Projekt
Research Knowledge Discovery

Detektion von Experten und Aufbau eines Recommender-Systems für die TH Köln

30.06.2021

Projektgruppe 2 - DIS - 18

Autoren

Pia Störmer

Matteo Meier

Jüri Keller

Sascha Gharib

Martin Bilko

Verena Pawlas

Saskia Brech

Michelle Reiners

Fabian Ax

Constantin Krah

Leon Munz

Andreas Kruff

Fabian Gitzler

Jonas Dudda

Annika Füssel

Inhaltsverzeichnis

1. Einleitung

Im Rahmen des Projekts Research Knowledge Discovery, haben wir Studierende die Basis für ein Recommender System aufgebaut. Dieses hat zum Ziel externen Forschungsinteressierten, wie z.B. Firmen und Forschern zu helfen Experten zu Themengebieten zu finden.
Für interne Forschende der Technischen Hochschule Köln (TH Köln) und externe Kooperationspartner ist es nicht immer leicht die Kompetenzen einer Einrichtung wie der TH Köln zu erfassen. Momentan sind Experten der TH Köln über organisatorische Einheiten wie Fakultäten, Institute oder Studiengänge aufzufinden. In Situationen, in denen das Informationsbedürfnis noch nicht klar erfasst ist, also keine Zuordnung zu einem dieser organisatorischen Bereiche möglich ist, wird eine zielgerichtete Adressierung von Forschungsfragen praktisch unmöglich. Ein Knowledge Discovery System kann helfen eine neue Sichtweise zu schaffen und die vorhandenen Experten durch neue Blickwinkel auffindbar zu machen.

Ansätze zur Erstellung eines solchen Systems lassen sich weitestgehend in drei Kategorien gliedern, Generative Probabilistic Models, Voting Models, Network-Based Models. Die Generative Probabilistic Models ranken die Experten basierend auf der Wahrscheinlichkeit wie sehr sie zu einem Thema passen. Hierzu werden Dokumente die im Zusammenhang mit den Personen stehen als Zeichen für Expertise betrachtet. Voting Models kombinieren mehrere sortierte Listen von Dokumenten wie Stimmen für Experten. Der Experte mit den meisten Stimmen wird abschließend empfohlen. Network-Based Models betrachten vor allem die Verknüpfungen zwischen Experten und Dokumenten. Anhand von Methoden der Netzwerkanalyse können so Experten ermittelt werden die zu empfehlen sind. (Lin et al. 2016)

Alle Ansätze finden Anwendung in unterschiedlichen Bereichen. Für die Entscheidung eines Systems sind unterschiedliche Faktoren zu berücksichtigen. Basierend auf der Anzahl von potentiellen Experten und den zugrundeliegenden Dokumenten haben wir uns hauptsächlich mit den Generative Probabilistic Models beschäftigt.

Das Projekt erstreckte sich über zwei Studiensemester. Zu Beginn befassten wir uns ausführlich mit der Literaturrecherche und dem Erarbeiten eines Konzeptes (Roadmap) (DIS-KD Projekt, 2021). Anschließend wurden die zur Verfügung stehenden Daten der Hochschulwebsite gesammelt und gesichtet (Th-Köln (2021) Personen von A-Z). Im zweiten Semester fokussierten wir den Aufbau des Recommender-Systems. Im Projektverlauf sind wir auf zahlreiche Hindernisse gestoßen, die wir im Folgenden beschreiben und analysieren.

1.1 Projektmanagement & Organisation

Das Projekt wurde von den Teilnehmenden Studierenden vorangehend durch ein „Management Team“ selbst organisiert. Zu den Aufgaben des Teams gehörten Aufgabenverteilung, Zeit- und Projektplanung, Konzeption, Dokumentation und Literaturrecherche.

Darüber hinaus war das Managementteam für die Vorbereitung der wöchentlichen Meetings zuständig und reflektierte im Anschluss den Projektfortschritt, aus dem neue Aufgaben definiert wurden.

