithilfe von Weblaw ein expliziter Testfall ausgewählt. Dieser Muster-Testfall wird als Nächstes aufgeführt.

4.2 Muster-Testfall

Der Muster-Testfall dient dazu, einen klaren roten Faden in dieser Bachelorarbeit zu bilden. Er wurde ausgewählt, weil er einen klaren und leicht verständlichen Sachverhalt aufweist. Dieser Muster-Sachverhalt dient in den folgenden Kapiteln als Ausgangspunkt für die Evaluation und die Resultate. Der Muster-Testfall ist öffentlich und kann im Anhang 12.2 oder auch auf „lawsearch.ch“ eingesehen werden. Der Titel der Gerichtsentscheids ist 6S.522/2006.

A. Z._____ musste sich am 22. März 2001 um 19.30 Uhr in Unterehrendingen einer polizeilichen Kontrolle unterziehen, nachdem er zuvor seinen Personenwagen gelenkt hatte. Die Blutprobe von 20.15 Uhr ergab eine Blutalkoholkonzentration von 2,11 - 2,33 Promille.

B. Das Bezirksamt Baden verurteilte Z._____ am 11. April 2002 wegen Fahrens in angetrunkenem Zustand zu einer unbedingten Gefängnisstrafe von 30 Tagen und Fr. 1'800.-- Busse. Gleichzeitig widerrief es den bedingten Strafvollzug einer Gefängnisstrafe von 24 Tagen, die das Bezirksamt am 19. Juli 2000 ausgesprochen hatte.

Gegen diesen Strafbefehl erhob der Verurteilte Einsprache mit dem Antrag, er sei wegen Unzurechnungsfähigkeit freizusprechen. Gestützt auf ein psychiatrisches Gutachten sprach das Bezirksgericht Baden Z._____ am 17. Juni 2004 von Schuld und Strafe frei und trat auf den Widerrufsantrag nicht ein.
Auf Berufung der Staatsanwaltschaft des Kantons Aargau sprach das Obergericht des Kantons Aargau Z._____ am 16. Oktober 2006 schuldig des fahrlässigen Führens eines Motorfahrzeugs in angetrunkenem Zustand gemäss Art. 12 StGB i.V.m. Art. 91 Abs. 1 SVG und bestrafte ihn mit 20 Tagen Haft und einer Busse von Fr. 1'000.--. Den Vollzug der Freiheitsstrafe schob das Gericht zugunsten einer ambulanten Therapie auf. Hinsichtlich der früheren Freiheitsstrafe verlängerte es die Probezeit um ein Jahr.

C. Z._____ führt staatsrechtliche Beschwerde und Nichtigkeitsbeschwerde und beantragt, der angefochtene Entscheid sei aufzuheben.
Das Obergericht und die Staatsanwaltschaft haben auf Gegenbemerkungen verzichtet.

Zusammengefasst handelt sich bei diesem Sachverhalt um eine Person, welche betrunken Auto gefahren war. Die beschuldigte Person erhob eine Einsprache. Dadurch konnte die Person eine Minderung der Strafe seitens des Obergerichts erreichen.

Diese Art der Dateien wird bei Lawsearch in Form eines XML-Dokuments gespeichert. Dieses wurde mithilfe von Thesaurus indexiert. Zusätzlich wurde das Dokument mit Metainformationen angereichert. Im nächsten Kapitel wird der Aufbau des XML-Dokuments erklärt.

4.3 XML-Aufbau

Das XML-Dokument ist aus vier Teilen aufgebaut: aus den Metainformationen, den Thesaurus-Übersetzungen, den von Thesaurus erkannten Wörtern und dem Body, in dem der Inhalt angezeigt wird.

Dieses Bild wurde seitens Weblaw AG zensiert, weil diese Informationen vertraulich sind

Abbildung 15 XML-Aufbau

In Abbildung 15 ist ein XML-File eines Dokuments zu sehen. Die für diese Arbeit wichtigsten Informationen wurden daraus extrahiert. Diese werden in der nachfolgenden Tabelle 2 beschrieben.

Tags	Beschreibung
<dataSource>	DataSource beschreibt die Art des Dokuments, welches importiert wurde.
<doclang>	Doclang beinhaltet die Sprache des Textes. Mit dieser Information wurden die deutschen Texte extrahiert.
<url>	Bei der URL kann man ablesen, woher diese Dokumente stammen.
<thes_descriptor>	Unter diesem Tag wurden die einzelnen Thesaurus-Wörter, welche eine Übersetzung haben, eingetragen und mit einer ID versehen. Man kann auch einen Score von 1.00 ablesen. Das heisst, es gibt nur ein Wort, welches im Dokument gefunden wurde und welches im Thesaurus-Wörterbuch vorkommt. In diesem Beispiel ist es das Wort „Datenschutz“.
<tagtocode>	Unter diesem Tag wurden alle Wörter aufgelistet, welche im Thesaurus vorkommen und mit einer ID versehen wurden. Diese ID wird auch im Body angezeigt, damit man dieses Wort wiederfindet. Wichtig ist dabei der Score.

Tabelle 2 XML-Tags

Sobald die Daten ausgewählt sind, welche die Aufgabenstellung lösen, kann die Datenvorbereitung beginnen.

5 Datenvorbereitung

In diesem Kapitel wird die Datenvorbereitung erläutert. Diese ist jeweils in zwei unterschiedliche Datensätze aufgeteilt. Zum einen werden die Daten für die Netzwerkanalyse aufbereitet und zum anderen für die Word Embedding. Um die Daten in Elasticsearch zu indexieren, müssen die Daten ins JSON-Format umgewandelt werden.

5.1 Unterscheidung Word Embedding und Netzwerkanalyse

In den theoretischen Grundlagen wurden diese beiden Themen ausführlich vorgestellt. Mithilfe von Word Embedding werden die Wörter des Textes in Vektoren umgewandelt. Dann können die Vektoren von bestehenden Texten mit neu verfassten Texten verglichen werden. Mithilfe der Netzwerkanalyse ist es nicht möglich, neue Texte zu analysieren. Das Netzwerk wird über bereits bestehende Gerichtsentscheide gebildet.

Bei der Datenvorverarbeitung werden die Schritte aufgelistet, die es braucht, um die Daten für die unterschiedlichen Verfahren vorzubereiten.

5.1.1 Datenvorverarbeitung Word Embedding

Um die Vektoren der Wörter herzustellen, braucht es einen sehr grossen Korpus an Daten, damit man den Vektor von einzelnen Wörtern präzise bestimmen kann. Die Vektoren sollen die Wörter korrekt in semantischer Form repräsentieren. Es gilt generell: je mehr Texte aus einem bestimmten Fachgebiet trainiert wird, desto besser sind die Vektoren.
Deshalb wurden von der Webseite openjur.de noch mehr Gerichtsentscheide zum Korpus hinzugefügt. OpenJur ist eine deutsche freie Rechtsprechungsdatenbank, in welcher ca. 400‘000 Gerichtsentscheide für die Öffentlichkeit frei zur Verfügung gestellt werden [23]. Bevor die Vektoren mit fastText erstellt wurden, wurden alle Sonderzeichen aus den Daten entfernt. Auch wurden mögliche Absätze und überflüssige Leerzeichen gelöscht. Alle Wörter wurden in die Kleinschreibung umgewandelt. Die Reihenfolge der Wörter darf nicht geändert werden. Auch dürfen die Stoppwörter nicht entfernt werden, weil diese benötigt werden, um den Zusammenhang zwischen den einzelnen Wörtern im Text zu berechnen. Sobald man die Texte gesäubert hat, können diese für fastText verwendet werden.

5.1.2 Datenvorverarbeitung Netzwerkanalyse

Bei der Erstellung eines Netzwerkes oder eines Graphen werden die Inhalte der Gerichtsentscheide in den Knoten nicht gespeichert. Es werden nur der Gerichtsentscheide-Titel und die jeweils referenzierten Gerichtsentscheide-Titel in den Knoten und Kanten abgespeichert. Die Titel sind jeweils anders formatiert. Weblaw hat in seinen XML-Dokumenten in den Metadaten einen Tag namens „<dossvaria>“. Dort wurden die unterschiedlichen Arten von Gerichtsentscheid-Titeln aufgelistet. Das heisst, die Gerichtsentscheide-Titel müssen mithilfe des Tags „dossvaria“ in eine Grundform gebracht werden.
Damit die Daten schnell abrufbar sind, wurden nur die wichtigsten Daten aus den von Weblaw zur Verfügung gestellten Dokumenten extrahiert und ein JSON-File erstellt. Der Aufbau des JSON wird im Kapitel 5.3 „JSON-Aufbau“ beschrieben.

5.2 JSON-Aufbau

Um die Dokumente mit Elasticsearch zu indexieren, müssen die Daten im JSON-Format abgespeichert werden. In der Tabelle 3 wurde der JSON-Aufbau im Muster-Testfall aufgelistet.

Tags	Beschreibung
“title_forced”	title_forced ist die Grundversion des Titels eines Gerichtsentscheids. Dieser wird für die Netzwerkanalyse verwendet.
“sachverhalt”	Der Sachverhalt ist ein Boolean-Wert, welcher kennzeichnet, ob der Gerichtsentscheid einen Sachverhalt beinhaltet oder keinen Satzverhalt besitzt.
“bge”	BGE steht für Bundesgerichtsentscheide, welche in diesem Muster-Gerichtsentscheid vorkommen. Diese dienen als Grundlage für die Netzwerkanalyse, um das Netzwerk zu bilden.
“bgu”	BGer steht für Urteile des Bundesgerichts. Auch diese werden genutzt, um das Netzwerk zu bilden.
“sr_gesetze”	Es gibt Gerichtsentscheide, welche im Text nicht nur auf Gerichtsentscheide referenzieren, sondern auch auf Gesetze. In diesem Fall handelt sich um systematische Rechtssammlungsgesetze.
“text”	Hier wird der eigentliche Sachverhalt gespeichert.

Tabelle 3 JSON-Aufbau

```
{
    "title_forced": "BGer 6S.522/2006",
    "sachverhalt": "true",
    "gesch_nr": "6S.522/2006",
    "bges": {
        "bge": [
            "BGE-129-IV-49",
            "BGE-117-IV-292"
        ],
        "no_link_bge": "BGE-131-I-57"
    },
    "bgus": {
        "bgu": [
            "BGer 6S.522/2006",
            "BGer 6S.522/2006"
        ]
    },
    "bvges": null,
    "bvgers": null,
    "as_gesetze": null,
    "bbl_gesetze": null,
    "sr_gesetze": {
        "sr": [
            "SR-741.01",
            "SR-173.110",
            "SR-311.0"
        ]
    },
    "label_de": "Verfahren",
    "text": "A. Z._____ musste sich am 22. März 2001 um 19.30 Uhr in Unterehrendingen einer polizeilichen Kontrolle unterziehen, nachdem er zuvor seinen Personenwagen gelenkt hatte. Die Blutprobe von 20.15 Uhr ergab eine Blutalkoholkonzentration von 2,11 - 2,33 Promille. B. Das Bezirksamt Baden verurteilte Z._____ am 11. April 2002 wegen Fahrens in angetrunkenem Zustand zu einer unbedingten Gefängnisstrafe von 30 Tagen und Fr. 1'800.-- Busse. Gleichzeitig widerrief es den bedingten Strafvollzug einer Gefängnisstrafe von 24 Tagen, die das Bezirksamt am 19. Juli 2000 ausgesprochen hatte. Gegen diesen Strafbefehl erhob der Verurteilte Einsprache mit dem Antrag, er sei wegen Unzurechnungsfähigkeit freizusprechen. Gestützt auf ein psychiatrisches Gutachten sprach das Bezirksgericht Baden Z._____ am 17. Juni 2004 von Schuld und Strafe frei und trat auf den Widerrufsantrag nicht ein. Auf Berufung der Staatsanwaltschaft des Kantons Aargau sprach das Obergericht des Kantons Aargau Z._____ am 16. Oktober 2006 schuldig des fahrlässigen Führens eines Motorfahrzeugs in angetrunkenem Zustand gemäss Art. 12 StGB i.V.m. Art. 91 Abs. 1 SVG und bestrafte ihn mit 20 Tagen Haft und einer Busse von Fr. 1'000.--. Den Vollzug der Freiheitsstrafe schob das Gericht zugunsten einer ambulanten Therapie auf. Hinsichtlich der früheren Freiheitsstrafe verlängerte es die Probezeit um ein Jahr. C. Z._____ führt staatsrechtliche Beschwerde und Nichtigkeitsbeschwerde und beantragt, der angefochtene Entscheid sei aufzuheben. Das Obergericht und die Staatsanwaltschaft haben auf Gegenbemerkungen verzichtet. Das Bundesgericht zieht in"
}
```


Abbildung 16 JSON-Aufbau Muster-Testfall

In Abbildung 15 wird das gesamte JSON-File zum Muster-Testfall aufgezeigt. Sobald die JSON-Files erstellt wurden, können diese von Elasticsearch indexiert werden.

5.3 Elasticsearch

Elasticsearch erleichterte die Verwaltung über die von Weblaw zur Verfügung gestellten Daten. Da alle Dokumente in Elasticsearch indexiert sind, erfolgt die Suche in den Texten sehr schnell. Um eine bessere Übersicht über die Daten zu verschaffen und um die Abfrage über die Daten zu erleichtern, wurde das RESTful-Web-Interface „Kibana“ verwendet. [24]

5.4 Kibana

Kibana ist ein Open-Source-Datenvisualisierungs-Plugin für Elasticsearch. Kibana ermöglicht die vereinfachte Suche und die Visualisierung der Daten in Elasticsearch und war daher essentiell für die Planung der Vorverarbeitung der Daten und für das weitere Verständnis der Daten [25].

6 Modellierung

Modellierung beschreibt die zwei wesentlichen Verfahren, welche in dieser Bachelorarbeit implementiert wurden. Im Kapitel 6.1 wird der Aufbau eines Netzwerks, zwischen sich gegenseitig verlinkenden Gerichtsentscheiden, beschrieben. Kapitel 6.2 beschreibt die Implementierung eines Word Embedding mit fastText.

6.1 Modellierung Netzwerkanalyse

In diesem Kapitel wird beschrieben, wie die Analyse eines Netzwerkes bestehend aus Gerichtsentscheiden bestritten wird. Anhand der Analyse werden zwei verschiedene Ranglisten erstellt. In den Ranglisten werden die Nachbarn eines gegebenen Gerichtsentscheides nach Relevanz absteigend sortiert. Die Relevanz kann unterschiedlich berechnet werden und schliesslich wurden zwei Verfahren selektiert, welche die Relevanz eines Nachbarn am besten messen.
In den Kapiteln „Adjazenzliste", „NetworkX", „Netzwerkanalyse mit NetworkX" wird der Werdegang des Netzwerkes oder Graphen beschrieben. In den Kapiteln „Custom-Scoring" und „PageRank-Scoring" werden die Ranglisten für die Nachbarknoten erläutert.

6.1.1 Adjazenzliste

Eine Adjazenzliste ist eine Sammlung ungeordneter Listen, die zur Darstellung eines Graphen oder Netzwerkes verwendet wird [26]. Jede ungeordnete Liste in der Adjazenzliste beschreibt die Menge der Nachbarn eines Knotens. In jeder dieser ungeordneten Listen werden die Referenzen eines Gerichtsentscheids als Nachbarn beschrieben. Folglich gibt es für jeden Gerichtsentscheid eine Liste aus Referenzen und alle diese Listen werden zu einer Adjazenzliste zusammengeführt.

Gerichtsentscheid A:	Gerichtsentscheid D	Gerichtsentscheid E	Gerichtsentscheid Z
Gerichtsentscheid B:	Gerichtsentscheid A		
Gerichtsentscheid C:	Gerichtsentscheid D	Gerichtsentscheid G	
Gerichtsentscheid D:			

Tabelle 4 Illustration Adjazenzliste

In der Tabelle 4 wird die Adjazenzliste für Gerichtsentscheide beispielhaft illustriert. Für die fett-geschriebenen Gerichtsentscheide werden in der gleichen Zeile die Nachbarknoten notiert. Gerichtentscheid D hat keine Nachbarknoten und hat daher keine Einträge.
Die Adjazenzliste wird in der Python-Library „NetworkX" gebraucht, welche im folgenden Kapitel „NetworkX" näher beschrieben wird.

6.1.2 NetworkX

NetworkX ist eine Python-Library zur Erstellung, Manipulation und Untersuchung der Struktur, Dynamik und Funktionen komplexer Netzwerke oder Graphen [27].
Die Adjazenzliste wird in dieser Library gebraucht, um den Multigraphen zu erstellen. Dieser Multigraph wird schliesslich für die Netzwerkanalyse verwendet.

6.1.3 Netzwerkanalyse mit NetworkX

Bei dem Multigraphen werden nur die Titel der Gerichtsentscheide in die Knoten gespeichert. Die Kanten repräsentieren die Referenzen und auf den Kanten wird die Ähnlichkeitszahl als Gewichtung abgespeichert. NetworkX bietet für die Analyse des Multigraphen verschiedenste Verfahren an, zum Beispiel PageRank.

6.1.4 PageRank-Scoring

Der Pagerank-Algorithmus wird über den gesamten Graphen im Voraus berechnet. Jeder Gerichtsentscheid erhält einen PageRank-Score, welcher aussagt, wie wichtig dieser im Graphen ist. Mithilfe des implementierten Pagerank-Scorings wird für einen Knoten über alle Nachbarknoten eine Rangliste nach Relevanz sortiert erstellt. Der Gerichtsentscheid auf dem ersten Platz wird für ein Ausgangsdokument am relevantesten sein.

6.1.5 Custom-Scoring

Das Custom-Scoring erstellt ebenfalls eine Rangliste über alle Nachbarknoten. Allerdings wird nicht wie beim PageRank-Scoring im Voraus ein Score über alle Knoten im Graphen berechnet. Es wird bei den Nachbarknoten geschaut, wie oft diese von anderen Knoten referenziert werden und wie oft sie andere Nachbarknoten referenzieren. Es wird also bei den Nachbarknoten untersucht, wie viele eingehende Pfeile und wie viele ausgehende Pfeile die Nachbarknoten haben.

Dabei spielt bei der Gewichtung oder Score-Vergabe eine Rolle, ob die Nachbarknoten eingehende oder ausgehende Pfeile besitzen. Die exakte Gewichtung für die eingehenden und ausgehenden Pfeile bei den Nachbarknoten wird in Kapitel 7 “Umsetzung” genauer beschrieben.

6.2 Modellierung Word Embedding

In diesem Kapitel wird der Werdegang der Word Embedding Schritt für Schritt erklärt. Für die Erstellung der Word Embedding wurde die oben bereits erwähnte Library „fastText” verwendet. Vor der Erstellung der Word Embedding sollte festgestellt werden, dass die Daten, welche der Library mitgegeben werden (siehe Kapitel 5 „Datenvorbereitung”) richtig vorverarbeitet sind.

6.2.1 Model trainieren

In diesem Kapitel wird das Trainieren von Modellen anhand vorverarbeiteter Daten beschrieben.
Das Erstellen der Vektoren wird auch als „trainieren” bezeichnet. Trainiert man alle Wörter-Vektoren basierend auf allen Wörtern, die im Korpus vorkommen, erhält man am Ende vom fastText-Tool ein sogenanntes Model, welches alle Wörter-Vektoren in einem binären File abspeichert. Das Model wird ebenfalls als vec-File ausgegeben [14]. Bei der Implementierung wurde nur das binäre File verwendet.

Das Model repräsentiert gewissermassen die Rechtssprache in Gerichtsentscheiden und mit diesem Modell können nun mathematische Ähnlichkeitsberechnungen (z. B. Kosinus-Ähnlichkeit) zwischen Wörtern oder Texten durchgeführt werden. Beispielsweise kann berechnet werden, wie prozentual ähnlich die Wörter „Klagen” und „Anzeigen” sind. Hätte man ein Modell basierend auf Arzt-Berichten trainiert und die Ähnlichkeit zwischen „Klagen” und „Anzeigen” berechnet, würde man eine kleinere Prozentzahl erhalten als bei dem Modell basierend auf Gerichtsentscheiden. Ergo die Wörter sind sich hier weniger ähnlich als sie es in der Realität sind.

Um das Modell zu erstellen, welches die Rechtssprache am besten abbildet, wurden zuerst verschiedene Modelle basierend auf zwei verschiedenen Korpussen trainiert. Der erste Korpus enthält nur schweizerische Gerichtsentscheide von Weblaw. Der zweite enthält

schweizerische und deutsche Gerichtsentscheide von openJur. Es wurden zwei verschiedene Korpusse verwendet, um bei der Evaluation festzustellen, ob anhand zusätzlicher deutscher Gerichtsentscheide bessere Modelle kreiert werden können.

In einem weiteren Schritt wurden, basierend auf den zwei Korpussen, verschiedene Modelle mit verschiedensten Eigenschaften trainiert. fastText bietet zwei verschiedene Modell-Architekturen an: Einerseits das „Skipgram" und andererseits das „CBOW". Beide Verfahren werden im Kapitel 2.4.5 beschrieben. Im nächsten Kapitel 6.2.4 werden Parameter, welche beim Trainieren mitgegeben werden, um die unterschiedlichen Modelle zu trainieren, erklärt.

6.2.2 Unterschiedliche Parameter

Diese Parameter können beim Trainieren mitgegeben werden, müssen aber nicht mitgegeben werden. Werden diese nicht mitgegeben, wird bei jedem Parameter ein Default-Wert angenommen.

- Dimensionen: Die Dimension steuert die Grösse der Vektoren an. Je grösser sie sind, desto mehr Informationen können sie erfassen, sie benötigen dann aber mehr Daten. Wenn die Vektoren zu gross sind, sind sie schwieriger zu trainieren und beanspruchen daher mehr Zeit beim Trainieren. Der Default-Wert beträgt 100 [14].
- Epoch: Es steuert, wie oft die Daten durchlaufen werden beim Trainieren. So werden die Vektoren besser beziehungsweise können die Semantik besser abbilden. Der Default-Wert beträgt 5 [14].
- Learning-Rate: Je höher die Learning-Rate ist, desto schneller konvergiert das Modell zu einer Lösung, jedoch mit dem Risiko eines Overfittings an den Daten. Der Default-Wert beträgt 0.01 [14].

Bei Epoch wurde immer der Wert 10 angenommen. Die Daten sind also zehnmal durch das Training gelaufen, um sicherzustellen, dass die Vektoren gut geschliffen sind. Bei der Learning-Rate wurde der Wert 0.1 als guter Wert erkannt.
Anhand dieser Parameter wurden schlussendlich 12 Modelle trainiert. Diese werden nun in der nachfolgenden Tabelle 5 aufgelistet und erläutert. Die 12 Modelle werden im Kapitel 8 „Evaluation" ausgewertet.

Modelle	Architektur	Dimensionen	Korpus
lawsearch_skipgram_dim100	Skipgram	100	lawsearch
lawsearch_skipgram_dim200	Skipgram	200	lawsearch
lawsearch_skipgram_dim300	Skipgram	300	lawsearch
lawsearch_cbow_dim100	CBOW	100	lawsearch
lawsearch_cbow_dim200	CBOW	200	lawsearch
lawsearch_cbow_dim300	CBOW	300	lawsearch
lawsearch_openjur_skipgram_dim100	Skipgram	100	lawsearch + openjur
lawsearch_openjur_skipgram_dim200	Skipgram	200	lawsearch + openjur
lawsearch_openjur_skipgram_dim300	Skipgram	300	lawsearch + openjur
lawsearch_openjur_cbow_dim100	CBOW	100	lawsearch + openjur
lawsearch_openjur_cbow_dim200	CBOW	200	lawsearch + openjur
lawsearch_openjur_cbow_dim300	CBOW	300	lawsearch + openjur

Tabelle 5 Parameter fastText-Modelle

7 Umsetzung

In diesem Kapitel wird auf die Kernpunkte bei der Umsetzung des Prototyps eingegangen. Zudem werden wichtige Entscheidungen bei der Umsetzung ausführlich begründet.

Für die Umsetzung wurden verschiedene Komponenten verwendet: die Python-Umgebung, Elasticsearch und fastText. Es werden wichtige Code-Stellen analysiert, welche für die Logik des Prototyps wesentlich sind. Technische Details zur Umsetzung werden im Anhang beschrieben.

7.1 Python-Umgebung

In der Python-Umgebung werden die gesamte Logik bzw. die Python-Codes ausgeführt. Unter der Logik werden das Erstellen der Ranglisten für Nachbarknoten und die Ähnlichkeitsberechnung zwischen einem neuen Sachverhalt und einem bestehenden Sachverhalt verstanden. Nebst der Logik wurde die Python-Umgebung auch für die Vorverarbeitung der Daten verwendet. Die technischen Details zur Vorverarbeitung der Daten werden ebenfalls im Anhang aufgelistet.