Die weiteren Studierenden gründeten Subteams, die sich mit der Umsetzung des programmiertechnischen Teils befassten. Da die Studierenden in den begleitenden Veranstaltungen, neben dem Projektzeitraum noch wichtige Herangehensweisen erlernen, um technischen Herausforderungen der Knowledge Discovery zu begegnen, raten wir dazu die Professoren bei Problemen, die sich über zwei Wochen strecken, um Denkanstöße zu bitten. Andernfalls besteht die Gefahr, dass wertvolle Zeit im Projekt verloren geht (wie bei uns geschehen).

Rollenverteilung

Ein wichtigster Aspekt ist es den Studierenden die Möglichkeit zu geben sich frei entsprechenden Subgruppen zuzuordnen und eine Funktion im Projekt zu übernehmen die

A. ihren fachlichen Kompetenzen entspricht und B. erfahrene Arbeitsgruppen nicht zu trennen. Im Managementteam sollte die Rollenverteilung von Kontinuität geprägt sein und das Aufgabengebiet definiert.

Es bieten sich folgende Rollen für das Orga Team an:

Orga Team

- Moderation (Kommunikation administrieren, Meeting Vor und Nachbereitung, Fortschritt überwachen, proaktiv auf Herausforderungen/Probleme reagieren)

- ggf. Co-Moderation

- Protokollant (Meeting Protokoll, Admin Projektboard & Eintragung Arbeitspakete)

- Technische Leitung (“Qualitätskontrolle” & Unterstützt SubTeams bei Problemen)

Aufgabenverteilung

Im laufenden Projekt machten wir die Erfahrung, dass eine klare Aufgabenverteilung zielführender ist als zwischen den Aufgaben zu wechseln. Das führte dazu, dass wir erst während der Bearbeitung des Projektes eine klare Struktur definierten, die ab dem zweiten Studiensemester innerhalb der Projektgruppe gefestigt Anwendung fand.

Die Moderation ist zuständig für die Inhaltsplanung der Meetings und behält stets den Projektfortschritt im Auge. Der Protokollant ist über das Protokoll hinaus zuständig für das Projektmanagement Board, in dem alle Aufgaben hinterlegt werden. Aus dem beruflichen Umfeld adaptiert, ist zu empfehlen, dass ein Orga-Team Mitglied über die technischen Fähigkeiten verfügt die Fachgruppen bei Problemen zu unterstützen und ggf. die Arbeitspakete auf ihre Qualität zu überprüfen.

Projektmanagement Tools

Im ersten Projektsemester nutzten wir ein Canvas Board mittels Github (DIS-KD-Project. (2021) Kanban Board). Da dies allerdings nicht in der Regelmäßigkeit und Detailtiefe genutzt wurde, haben wir uns im zweiten Semester dazu entschieden auf Basis von Miro das gesamte Projekt abzubilden. Über Visualisierung von Prozessen hinaus lässt sich ebenfalls durch entsprechende Canvas Frames die Zuweisung von Arbeitspaketen ermöglichen. Bei regelmäßiger Pflege eignet sich ein Gesamtheitliches „Projekt-Board“ im studentischen Umfeld besser, als die Adaption von Projektmanagement Methoden wie Scrum (Scrum.org . 2021) oder Prince2 (ILX Group . 2021).

Kommunikation

Um eine regelmäßige Kommunikation zu ermöglichen, ohne Berechtigungseinschränkungen, eröffneten wir einen ZOOM (Zoom. 2021) Raum, der allen Projektteilnehmern jeder Zeit zugänglich war. Als interne schriftliche Kommunikations-Plattform der Studierenden wurde WhatsApp gewählt. Im laufenden Semester lernten wir parallel die Vorzüge von Mattermost (Mattermost Inc. 2021) kennen. Mit dem Wechsel auf die neue Kommunikations-Plattform war es den Studierenden möglich untereinander aber auch mit den entsprechenden Dozenten über einen Kanal zu kommunizieren.