7.2 Netzwerkanalyse

Für die Netzwerkanalyse werden die beiden Ranking-Verfahren PageRank-Scoring und Custom-Scoring, welche die Relevanz der Nachbarknoten schlussendlich beschreiben, genauer betrachtet.

7.2.1 PageRank + fastText

```
1  def pagerank_fasttext(input,index):
2      page_rank_fasttext = dict()
3      for a in neighbors_and_across_neighbors(input):
4          if G.has_edge(input,a):
5              page_rank_fasttext[a] = pr[a]*1000+ G[input][a][0]['weight']
6          if G.has_edge(a,input):
7              page_rank_fasttext[a] = pr[a]*1000+ G[a][input][0]['weight']
8          else:
9              page_rank_fasttext[a] = pr[a]*1000 + calculate_similarity(get_text_elastic(input,index), get_text_elastic(a,index))
10     page_rank_fasttext = collections.OrderedDict(sorted(page_rank_fasttext.items(), key=lambda t: t[1],reverse=True))
11     return page_rank_fasttext
```

Abbildung 17 Code PageRank + fastText

In Abbildung 17 wird die Funktion für die Ranglistenerstellung anhand von PageRank und fastText aufgezeigt, welche als Parameter den Multigraphen nimmt und als Rückgabewert ein Dictionary mit Nachbarknoten und ihre dazugehörigen PageRank-Scores gibt. Diese Funktion wird nun näher beschrieben.

Zeile 3:
Zuerst werden alle Nachbarn des Knotens gefunden und diese werden zugleich durchiteriert.

Zeile 4 - 7:
Für alle Nachbarn wird der PageRank-Score aus der PageRank-Dictionary ausgesucht und dieser wird um 1000 multipliziert, um mehr Gewichtung auf den PageRank-Score zu setzen. 1000 wurde als ein guter Faktor eingestuft, weil der PageRank-Score überall zwischen 1e -06 und 1e -04 schwankt. Würde der PageRank-Score mit 1000 multipliziert werden, würde er zwischen 1e -03 und 1e -01 schwanken und wäre somit fast gleich gross wie die Ähnlichkeitszahl, welche zwischen 1e -02 und 1e -01 schwankt. Auf die Ähnlichkeitszahl wurde bewusst eine höhere Gewichtung gesetzt, weil die semantische Ähnlichkeit zwischen den Gerichtsentscheiden als wichtiger empfunden wird als die Referenzierung zwischen den Gerichtsentscheiden.

Nachdem der Score vom Dictionary abgerufen wurde, wird zum Score die Ähnlichkeitszahl, welche bei der Erstellung des Multigraphen von fastText schon berechnet wurde, addiert und in einem neuen Dictionary gespeichert.

Zeile 9:

Für die „across-neighbors-Knoten" hingegen muss die Ähnlichkeit im Nachhinein von fastText berechnet werden, da diese Knoten zum Ausgangsknoten keine Kanten haben und somit keine Ähnlichkeitszahl besitzen. Die Ähnlichkeit wird in der Funktion „calculate_similarity" berechnet, in welcher eine Abfrage nach dem Text bei Elasticsearch stattfindet. Die Ähnlichkeit zwischen den Texten des Ausgangsknoten und des „across-neighbors-Knoten" wird nach der Abfrage berechnet und schliesslich zum PageRank-Score addiert. Näheres zu dieser Funktion wird im Anhang oder im Code selbst beschrieben.

Zeile 10 - 11:

Am Ende wird die Rangliste nach Score absteigend über die Nachbarknoten erstellt und zurückgegeben.

7.2.2 Custom-Scoring + fastText

```
1  def custom_scoring_fasttext(input):
2      neighbors_scoring = dict()
3      score = 0
4      flag = True
5      all_neighbors = neighbors_and_across_neighbors(input)
6      across_neighbors = across_out_edge(input)|across_in_edge(input)
7      for a in all_neighbors:
8          if a in G.successors(input):
9              score = 100
10             if a in G.predecessors(input):
11                 score = 110
12             if G.successors(a):
13                 for s in G.successors(a):
14                     score = score + 10
15             if G.predecessors(a):
16                 for s in G.predecessors(a):
17                     score = score + 20
18         elif a in G.predecessors(input):
19             score = 90
20             if G.successors(a):
21                 for s in G.successors(a):
22                     score = score + 10
23             if G.predecessors(a):
24                 for s in G.predecessors(a):
25                     score = score + 20
26
27         #across_neighbors
28
29         elif a in across_out_edge(input):
30             score = 100
31             score = score + calculate_similarity(get_text_elastic(input), get_text_elastic(a)) * 1000
32             flag = False
33
34         elif a in across_in_edge(input):
35             score = 90
36             score = score + calculate_similarity(get_text_elastic(input), get_text_elastic(a)) * 1000
37             flag = False
38
39         if flag:
40             if G.has_edge(input,a):
41                 score = score + G[input][a][0]['weight'] * 1000
42             if G.has_edge(a,input):
43                 score = score + G[a][input][0]['weight'] * 1000
44         neighbors_scoring[a] = score
45         neighbors_scoring = collections.OrderedDict(sorted(neighbors_scoring.items(), key=lambda t: t[1],reverse=True))
46     return neighbors_scoring
```

Abbildung 18 Code Custom-Scoring + fastText

In Abbildung 18 wird aufgezeigt, wie das Custom-Scoring zusammen mit fastText implementiert wurde. Als Parameter nimmt die Funktion ebenfalls den Titel des gegebenen Gerichtsentscheides und als Rückgabewert gibt sie gleichermassen eine Rangliste aus.

Zeile 7 - 25:

In diesen Zeilen wird die Gewichtung über die direkten Nachbarn anhand von Grafiken erläutert. A ist dabei der Ausgangsknoten oder der gegebene Gerichtsentscheid. B ist der Nachbar des Knotens A. Ein Pfeil von A nach B bedeutet, dass B in A referenziert wird.

Zeile 18 - 19:

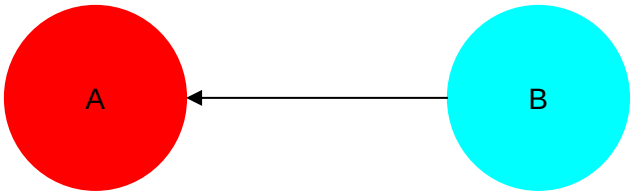


Abbildung 19 Custom-Scoring Gewichtung 1

Wenn A von B referenziert wird, erhält B 90 Punkte.

Zeile 8 - 9:

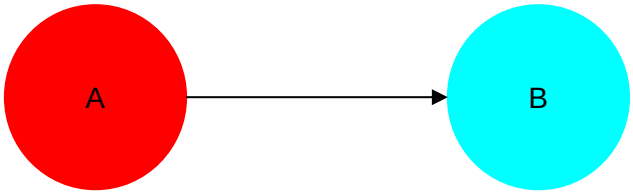


Abbildung 20 Custom-Scoring Gewichtung 2

Wenn B von A referenziert wird, erhält B 100 Punkte. B erhält in diesem Fall mehr Punkte, weil Gerichtsentscheide, welche im gegebenen Gerichtsentscheid (A) erwähnt werden, für den Benutzer interessanter sind.

Zeile 10 – 11:

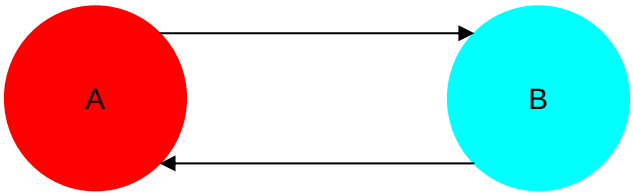


Abbildung 21 Custom-Scoring Gewichtung 3

Wenn B von A referenziert wird, und A von B referenziert wird, erhält B 110 Punkte.

Zeile 12 - 14 und 20 - 22:

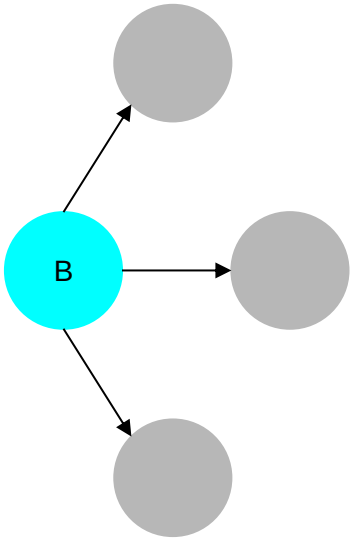


Abbildung 22 Custom-Scoring Gewichtung 4

Gegeben, dass B ein Nachbar von A ist, erhält B für jeden Gerichtsentscheid, der von B referenziert wird, 10 Punkte.

Zeile 15 - 17 und 23 - 25:

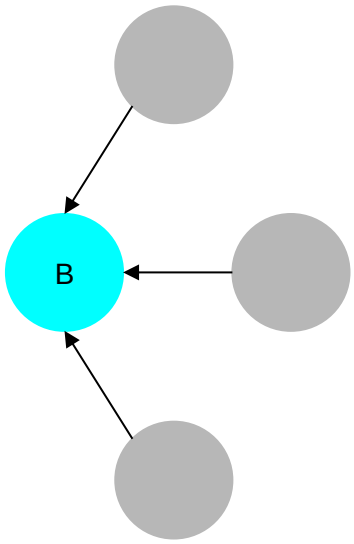


Abbildung 23 Custom-Scoring Gewichtung 5

Gegeben, dass B ein Nachbar von A ist, erhält B für jeden Gerichtsentscheid, welcher B referenziert, 20 Punkte. Ein Nachbar, welcher von anderen Gerichtsentscheiden referenziert wird, ist für den Benutzer interessanter als ein Nachbar, welcher auf andere Gerichtsentscheide referenziert. Deshalb werden in diesem Fall doppelt so viele Punkte vergeben wie in dem vorhergehenden Fall.

Zeile 29 - 37:

In diesen Zeilen findet die Punktevergabe für die „across-neighbors-Knoten" statt. Nachbarknoten eines „across-neighbor-Knoten", welche gar keine Kanten zwischen sich und A haben, haben keinen Einfluss auf die Relevanz und somit werden für einen „across-neighbor-Knoten", welcher Nachbarknoten besitzt, keine zusätzlichen Punkte wie bei den direkten Nachbarn vergeben.

Zeile 29 - 30:

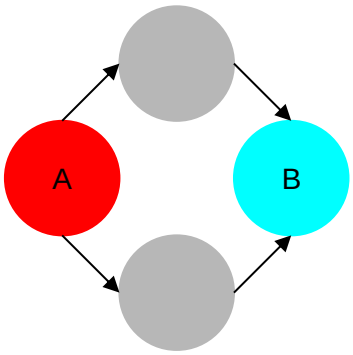


Abbildung 24 Custom-Scoring Gewichtung "across-neighbors-Knoten" 1

Die Berechnung des „across-neighbors-Knoten" in dem Fall, wie er in Abbildung 24 illustriert wird, findet in der Funktion „across_out_edge" statt. Hier werden für B 100 Punkte vergeben.

Zeile 34 - 35:

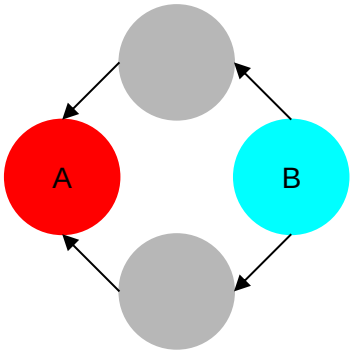


Abbildung 25 Custom-Scoring Gewichtung "across-neighbors-Knoten" 2

Die Berechnung des „across-neighbors-Knoten" in dem Fall, wie er in Abbildung 25 illustriert ist, findet in der Funktion „across_in_edge" statt. Hier werden für B 90 Punkte vergeben. Dieser Fall ist ähnlich dem Fall, welcher in Abbildung 24 dargestellt wird, und daher werden aus demselben Grund weniger Punkte vergeben als im vorhergehenden Fall.

Zeile 31, 36, 41, 43:

In diesen Zeilen wird der Score berechnet, welcher aus den vergebenen Punkten und der Ähnlichkeitszahl besteht. In den Zeilen 31 und 36 muss die Ähnlichkeitszahl im Nachhinein mit fastText berechnet werden, da keine Kante zwischen den „across-neighbors-Knoten" und dem gegebenen Knoten existiert, aus dem die Ähnlichkeitszahl ausgelesen werden kann. In den Zeilen 41 und 43 kann die Ähnlichkeitszahl aus den Kanten einfach herausgelesen werden. Die Ähnlichkeitszahl wird um 1000 multipliziert, damit die Ähnlichkeitszahl und die Punkte für die Referenzen in etwa gleich gross sind. Hier hingegen wurde auf die Referenzierungen zwischen den Gerichtsentscheiden mehr gewichtet als auf die semantische Ähnlichkeit zwischen den Gerichtsentscheiden, da die Gesamtpunktzahlen für die Anzahl der Referenzen zwischen den Gerichtsentscheiden vollkommen unterschiedlich sein können, und folglich kann nicht wie beim PageRank bestimmt werden, in welchem Bereich die Punktzahlen liegen können. Sind die Punktzahlen bei allen Nachbarknoten klein und ist die Gewichtung auf die semantische Ähnlichkeit sehr gross, überwiegt die semantische Ähnlichkeit vollkommen und die Referenzen spielen kaum eine Rolle. Der Faktor 1000 wurde als eine gute Gewichtung betrachtet, da diese Gewichtung fast so gross ist wie minimale Punktzahl eines Nachbarknoten, nämlich 90.

Zeile 44 - 46:

Die Scores werden in einem Dictionary gespeichert und aus dem Dictionary wird eine geordnete Rangliste erstellt und als Rückgabewert ausgegeben.

7.3 Word-Embedding

Anhand der fastText-Modelle, welche die Wörter-Vektoren beinhalten, können, wie erwähnt, mathematische Berechnungen durchgeführt werden. Im folgenden Kapitel wird die Implementierung der Kosinus-Ähnlichkeit im Detail erklärt.

7.3.1 Kosinus-Ähnlichkeit

```
1  def calculate_similarity(text1, text2):
2      text1 = preprocessing(text1)
3      text2 = preprocessing(text2)
4      vec1 = model.get_numpy_text_vector(text1)
5      vec2 = model.get_numpy_text_vector(text2)
6      vec1 = vec1.reshape(1,-1)
7      vec2 = vec2.reshape(1,-1)
8
9      similarity = cosine_similarity(vec1, vec2)
10     score = similarity * similarity * similarity
11     return score
```

Abbildung 26 Code Kosinus-Ähnlichkeit

In Abbildung 26 wird aufgezeigt, wie die Ähnlichkeitsberechnung implementiert wurde.
Diese Funktion nimmt zwei Texte als Parameter an und gibt als Rückgabewert die Ähnlichkeitszahl.

Zeile 2 - 3:
Als erstes werden die zu vergleichenden Texte überarbeitet. Die Funktion „preprocessing" wird im Code genauer beschrieben.

Zeile 4 - 5:
Hier werden die überarbeiteten Texte dem fastText-Modell übergeben. Mit der Funktion "get_numpy_text_vector" wird aus den Vektoren eines Wortes sowie aus den Vektoren, die mithilfe von Wort-Ngramms berechnet wurden, der Durchschnitt berechnet und der Durchschnitt wird als ein nicht-normalisierter Numpy-Vektor zurückgegeben [14].

Hierbei hätte ebenfalls die Funktion „get_numpy_sentence_vector" verwendet werden können. Diese berechnet aber den Text-Vektor als den Durchschnitt aller normalisierten Word-Vektoren [14]. Da diese Funktion bei der Berechnung des Text-Vektors nicht die Wort-Ngramms mit einbezieht und zu grösseren Ähnlichkeitszahlen führt, wurde auf sie verzichtet. Stattdessen wurde die Funktion „get_numpy_text_vector" gewählt.

Zeile 6 - 7:

Nachdem die Text-Vektoren berechnet wurden, werden diese Vektoren umgeformt.

Zeile 9 - 10:

In der Funktion „cosine_similarity" wird nun die Ähnlichkeit zwischen den Text-Vektoren berechnet. Die Ähnlichkeitszahl entsteht, indem die Zahl, welche von „cosine_similarity" zurückgegeben wird, sich selbst zweimal multipliziert. Anders gesagt, sie entsteht, indem die Zahl mit Exponent 3 potenziert wird. Da die Ähnlichkeitszahlen sehr nahe beieinander waren, wurde diese Skalierung vorgenommen, um die Abstände zwischen den verschiedenen Ähnlichkeitszahlen zu vergrössern. Die weniger ähnlichen Paare von Gerichtsentscheiden wurden dadurch, sozusagen, mehr bestraft. Folglich ist es zwischen den verschiedenen Ähnlichkeitszahlen nun besser ersichtlich, welche Paare von Gerichtsentscheiden ähnlich sind und welche Paare weniger ähnlich sind.

8. Evaluation

In diesem Kapitel wird beschrieben, wie die unterschiedlichen implementierten Verfahren evaluiert werden. Für die Evaluation wurde in dieser Bachelorarbeit der Kendall-Score verwendet. Der Kendall-Score betrachtet, wie stark Objekte aus zwei Listen korrelieren. Es werden jeweils die Verfahren Netzwerkanalyse, Word-Embedding und gemischte Verfahren mit dem goldenen Standard verglichen. Dann wurde ein Experiment mit einer rechtskundigen Person durchgeführt, bei dem die besten Verfahren manuell bewertet wurden.

8.1 Goldener Standard

Für die Evaluation wurde von einer rechtskundigen Person ein goldener Standard erstellt. Das bedeutet, die Person sollte für den Muster-Testfall eine persönliche Top-10-Rangliste mit Gerichtsentscheiden erstellen, in welcher sie die zum Muster-Testfall ähnlichsten Gerichtsentscheide auflistet. Das Problem war, dass der Jurist keinen Überblick über alle Gerichtsentscheide haben konnte. Die unterschiedlichen Verfahren wurden dann anhand der Kendall-Score-Berechnung mit dem goldenen Standard verglichen.

8.1.1 Verwendetes Verfahren: Netzwerkanalyse

Für den Muster-Testfall gab es zwei Gerichtsentscheide, die im Text erwähnt wurden. Das heisst, es gab nur zwei Nachbarn. Beide Nachbarn haben die gleiche Rangreihenfolge gehabt, deshalb ist dieses Verfahren wenig aussagekräftig. Das ist ein grosses Problem der Netzwerkanalyse nur mit den direkten Nachbarn: weil wenn es keine Nachbarn oder sehr wenige Nachbarn gibt, kann keine sinnvolle Rangfolge erstellt werden. Die Lösung wäre, die Anzahl Nachbarn zu erweitern mit den Nachbarn dieser Nachbarn.

8.1.2 Verwendetes Verfahren: Word-Embedding

fastText-Modelle lassen sich auf viele verschiedene Arten trainieren. Das Verfahren wurde mit vielen verschiedenen Parametern trainiert. Auch wurden die Modelle jeweils mit deutschen Gerichtsentscheiden von Openjur trainiert. In Tabelle 6 werden die Erkenntnisse aus den verschiedenen Parametern aufgezählt.

Parameter	Begründung
DIM100 vs. DIM200 vs. DIM300	Sehr auffällig ist: je mehr Dimensionen die Vektoren haben, desto bessere Resultate werden geliefert. Je mehr Dimensionen die Wörter-Vektoren haben, desto mehr Informationen können diese tragen.
SKIPGRAM vs. CBOW	Auch kann man sehen, dass SKIPGRAM bessere Resultate liefert als CBOW. SKIPGRAM versucht, ähnliche Worte anhand der naheliegenden Worte zu erkennen. CBOW bezieht sich auf den ganzen Kontext.
Mit Openjur vs. ohne Openjur	Hier kann man sehr gut erkennen, dass nicht mehr Dokumente unbedingt zu besseren Resultaten führen. Der Kendall-Score ist sogar mit den gelernten Openjur-Dokumenten schlechter. Das kann bedeuten, dass die schweizerischen Gerichtsentscheide und die deutschen Gerichtsentscheide sich vom Kontext unterscheiden.

Tabelle 6 Parameter Erkenntnisse

Die verschieden trainierten fastText-Modelle wurden jeweils mit dem goldenen Standard verglichen. Dabei wurde für jedes Modell der Kendal-Tau-Score berechnet. In Abbildung 27 ist zu sehen, wie gut jedes Modell abgeschnitten hat.

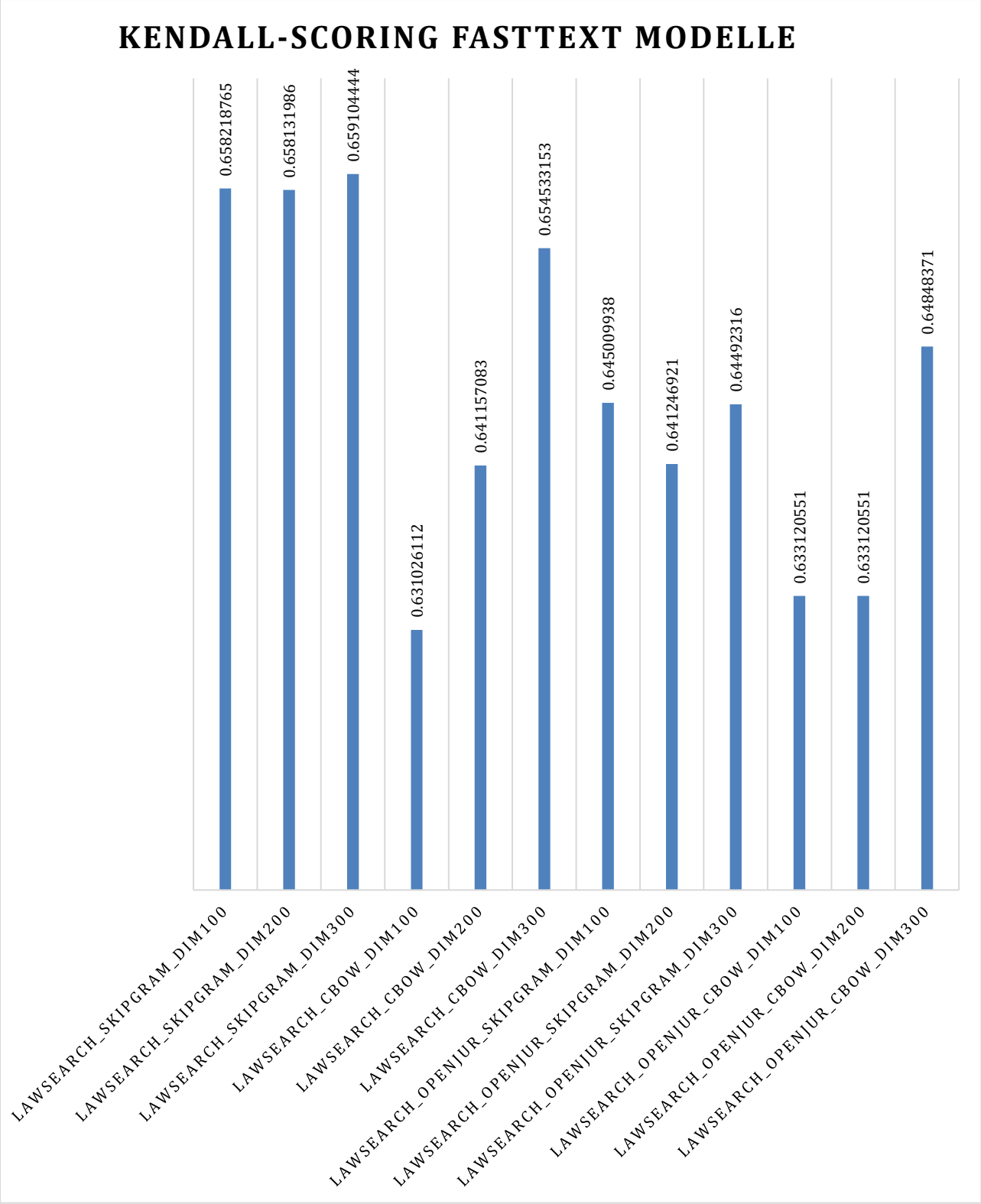


Abbildung 27 Kendall-Scoring fastText-Modelle

Das beste Resultat liefert der Skipgram-Algorithmus mit 300 Dimensionen. Bei gemischten Verfahren werden die Verfahren der Netzwerkanalyse und fastText kombiniert.