2. Methoden/Projektpipeline

Zur Bewältigung der beschriebenen Problemstellung gibt es verschiedene State-of-the-Art-Ansätze. Für den zugrundeliegenden Use Case wurden zwei unterschiedliche Ansätze in Betracht gezogen: Ad-Hoc-Retrieval und Browsing. Beide Ansätze erzeugen eine sortierte Liste mit Experten passend zu einem Thema. Sie basieren beide auf vermeintlichen Experten, Dokumenten und Themen. Diese stehen alle mit den Kandidaten in Zusammenhang und können vollkommen verschiedenartig sein. Sie dienen als Zeichen für Expertise. Themen sollen das Informationsbedürfnis der suchenden Personen ausdrücken. Um die Themen mit den Experten in Verbindung zu setzen, also das Informationsbedürfnis mit einer Liste von passenden Experten zu verbinden, werden die Dokumente als Proxy eingesetzt.

2.1. Ad-Hoc Retrieval Ansatz

Beim Ad-Hoc-Retrieval wird dem Nutzer ein Suchschlitz zur Verfügung gestellt. Der Nutzer kann wie bei der Google-Suche eine beliebige Suchanfrage absetzen, z.B. nach einem Themengebiet suchen und erhält als Ergebnis die entsprechenden Experten auf diesem Gebiet, in gerankter Reihenfolge. Die vermeintlichen Experten werden dabei nach der Wahrscheinlichkeit gerankt, dass sie Experte auf dem Gebiet sind.

Beim Ad-Hoc Retrieval Ansatz wird das Thema, zu dem eine suchende Person Expertise braucht von ihr selbst definiert. Jeder für die Person relevante Term kann der Suche hinzugefügt werden. Durch Boolesche Operatoren können zudem Terme ausgeschlossen werden und so das Thema weiter spezifiziert werden.
Um die Terme der Suche hinzuzufügen, also das Thema der Suche zu erstellen, bedarf es einem gefestigteren Informationsbedürfnis als beispielsweise beim Browsing Ansatz. Die Suchende Person muss zu Beginn der Suche schon wissen zu welchem Thema sie Expertise braucht. Um so spezifischer die Suche aufgebaut ist, umso spezifischer sind auch die Ergebnisse.
Die Relation zwischen Experten und Thema entsteht beim Ad-Hoc Retrieval Ansatz spontan und wird nicht, wie beim Browsing Ansatz vorberechnet. Anhand der Suchterme werden, wie im klassischen Retrieval, relevante Dokumente geranked. Die vermeintlichen Experten die mit diesen Dokumenten verbunden sind, werden dann extrahiert und auch gerankt. In dieses Experten Ranking können weitere Faktoren mit einfließen. So kann beispielsweise berücksichtigt werden wie viele Dokumente einem vermeintlichen Experten zugeordnet sind, wie viele vermeintliche Experten in einem Dokument genannt werden oder auch wie groß die Distanz zwischen Thema und vermeintlichem Experten im Dokument ist.

Durch die freie Themenwahl der suchenden Person und den Suchschlitz können die Ergebnisse kaum kontrolliert werden. Zudem wird auch ein Ranking erstellt, wenn es lediglich kandidaten mit minimaler Wahrscheinlichkeit gibt. Bei einer überschaubaren Anzahl von Themengebieten oder vermeintlichen Experten eignet sich dieser Ansatz daher schlecht.

2.2. Browsing Modell

Die Idee des Browsing Modells ist es eine hierarchische Abbildung von Themengebieten zu schaffen bzw. Topics zu generieren und dem Nutzer auf einer Website zu präsentieren. Hierbei stellt der Nutzer keine direkte Suchanfrage, sondern „browst“ nach einem gewünschten Thema oder Themengebiet. Dieses Modell kann im Vergleich zum Ad-Hoc-Retrieval vorberechnet werden.