8.1.3 Gemischte Verfahren

Bei den gemischten Verfahren werden die Verfahren der Netzwerkanalyse mit dem besten Modell aus fastText kombiniert. In Abbildung 28 ist zu sehen, dass der Custom-Scoring viel schlechter ist als PageRank. Das kann daran liegen, dass zum einen die Parameter des Custom-Scorings besser abgestimmt werden müssen. Ein Problem des Custom-Scorings ist auch, dass dieses Verfahren nur die nächsten Nachbarn und ihre Kanten betrachtet. Beim PageRank werden über alle Knoten und alle Kanten Scores gebildet. Eine wichtige Anmerkung ist hier, dass die Kendall-Score-Werte nicht zwischen den einzelnen Tabellen verglichen werden können, weil diese eine unterschiedliche Anzahl an Elementen in den Ranglisten haben.

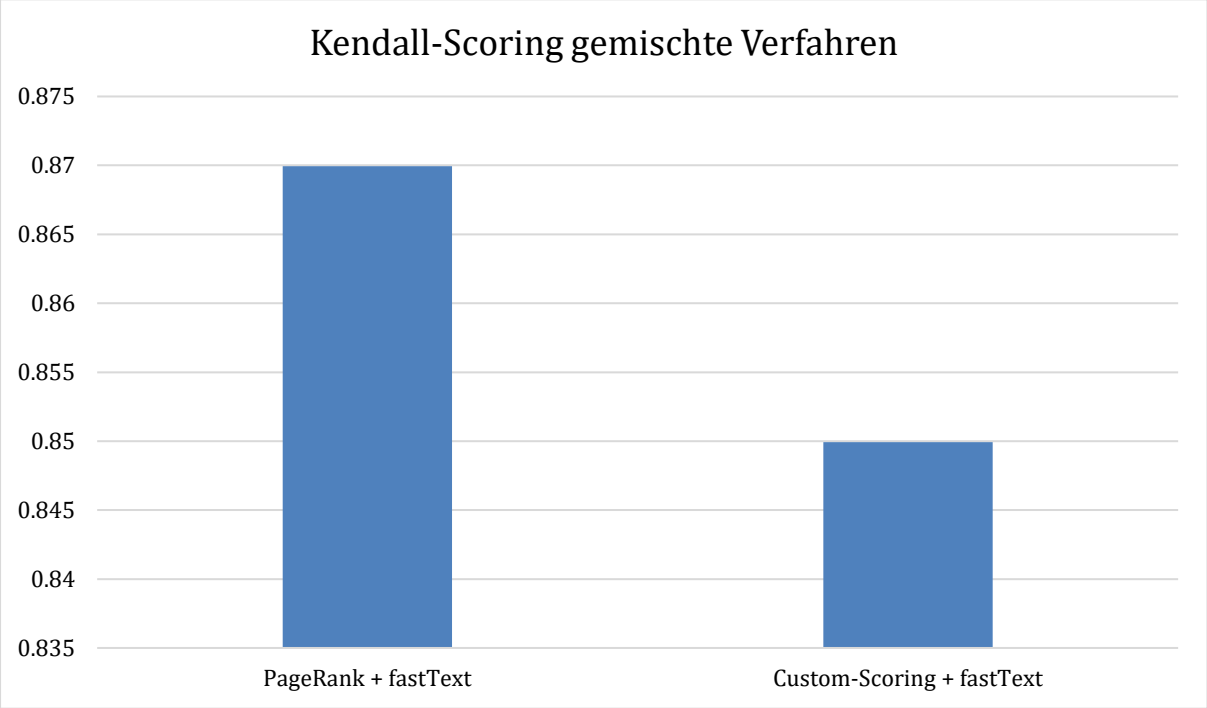


Abbildung 28 Kendall-Scoring gemischte Verfahren

Alle erstellten Werte wurden mithilfe des Kendall-Scores mit dem goldenen Verfahren verglichen. Dabei konnten die besten Verfahren aufgrund des Muster-Testfalls und mit dem Vergleich zum goldenen Standard ausgewählt werden. Es wurden fünf Verfahren ausgewählt.

8.2 Experiment mit dem Juristen

Nachdem die besten Verfahren ausgewählt wurden, werden Top-5-Ranglisten mit den fünf ähnlichsten Gerichtsentscheiden für die jeweiligen Verfahren generiert. Es werden jeweils die fünf ähnlichsten Dokumente mit dem Custom-Scoring-Verfahren, mit dem PageRank-Verfahren, mit dem fastText-Verfahren und den gemischten Verfahren erstellt und in eine Excel-Liste zur Auswertung hinzugefügt.
Neben diesen fünf Verfahren wurden die Top-5-Gerichtsentscheide von zwei Baselines auch zur Bewertung in die Liste eingefügt.
Zuletzt wurden die Top-5-Gerichtsentscheide des goldenen Standards in die Liste eingefügt. Die einzelnen Gerichtsentscheide wurden mit einer ID versehen und durchgemischt. Der Jurist muss nun die einzelnen Gerichtsentscheide bewerten. In Tabelle 7 wird erklärt, wie der Jurist die einzelnen Gerichtsentscheide bewertet hat.

Wertung von 0 - 5	**Beschreibung**
0 - gar nicht ähnlich	Die beiden Texte haben gar keine Gemeinsamkeiten.
1 - fast nicht ähnlich	Es geht grob um das gleiche Thema, aber die Texte sind sich nicht ähnlich.
2 - ein wenig ähnlich	Es geht grob um das gleiche Thema und einige wenige Sätze sind sich ähnlich.
3 - teilweise ähnlich	Es geht um das gleiche Thema und einige Sätze beschreiben den gleichen Sachverhalt.
4 - sehr ähnlich	Es geht um das gleiche Thema, die Sätze beschreiben einen sehr ähnlichen Sachverhalt und haben etwa den gleichen Umfang.
5 - identisch	Die Themen, der Sachverhalt und der Umfang sind identisch. Die Sätze bedeuten das gleiche, können aber umformuliert sein oder an einer anderen Position im Text stehen.

Tabelle 7 Erklärung zur Bewertung

Zudem wurde ein goldener Standard von dem Juristen erstellt. Der Jurist ist ein Experte der Firma Weblaw AG mit Rechtskenntnissen. Die einzelnen Ranglisten mit den ähnlichsten Gerichtsentscheiden wurden mithilfe von unterschiedlichen Verfahren erstellt. Der Kendall-Score konnte eine Aussage über die Korrelation zwischen zwei Ranglisten machen. Die Resultate der bewerteten Gerichtsentscheide nach ihrer Ähnlichkeit mit dem Muster-Testfall werden im folgenden Kapitel 9 erläutert.

9. Resultate

Die Resultate beschreiben die manuelle Auswertung des Juristen. Diese beinhalten zwei Auswertungen. Unter Auswertung wird das manuelle Bewerten der einzelnen Verfahren auf einer Skala von 1 bis 5 verstanden. Einmal wurde die Auswertung mit dem Muster-Testfall erstellt. Das Page-Rang-Verfahren und das Custom-Scoring-Verfahren ergaben die gleichen Werte. Deshalb wurde für diesen Fall noch einmal ein anderer Testfall ausgewählt und das Experiment wiederholt. Was die zwei Baselines bedeuten, wird in den zwei folgenden Kapiteln („Baseline Volltextsuche“ und „Baseline Gesetzesartikel“) erläutert.

9.1 Baseline Volltextsuche

Die Baseline stellt einen Standard auf, welcher in dieser Bachelorarbeit mindestens erreicht werden soll. Die erste Baseline ist die der Volltextsuche oder von Lawsearch. Da auf der Webseite keine grossen Texte an Lawsearch zur Suche mitgegeben werden können, wurde die Suche auf Elasticsearch simuliert. Elasticsearch zeigt die ähnlichsten Gerichtsentscheide mit Hilfe von TF-IDF. Es gilt, mit dem implementierten Verfahren ein besseres Resultat zu bekommen.

9.2 Baseline Gesetzestexte

Die zweite Baseline ist eine Vermutung. Nach einer intensiven Auseinandersetzung mit den Gerichtsentscheiden ist aufgefallen, dass zwei Gerichtsentscheide mehr Gesetztestexte gemeinsam haben, wenn sie vom Kontext her ähnlicher sind. Deshalb haben wir die prozentuale Übereinstimmung der Gesetzestexte zwischen zwei Gerichtsentscheiden als Baseline genommen und diese vom Juristen bewerten lassen.

9.3 Resultate Experiment

Die Resultate des Experiments ergeben, dass das fastText-Verfahren mit Skipgram vom Juristen als die beste Variante aller Verfahren bewertet wurde, dicht gefolgt von der Volltextsuche. Diese hat auch gut abgeschnitten. Beide Verfahren, das Word-Embedding und das TF-IDF-Verfahren, scheinen sehr gut zu funktionieren, wobei das Verfahren mit Word Embedding für den Muster-Testfall das beste Resultat lieferte. Die schlechteste Bewertung hat die Baseline Gesetzestexte erhalten. Die Vermutung, dass die prozentuale Übereinstimmung der Gesetzestexte mit der Ähnlichkeit der Sachverhalte einen Zusammenhang hat, hat sich nicht bestätigt.

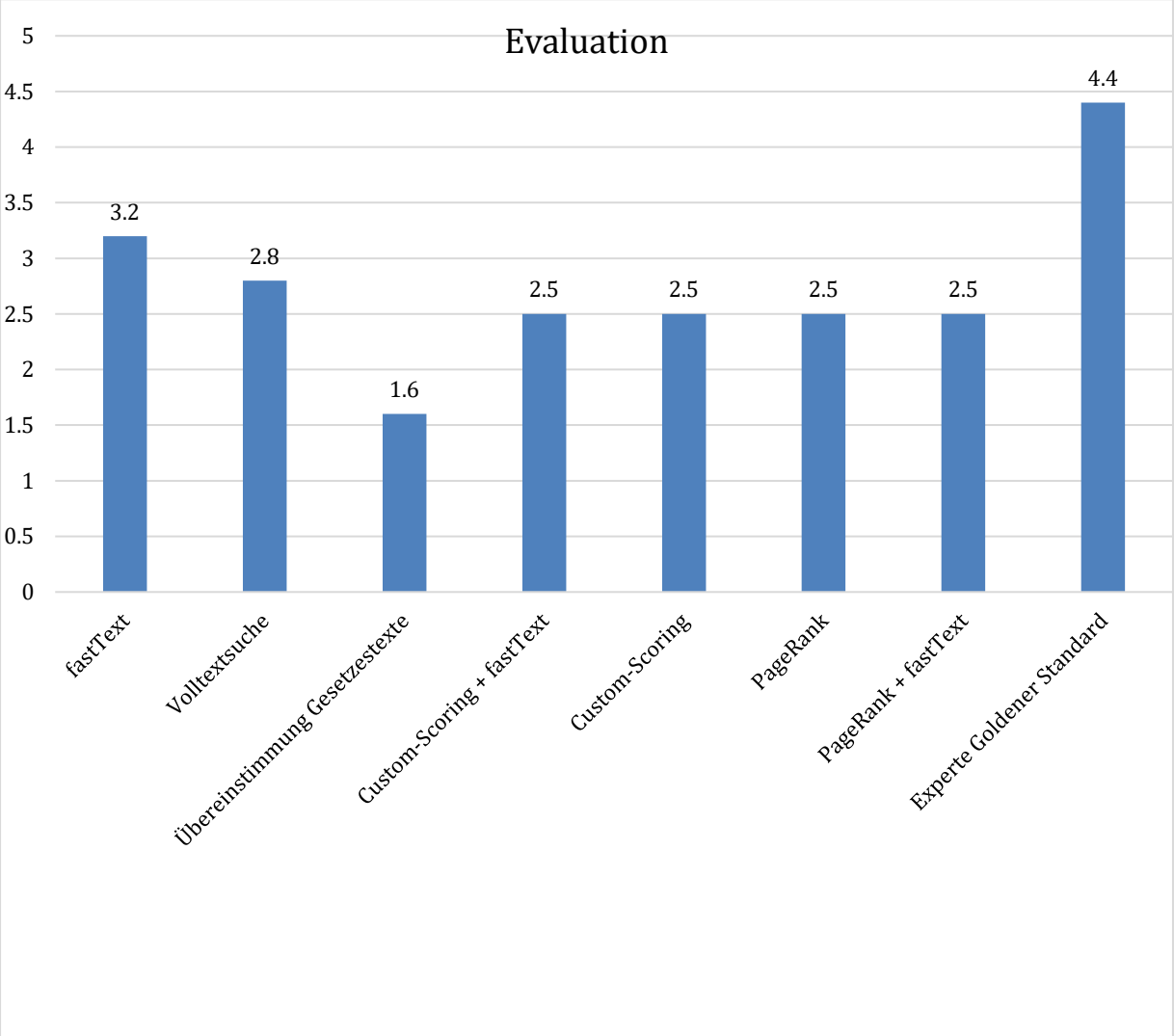


Abbildung 29 Evaluation vom Juristen

Die Verfahren mit PageRank und Custom-Scoring haben zum gleichen Ergebnis geführt. Deshalb wurde ein neuer Testfall erstellt, welcher mehr referenzierte Gerichtsentscheide enthielt. Der Jurist musste diese dann bewerten. Beim Custom-Scoring haben sich gar keine Ähnlichkeiten ergeben. Das PageRank-Verfahren hat besser abgeschnitten als das Custom-Scoring. Es wurden aber die Top 5 referenzierten Gerichtsentscheide sehr niedrig, mit einer 1, bewertet. Kein Gerichtsentscheid schaffte es auf eine 2. Das heisst, beide Ergebnisse fallen schlecht aus.

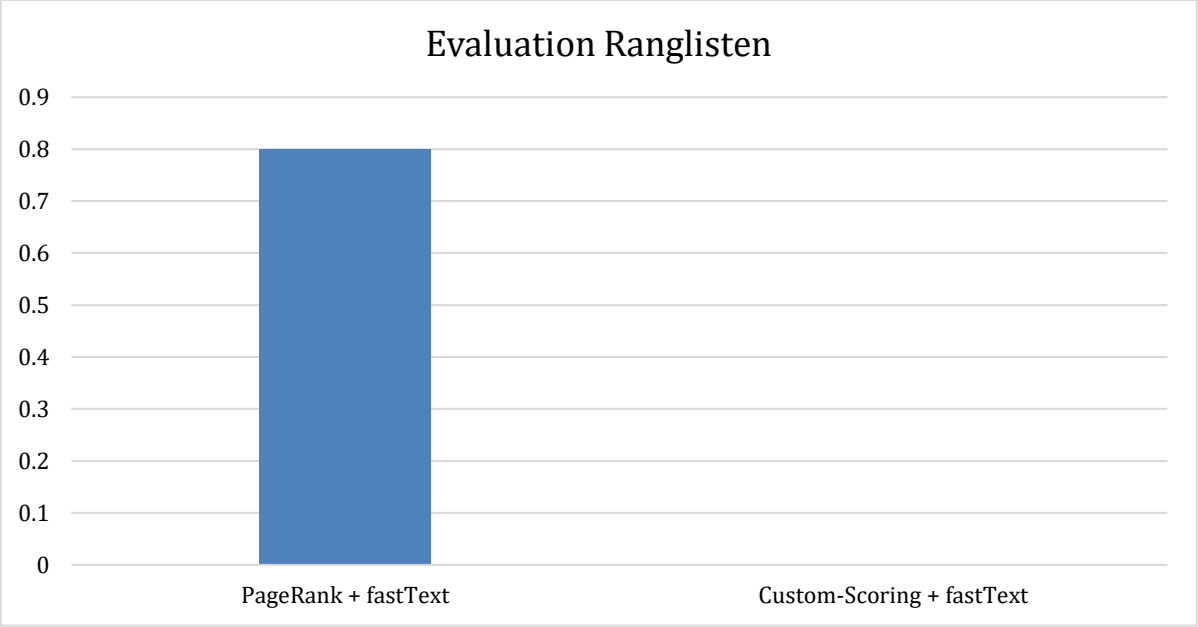


Abbildung 30 Evaluation vom Juristen zu den gemischten Verfahren

10. Diskussion und Ausblick

In dieser Diskussion werden die Evaluation und die Resultate dieser Bachelorarbeit betrachtet und bewertet. Es wird geschaut, ob die Aufgabenstellung erfüllt wurde. Am Ende wird noch ein Ausblick zum Potenzial dieser Arbeit gegeben.

Die Resultate sprechen eine deutliche Sprache. Der Jurist hat für fastText mit Skipgram die beste Bewertung im Vergleich zu den anderen aufgeführten Verfahren abgegeben. Das wurde von uns auch so erwartet, weil mithilfe von Word-Embedding die Texte und ihre Vektoren zuverlässig verglichen werden können. Die Resultate und Ergebnisse wurden anhand weniger Testfälle ausgeführt. Man könnte noch mehr Testfälle vom Juristen erstellen lassen und diese wieder mithilfe der Kendall-Scores mit den unterschiedlichen Verfahren vergleichen. Auch kann man beim Implementieren in Lawsearch eine Funktion einbauen, bei der der User die Suchresultate bewerten kann. Eine weitere Variante wäre, die Ranglisten von mehreren Juristen bewerten zu lassen.

Was leider enttäuscht hat, ist die Netzwerkanalyse mit PageRank und Custom-Scoring. Beide Verfahren haben schlecht abgeschnitten. Das kann darauf zurückgeführt werden, dass innerhalb eines Gerichtsentscheids zu wenig andere Gerichtsentscheide verlinkt werden. Auch sind die verlinkten Gerichtsentscheide in den meisten Fällen gar nicht ähnlich. Meistens werden in einem verlinkten Gerichtsentscheid nur einzelne Sätze verlinkt und im Gerichtsentscheid besprochen, nicht ganze Gerichtsentscheide. Die Vermutung, dass die zweite Baseline Gesetzestexte aus Kapitel 9.2 ein gutes Resultat zeigen würde, konnte sich nicht bewahrheiten. Wie bei den Gerichtsentscheiden, liegt es auch hier daran, dass zu viele Gerichtsentscheide auf zu wenige Gesetzestexte verweisen.

Beide Verfahren haben eine hohe Relevanz. Jetzt ist es möglich, sich für ein Verfahren zu entscheiden, welches die besten Resultate liefert, um dieses noch weiter zu verbessern. Die Aufgabenstellung wurde somit erreicht. Es wurden mehrere Verfahren implementiert, um die Besten auszuwählen und von einem Juristen bewerten lassen zu können.

Diese Arbeit und der dabei entwickelte Prototyp können nun von Weblaw genutzt werden, um den Prototyp in Lawsearch zu integrieren und dem Benutzer die Möglichkeit zu geben, von einem Dokument zum nächsten Dokument zu gehen. Auch bieten die Resultate von fastText

eine sehr gute Möglichkeit, die aktuelle Volltextsuche mit einer neuen Suche zu ergänzen, bei der der Benutzer einen neuen Text verfassen kann. Es werden dann ähnliche Texte zu diesem neu verfassten Text angezeigt.

Diese Bachelorarbeit und die erstellten Resultate geben Lawsearch einen sehr grossen Vorsprung in der Auswahl der passenden Algorithmen. Auch wurde ein Prototyp programmiert, welcher in Lawsearch eingebaut werden kann. Zudem bietet diese Arbeit grosses Potenzial für die Zukunft. Es ist der erste Schritt, um eine intelligente Suche für Anwälte und später dann auch für Kunden zu ermöglichen.

Appendix

Im Anhang wurden die Dokumente eingefügt, welche für diese Bachelorarbeit relevant sind. Das Projektvorgehen beschreibt, wie wir uns während der Arbeit an der Bachelorarbeit organisiert haben und wie wir die Arbeit angegangen sind.

A Projektvorgehen

Die Organisation der Aufgaben wurde über Taiga vorgenommen. Taiga ist ein Ticket-Management-System mit einer Kanban-Board-Oberfläche. Es wurde an der gesamten Bachelorarbeit gemeinsam vor Ort gearbeitet, jeweils am Wochenende und manchmal auch unter der Woche. Alle Entscheidungen wurden ausdiskutiert und gemeinsam getroffen. Es wurden wöchentliche Meetings mit den Betreuern Mark Cieliebak und Don Tuggener abgehalten, wo die nächsten Schritte besprochen wurden. Es gab auch mehrere Meetings mit Blaise Dévaud und Christian Sprecher von der Firma Weblaw AG um die zur Verfügung gestellten Informationen zu besprechen und die Wünsche für die Implementierung.

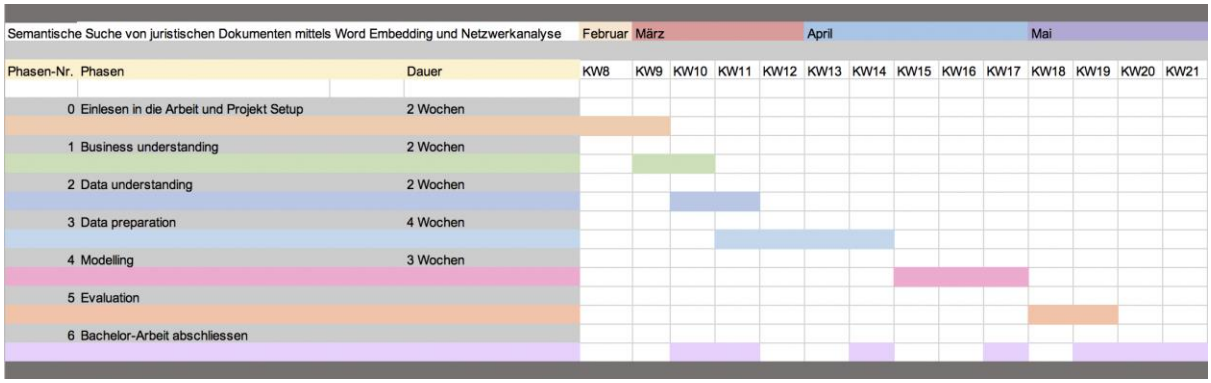


Abbildung 31 Projektplan

In der Phase 0 haben wir uns in das Thema Word Embedding und Netzwerkanalyse eingelesen. Dann ging es darum die aktuelle Software der Firma Weblaw AG und ihre Restriktionen zu verstehen. Gemeinsam mit der Firma Weblaw AG wurde eine neue Aufgabenstellung erarbeitet. Auch wurden und Gerichtsentscheide zur Verfügung gestellt, welche wir in Phase 2 analysiert haben. Da es ca. 800‘000 Dokumente waren, haben wir viel Aufwand auf uns genommen um die Daten aufzubereiten, um dann eine kleine Menge an Daten auszuwählen. Beim Modelling in Phase 4 wurden beide verfahren, die Netzwerkanalyse und das Word

Embedding implementiert. In der Evaluation-Phase 5 wurden die Resultate analysiert. Es wurden Experimente mit dem Juristen zusammen durchgeführt. Immer wieder wurden während jeder Phase einige Passagen Dokumentation geschrieben. Am Ende musste dann nur noch die Bachelorarbeit finalisiert werden. Die Bachelorarbeit durchlief mehrere Korrekturen von Personen, welche die Arbeit durchgelesen haben und ein Wissenschaftliches Lektorat.

Während der gesamten Bachelorarbeit gab es keine grösseren Schwierigkeiten. Die Datenaufbereitung wurde ein wenig unterschätzt. Am Ende wurden sehr aussagekräftige Resultate erstellt, ein lauffähiger Prototyp programmiert, und sehr viel gelernt. Diese Bachelorarbeit bietet einen sehr grossen Mehrwert für die Firma Weblaw AG.

B Gerichtsentscheid Muster-Testfall

Der komplette Gerichtsentscheid, auf welchen in dieser Bachelorarbeit referenziert wird, wird hier im Anhang gezeigt.

Gerichtsurteil-Titel: BGer 6S.522/2006
Der komplette Gerichtsentscheid ist auch auf der Webseite von Weblaw zu finden:
https://entscheide.weblaw.ch/cache.php?link=08.02.2007_6P.2272006&sel_lang=de

Urteil vom 8. Februar 2007
Kassationshof

Besetzung
Bundesrichter Schneider, Präsident,
Bundesrichter Wiprächtiger, Mathys,
Gerichtsschreiber Borner.

Parteien
Z._____,
Beschwerdeführer,
vertreten durch Rechtsanwalt Dr. Peter F. Siegen,

gegen

Staatsanwaltschaft des Kantons Aargau,
Frey-Herosé-Strasse 12, Wielandhaus, 5001 Aarau,
Obergericht des Kantons Aargau, Strafgericht,
3. Kammer, Obere Vorstadt 38, 5000 Aarau.

Gegenstand
6P.227/2006
Strafverfahren; Willkür, Treu und Glauben,

6S.522/2006
SVG-Widerhandlung,

Staatsrechtliche Beschwerde (6P.227/2006) und Nichtigkeitsbeschwerde (6S.522/2006) gegen das Urteil des Obergerichts des Kantons Aargau, Strafgericht, 3. Kammer, vom 16. Oktober 2006.

Sachverhalt:

A.
Z._____ musste sich am 22. März 2001 um 19.30 Uhr in Unterehrendingen einer polizeilichen Kontrolle unterziehen, nachdem er zuvor seinen Personenwagen gelenkt hatte. Die Blutprobe von 20.15 Uhr ergab eine Blutalkoholkonzentration von 2,11 - 2,33 Promille.

B.
Das Bezirksamt Baden verurteilte Z._____ am 11. April 2002 wegen Fahrens in angetrunkenem Zustand zu einer unbedingten Gefängnisstrafe von 30 Tagen und Fr. 1'800.-- Busse. Gleichzeitig widerrief es den bedingten Strafvollzug einer Gefängnisstrafe von 24 Tagen, die das Bezirksamt am 19. Juli 2000 ausgesprochen hatte.

Gegen diesen Strafbefehl erhob der Verurteilte Einsprache mit dem Antrag, er sei wegen Unzurechnungsfähigkeit freizusprechen. Gestützt auf ein psychiatrisches Gutachten sprach das Bezirksgericht Baden Z._____ am 17. Juni 2004 von Schuld und Strafe frei und trat auf den Widerrufsantrag nicht ein.

Auf Berufung der Staatsanwaltschaft des Kantons Aargau sprach das Obergericht des Kantons Aargau Z._____ am 16. Oktober 2006 schuldig des fahrlässigen Führens eines Motorfahrzeugs in angetrunkenem Zustand gemäss Art. 12 StGB i.V.m. Art. 91 Abs. 1 SVG und bestrafte ihn mit 20 Tagen Haft und einer Busse von Fr. 1'000.--. Den Vollzug der Freiheitsstrafe schob das Gericht zugunsten einer ambulanten Therapie auf. Hinsichtlich der früheren Freiheitsstrafe verlängerte es die Probezeit um ein Jahr.

C.
Z._____ führt staatsrechtliche Beschwerde und Nichtigkeitsbeschwerde und beantragt, der angefochtene Entscheid sei aufzuheben.

Das Obergericht und die Staatsanwaltschaft haben auf Gegenbemerkungen verzichtet.

Das Bundesgericht zieht in Erwägung:

1.
Das angefochtene Urteil ist vor dem Inkrafttreten des Bundesgesetzes vom 17. Juni 2005 über das Bundesgericht (Bundesgerichtsgesetz, BGG; SR 173.110) am 1. Januar 2007 ergangen. Auf das Rechtsmittel dagegen ist noch das bisherige Verfahrensrecht anwendbar (Art. 132 Abs. 1 BGG, e contrario), hier somit dasjenige der eidgenössischen Nichtigkeitsbeschwerde nach Art. 268 ff. BStP und der staatsrechtlichen Beschwerde gemäss Art. 84 ff. OG. Am 1. Januar 2007 ist auch der revidierte Allgemeine Teil des Strafgesetzbuches in Kraft getreten. Die neuen Bestimmungen sind hier aber noch nicht von Bedeutung, da das Bundesgericht im Verfahren der Nichtigkeitsbeschwerde nur prüft, ob das kantonale Gericht das eidgenössische Recht richtig angewendet habe (Art. 269 Abs. 1 BStP), mithin das Recht, welches im Zeitpunkt der Ausfällung des angefochtenen Urteils noch gegolten hat (BGE 129 IV 49 E. 5.3 S. 51 f., mit Hinweisen). I. Staatsrechtliche Beschwerde

2.
Die Vorinstanz geht davon aus, der Beschwerdeführer sei aufgrund der Folgen eines Schädelhirntraumas von 1999 und der akuten Alkoholintoxikation zum Zeitpunkt der Tat unzurec nungsfähig gewesen. Sie wirft ihm jedoch vor, er hätte bei der gebotenen Sorgfalt vorhersehen können, dass er nach dem Alkoholkonsum noch Auto fahren werde. In rechtlicher Hinsicht

nimmt sie dementsprechend eine fahrlässige actio libera in causa im Sinne des Art. 12 StGB an.

An der erstinstanzlichen Verhandlung habe der Beschwerdeführer ausgesagt, er kümmere sich um den Hund, füttere die Fische, besorge die Pflanzen und mache den Haushalt, soweit es gehe. Zudem sei es ein automatischer Vorgang, ins Auto zu steigen, um mit dem Hund "Gassi" zu gehen. Er sei bereits einmal wegen Fahrens in angetrunkenem Zustand zu einer bedingten Gefängnisstrafe verurteilt worden. Damit sei für ihn vor Trinkbeginn - als er noch zumindest teilweise zurechnungsfähig gewesen sei - vorhersehbar gewesen, dass er auch an jenem Tag seinen Hund mit dem Auto "Gassi" führen und somit nach dem Alkoholkonsum in angetrunkenem Zustand fahren würde.

2.1 Der Beschwerdeführer wirft der Vorinstanz willkürliche Beweiswürdigung vor. Er sei nicht dazu befragt worden, ob, wann und wie er für gewöhnlich den Hund "Gassi" führe. Die Vorinstanz stütze sich bei ihrer Feststellung auf Aussagen, die er anlässlich der erstinstanzlichen Hauptverhandlungen und bei der psychiatrischen Begutachtung gemacht habe. Diese hätten sich jedoch auf den Ablauf des fraglichen Tages bezogen. Da ihm vorgeworfen werde, er hätte schon am Morgen voraussehen müssen, dass er am Abend den Hund mit dem Auto zum "Versäuberungsplatz" bringen würde, hätte darüber ein Beweisverfahren durchgeführt werden müssen. Dazu sei es aber nicht gekommen. Seine Äusserungen im fraglichen Zusammenhang würden nicht auf eine allgemeine Gewohnheit hinweisen. Namentlich die Aussage, es sei eine automatische Tätigkeit, in das Auto zu steigen, um "Gassi" zu gehen, heisse nicht, er habe das Auto jeden Abend zu diesem Zweck benützt. Diese Antwort mache vielmehr nur klar, wie es möglich gewesen sei, trotz seiner Unzurechnungsfähigkeit mit dem Auto von zuhause wegzufahren.

2.2 Vor dem Bezirksgericht sagte der Beschwerdeführer aus, er habe am fraglichen Tag mit dem Zug nach Zürich fahren wollen. Nach dem Einsteigen habe er an seinen Hund gedacht, weshalb er retour gefahren sei, wobei er vorher in Baden noch etwas gegessen und getrunken habe. Dann sei er nach Hause gegangen, wo dann der Hund einmal "angegeben" habe, dass er hinaus müsse. Er habe die Schlüssel im Schlafzimmer geholt. Es sei ein automatischer Vorgang, ins Auto zu steigen, um mit dem Hund raus zu gehen. Dem psychiatrischen Gutachter gegenüber erwähnte der Beschwerdeführer, er habe sich unterwegs im Zug anders en schieden und sei nach Baden in ein Restaurant gegangen, um zu essen. Dort habe er einen

Aperitif und Wein und später noch einen Kaffee-Schnaps getrunken. Anschliessend habe er an seinen Hund gedacht und sei mit dem Bus nach Unterehrendingen zurückgekehrt. Zuhause habe er Fernsehen geschaut und möglicherweise noch 1 bis 2 Bier getrunken. Es sei dann schon ca. 18.00 oder 19.00 Uhr gewesen. Dies sei wohl die "Gassi"-Zeit für den Hund gewesen, so dass er automatisch das Auto genommen und mit dem Hund in die Nähe eines Friedhofs gefahren sei.

Aus diesen Äusserungen durfte das Obergericht willkürfrei den Schluss ziehen, der Beschwerdeführer habe regelmässig am Abend das Auto benützt, um mit dem Hund "Gassi" zu gehen, weshalb es für ihn auch vorhersehbar gewesen sei, dass er am fraglichen Tag noch sein Motorfahrzeug benützen würde. Willkür in der Beweiswürdigung liegt nämlich nur vor, wenn diese offensichtlich unhaltbar ist, mit der tatsächlichen Situation in klarem Widerspruch steht, auf einem offenkundigen Fehler beruht oder in stossender Weise dem Gerechtigkeitsgedanken zuwiderläuft (BGE 131 I 57 E. 2 mit Hinweis). Davon kann im vorliegenden Fall nicht gesprochen werden. Im Einklang mit der Schlussfolgerung des Obergerichts stehen im Übrigen auch die Angaben des Beschwerdeführers in der Berufungsantwort, er sei um 18.45 Uhr vom Spaziergang zurückgekehrt und danach zu einem unbekannten Zeitpunkt zur Fahrt aufgebrochen. Man müsse davon ausgehen, dass dies wenigstens eine Stunde vor der Heimkehr gewesen sei, was der üblichen Zeit zum "Gassimachen" mit Autofahrt und Spaziergang entspreche.

2.3 Soweit der Beschwerdeführer den Sachverhalt im obergerichtlichen Urteil ergänzt, ohne darzutun, auf welche Anhaltspunkte er sich abstützt und inwieweit sich das Obergericht in willkürlicher Weise darüber hinweggesetzt haben soll, ist auf die Beschwerde nicht einzutreten (Art. 90 Abs. 1 lit. B OG).

3.
Die staatsrechtliche Beschwerde ist nach dem Gesagten abzuweisen, soweit darauf eingetreten werden kann. Bei diesem Ausgang des Verfahrens hat der Beschwerdeführer die bundesgerichtlichen Kosten zu tragen (Art. 156 Abs. 1 OG). II. Eidgenössische Nichtigkeitsbeschwerde

4.

Der Beschwerdeführer wirft der Vorinstanz vor, übertrieben hohe Anforderungen an seine Sorgfaltspflicht zu stellen. Er habe in nüchternem Zustand und "nur" zufolge seiner gesundheitlichen Beschwerden (Schädelhirntrauma mit Frontalhirnsyndrom und Korsakow-Syndrom) in verminderter Zurechnungsfähigkeit nicht damit rechnen müssen, am Abend angetrunken Auto zu fahren. Dies habe er an diesem besonderen Tag auch und gerade deshalb nicht annehmen müssen, weil er den Hund am Vormittag mit sich nach Zürich habe mitnehmen wollen. Auf dem geplanten Tagesausflug hätte der Hund ohne weiteres und mehrmals die Möglichkeit gehabt, seine Notdurft zu verrichten. Es verletze Bundesrecht, wenn der Kreis der zu beachtenden Sorgfaltspflichten derart weit gezogen werde. Er dürfe nicht bestraft werden, weil er den Hund zuhause vergessen habe und weil er daher am Abend in bereits vollständig unzurechnungsfähigem Zustand den Hund, der den ganzen Tag im Haus gewesen sei, mit dem Auto zum "Gassimachen" geführt habe. Unter solchen Umständen könne nicht von einer actio libera in causa gesprochen werden. Die vorinstanzliche Auslegung von Art. 12 StGB führe dazu, dass letztlich immer mit der Möglichkeit einer Autofahrt gerechnet werden müsse. Dies könne nicht die Meinung des Gesetzgebers sein. Von einer actio libera in causa könne nicht schon die Rede sein, wenn irgendeine noch so entfernte Möglichkeit einer späteren Autofahrt bestehe, sondern nur, wenn die konkrete Möglichkeit dazu bestehe oder wenn wenigstens mit einer überwiegenden Wahrscheinlichkeit damit gerechnet werden müsse.

4.1 Gemäss Art. 12 StGB sind die Bestimmungen der Art. 10 und 11 StGB über die Unzurechnungsfähigkeit bzw. die verminderte Zurechnungsfähigkeit nicht anwendbar, wenn die schwere Störung oder die Beeinträchtigung des Bewusstseins vom Täter selbst in der Absicht herbeigeführt wurde, in diesem Zustande die strafbare Handlung zu verüben. Das Gesetz umschreibt damit die vorsätzliche so genannte actio libera in causa (d.h. das verantwortliche Ingangsetzen des Geschehensablaufs). Der Grundsatz ist aber auch anwendbar bei der fahrlässigen actio libera in causa: Die Verminderung der Zurechnungsfähigkeit bzw. deren gänzlicher Ausschluss ist unbeachtlich, wenn der Täter in diesem Zustand eine fahrlässige Straftat begeht und die Tat für ihn zur Zeit, als er noch voll zurechnungsfähig war, bei pflichtgemässer Aufmerksamkeit voraussehbar war (BGE 117 IV 292 E. 2 mit Hinweisen).

4.2 Aufgrund des verbindlichen Sachverhalts entsprach es einer Gewohnheit des Beschwerdeführers, jeweils am Abend seinen Hund mit dem Auto "Gassi" zu fahren. Dass es am fraglichen Tag hätte anders sein sollen, stellt die Vorinstanz nicht fest. Gerade weil der Beschwerdeführer - wie er selbst vorbringt - den Hund vergessen hatte und dieser zuhause bleiben

musste, war es naheliegend, dass er auch an diesem Abend mit dem Hund "Gassi" fahren würde. Diese Wahrscheinlichkeit lag für den Beschwerdeführer auf der Hand und war für ihn trotz seiner gesundheitlichen Beeinträchtigung erkennbar, als er zu trinken begann. Indem er sich darüber hinwegsetzte, verletzte er seine Sorgfaltspflicht. Die Vorinstanz hat somit zu Recht Fahrlässigkeit bejaht und den Beschwerdeführer in Anwendung von Art. 12 StGB des Fahrens in angetrunkenem Zustand im Sinne von Art. 91 Abs. 1 SVG schuldig erklärt.

4.3 Die Nichtigkeitsbeschwerde ist deshalb abzuweisen. Bei diesem Ausgang des Verfahrens trägt der Beschwerdeführer die bundesgerichtlichen Kosten (Art. 278 Abs. 1 BStP).

Demnach erkennt das Bundesgericht:

1.
Die staatsrechtliche Beschwerde wird abgewiesen, soweit darauf einzutreten ist.

2.
Die Nichtigkeitsbeschwerde wird abgewiesen.

3.
Die Gerichtsgebühr von insgesamt Fr. 4'000.-- wird dem Beschwerdeführer auferlegt.

4.
Dieses Urteil wird dem Beschwerdeführer, der Staatsanwaltschaft des Kantons Aargau und dem Obergericht des Kantons Aargau, Strafgericht, 3. Kammer, schriftlich mitgeteilt. Lausanne, 8. Februar 2007 Im Namen des Kassationshofs des Schweizerischen Bundesgerichts Der Präsident: Der Gerichtsschreiber:

C Codedokumentation

Die Codedokumentation wurde jeweils innerhalb der Jupyter-Notebooks beschrieben.

D Softwarecode und Verzeichnis der abgelegten Daten

Der Softwarecode wurden jeweils in Python geschrieben und in Jupyter-Notebooks auf dem Datenträger abgelegt. Auch wurde ein Verzeichnis erstellt, der beschreibt welche Dokumente der Datenträger beinhaltet.

E Verwendung des Prototyps

In dieser Sektion werden zuerst für das Verwenden des Prototyps die Voraussetzungen beschrieben. Danach wird ein kleiner Guide beschrieben, wie der Prototyp verwendet werden kann.

E.1 Betriebssystem

Es werden grundsätzlich bei allen verwendeten Komponenten (oder Software-Paketen) alle modernen Betriebssysteme (Windows, macOS, Linux) unterstützt.
Bei dieser Bachelorarbeit hat sich erwiesen, dass beim Linux (Ubuntu 16.04) das Installieren benötigter Software-Pakete oder Python-Module am einfachsten war. Daher wird empfohlen für den Prototyp ein Linux-System zu verwenden.

E.2 Software

Folgende Software-Pakete sollten installiert werden:

- Anaconda (Version: > 5.0)
- Elasticsearch (Version: 6.2)
- fastText (Version: 0.1)
- Python (Version: 3.6.4)

Nach der Installation von Anaconda sollte sichergestellt werden, dass Jupyter Notebook mitinstalliert wurde. Jupyter Notebook wird gebraucht, um die IPython-Notebook Skripte auszuführen.

E.3 Python-Module

Folgende Python-Module wurden in den Skripts verwendet.

- networkx (Version 2.1)
- pyfasttext (Version 0.4.4)
- justext (Version 1.4)
- xmltodict (Version 0.11.0)
- beautifulsoup (Version 4.6.0)
- xmljson (Version 0.1.9)
- pandas (Version 0.22.0)
- numpy (Version 1.14.0)
- elasticsearch (Version 6.2.0)
- sklearn
- fnmatch

E.4 Verzeichnisstruktur

Die Verzeichnisstruktur für die Skripts sieht folgendermassen aus:

- gpickle_files: Hier sollten die Graphen in gpickle-Format abgespeichert werden und ausgelesen werden
- pickle_files: Hier sollten gewisse Daten wie, Gerichtsentscheide-Titel abgespeichert werden und ausgelesen werden.
- models: Hier sollten die fastText-Modelle nach dem Trainieren abgespeichert werden und für Berechnungen wieder ausgelesen werden.

Ausserhalb dieser Verzeichnisse sollten sich die IPython-Notebooks befinden, welche auf diese Verzeichnisse zugreifen.

E.5 Verwendung der Skripte

Um die fastText-Modelle und die Graphen zu erhalten, müssen die Skripts in einer bestimmten Reihenfolge ausgeführt werden. Die einzelnen Skripts können je nach dem auch einzeln ausgeführt werden. Spezifische Informationen zu den Skripts, wie benötigte Parameter und Rückgabewerte für einzelne Funktionen, können in den IPython-Notebooks gefunden werden.

Für die Verwendung der Skripte werden natürlich die richtigen Daten vorausgesetzt. Die Skripts sind so geschrieben, dass sie grundsätzlich nur die XML-Dokumente von Weblaw verwenden können. Für die Verwendung anderer Dokumente müssen in den Skripts mehrere Veränderungen vorgenommen.

Es wird nun Schritt für Schritt kurz erklärt, wie die fastText-Modelle und Graphen anhand der Skripts kreiert werden können. Die Skripts, welche für die diese Schritte in Frage kommen, werden in der Tabelle 8 aufgelistet.

1. Die Daten müssen zuerst vorverarbeitet werden.
    a. Alle deutschen Dokumente müssen extrahiert werden
    b. Es müssen alle Gerichtsentscheide extrahiert werden
    c. Aus den XML-Dokumenten sollen die wichtigsten Tags und ihre Inhalte, vorverarbeitet herausfiltriert werden und in neue JSON-Files abgespeichert werden.
    d. Die JSON-Files sollten in Elasticsearch indexiert werden.
    e. Sind mehr Gesetzesentscheide gewünscht, können Texte von openjur.de bezogen werden.
2. Eine oder mehrere fastText-Modelle sollten trainiert werden.
3. Eine Adjazenzliste sollte kreiert werden.
4. Ein Multigraph sollte anhand der Adjazenzliste und des fastText-Modells kreiert werden.

Skriptname	Schritt
extarct_german_documents.ipynb	1a
partitioning_extracting_documents.ipynb	1b
xml_to_json_conversion.ipynb	1c
elasticsearch_bulk_upload.ipynb	1d
openjur_webcrawler_text_extraction.ipynb	1e
fasttext_training.ipynb	2
create_adjacency_list.ipynb	3
create_networkx_graph.ipynb	4

Tabelle 8 Skriptname und zugehörige Schritte

Nach der Ausführung dieser Schritte können nun die IPython-Notebooks mit den Hauptfunktionalitäten ausgeführt werden.

Mit `network_analysis.ipynb` kann nun für einen gegebenen Gerichtsentscheid die relevantesten Nachbargerichtsentscheide gefunden werden.

Mit `find_similiar_textes.ipynb` kann für einen neuen Sachverhalt die ähnlichsten Sachverhalte oder Gerichtsentscheide (Top 10) gefunden werden.

Glossar

Hier werden die Begriffe erklärt, die in der Arbeit nicht beschrieben werden.

Begriff	Erklärung
Jurist	Sobald man das Studium in der Rechtswissenschaft abgeschlossen hat, ist man Jurist.
Indexieren	Bei der Indexierung werden Schlagworte zu einem Dokument zugeordnet. Damit lassen sich dann Sachverhalte erschliessen.
Bag of Words	Ein Bag of Words ist ein Set der im Text enthaltenen Wörter. Zusätzlich wird die Anzahl gezählt, wie oft ein Wort im Dokument vorkommt.
Regeste	Mit Hilfe der Regeste wird der rechtsrelevante Inhalt zusammengefasst.
Overfitting	Mit Overfitting oder Überanpassung ist die Klassifizierung eines Modells gemeint, welches zu viele erklärende Variablen enthält. Dieses Modell passt nur auf einen bestimmten Satz an Daten.
Korpus	Ein Korpus ist eine Sammlung von Dokumenten.
Score	Anzahl Punkte als Mass für ein Verfahren. Zum Beispiel sagt eine hohe Anzahl Punkte aus, dass das Verfahren gut ist.
Webcrawler	Mit dem Webcrawler können automatisch Inhalte aus Webseiten analysiert und extrahiert werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] T. K. E. G. L. E. W. K. M. S. P. B. Karl Moritz Hermann, Teaching Machines to Read and Comprehend, Montreal, Canada, 2015.

[2] T. F. Foster Provost, Data Science for Business, O'Reilly Media, 2013.

[3] S. B. Lawrence Page, The PageRank Citation Ranking: Bringing Order to the Web, 1999.

[4] P. N. Stuart J. Russell, Artificial Intelligence: A Modern Approach, Prentice Hall, 2009.

[5] M. Newman, Networks: An Introduction, OUP Oxford, 2010.

[6] C. Zhai, „coursera," Coursera, [Online]. Available: https://www.coursera.org/learn/text-retrieval. [Zugriff am März 2018].

[7] „Bundesverwaltungsgericht (Schweiz)," Wikipedia, 18 Mai 2018. [Online]. Available: https://de.wikipedia.org/wiki/Bundesverwaltungsgericht_(Schweiz). [Zugriff am Mai 2018].

[8] „Bundesgericht (Schweiz)," Wikipedia, 27 Mai 2018. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Bundesgericht_(Schweiz)&oldid=177787886. [Zugriff am Mai 2018].

[9] „Bundesstrafgericht," Wikipedia, 18 Mai 2018. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Bundesstrafgericht&oldid=177523169. [Zugriff am Mai 2018].

[10] „Apache Lucene," Wikpedia, 17 Oktober 2017. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Apache_Lucene&oldid=170070276. [Zugriff am Mai 2018].

[11] „Graph (Graphentheorie)," Wikipedia, 28 Mai 2018. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Graph_(Graphentheorie)&oldid=177818798. [Zugriff am Mai 2018].

[12] „unternehmer.de," unternehmer.de, [Online]. Available: https://www.unternehmer.de/lexikon/online-marketing-lexikon/pagerank. [Zugriff am Mai 2018].

[13] „Word embedding," Wikipedia, 29 Mai 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Word_embedding&oldid=843466192. [Zugriff am Mai 2018].

[14] „fastText - Word representations,“ Facebook Open Source, 2018. [Online]. Available: https://fasttext.cc/docs/en/unsupervised-tutorial.html. [Zugriff am Mai 2018].

[15] „Kosinus-Ähnlichkeit,“ Wikipedia, 24 Juni 2017. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Kosinus-%C3%84hnlichkeit&oldid=166682124. [Zugriff am Mai 2018].

[16] C. S. Perone, „Terra Incognita,“ 12 September 2013. [Online]. Available: http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/. [Zugriff am Mai 2018].

[17] „Tf-idf-Maß,“ Wikipedia, 29 März 2018. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Tf-idf-Ma%C3%9F&oldid=175531817.

[18] „http://www.tfidf.com/,“ [Online]. Available: http://www.tfidf.com/. [Zugriff am Mai 2018].

[19] „DL4J,“ Skymind, 2017. [Online]. Available: https://deeplearning4j.org/bagofwords-tf-idf. [Zugriff am Mai 2018].

[20] R. Agrawal, „Memento,“ [Online]. Available: https://ragrawal.wordpress.com/2013/01/18/comparing-ranked-list/. [Zugriff am Mai 2018].

[21] A. M. Z. William Webber, A similarity measure for indefinite rankings, New York: ACM, 2010.

[22] „iso25000,“ iso25000, 2018. [Online]. Available: http://iso25000.com/index.php/en/iso-25000-standards/iso-25010. [Zugriff am Mai 2018].

[23] „openJur,“ openJur, 2018. [Online]. Available: https://openjur.de/. [Zugriff am Mai 2018].

[24] „elastic,“ elastic, 2018. [Online]. Available: https://www.elastic.co/de/products. [Zugriff am Mai 2018].

[25] „Kibana,“ Wikipedia, 26 Januar 2018. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Kibana&oldid=173344015. [Zugriff am Mai 2018].

[26] „Adjacency list,“ Wikipedia, 6 März 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Adjacency_list&oldid=829050446. [Zugriff am Mai 2018].

[27] „NetworkX,“ NetworkX , 2018. [Online]. Available: https://networkx.github.io/. [Zugriff am Mai 2018].