Für die Erstellung eines Browsing Modells ist es notwendig Topics aus einem entsprechenden Dokumentenkorpus zu generieren, welche anschließend hierarchisiert werden können. Eine Möglichkeit zur Extraktion solcher Topics ist es ein Topic-Klassifizierungsmodell zu trainieren, bei welchem die Dokumente aus dem Korpus nach vordefinierten Topics gelabelt werden.. Diese vordefinierten Topics könnten anhand der Forschungsgebiete und Lehrgebiete der TH Köln intellektuell erschlossen und definiert werden. Für die Nutzung eines Klassifizierungsmodells sind jedoch Trainingsdaten erforderlich.

Eine weitere Möglichkeit ist die Nutzung von Topic Modeling. Topic Modeling ist ein generatives Wahrscheinlichkeitsmodell, welches unbekannte Topics aus großen Datenmengen extrahiert und zählt zu den Ansätzen des Unsupervised Learning. Ein oft genutztes Modell ist das Latent Dirichlet Allocation Modell (LDA)**.** LDA betrachtet Dokumente als eine Mischung aus verborgenen Topics (latent topics) und jedes Topic als eine Mischung von Termen. Die Anzahl der Topics wird zu Beginn festgelegt. Da die Dokumente eine Mischung aus verschiedenen Wörtern sind, hat jedes Wort eine Wahrscheinlichkeitsverteilung, die zu den Themen gehört. Daraus ergibt sich die Wahrscheinlichkeitsverteilung der Themen, die zu

jedem Dokument gehören, siehe Abbildung (Blei et al., 2003).

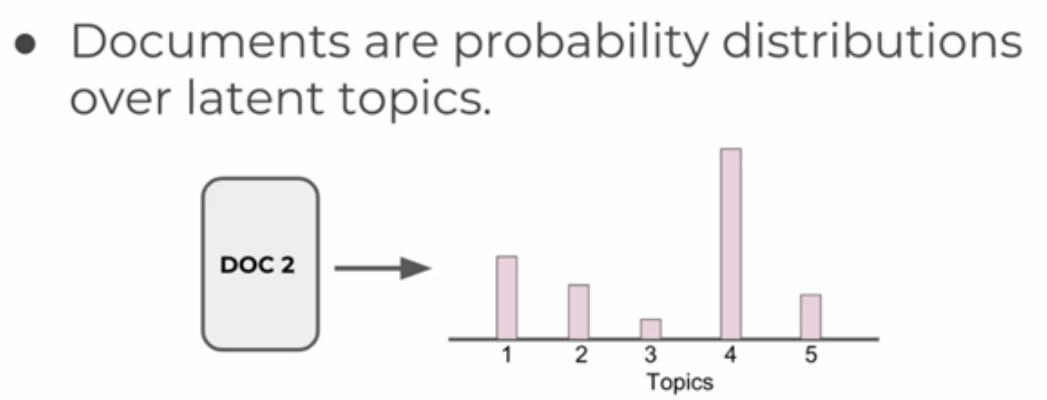


Abbildung X: Documents probabilty distributions,
Quelle: https://www.udemy.com/course/nlp-natural-language-processing-with-python/

2.3. Modell Entscheidung

Vergleicht man die Ansätze miteinander, gibt es einige Gründe das Browsing Modell zu verwenden. Während bei dem Browsing Modell und spezifisch LDA die Topics anhand von Wahrscheinlichkeiten ermittelt werden und somit vorberechnet werden kann, müssen bei der Freitextsuche die Fehler des Nutzers wie Rechtschreibfehler berücksichtigt werden. Außerdem kann es zu einem Wording-Problem kommen, wenn der Nutzer nicht die „richtigen“ Keywords verwendet. Dadurch werden eventuell Dokumente, die eigentlich das Informationsbedürfnis des Nutzers befriedigen sollten, nicht gefunden.

Betrachtet man die Vor- und Nachteile der beiden Ansätze, erschien das Browsing Modell der für unseren Usecase geeignetere Ansatz zu sein. Ausgehend von diesem gewählten Ansatz erwies sich das Topic Modeling, spezifisch der LDA-Ansatz, als besonders geeignet. Dies ist damit zu begründen, dass für die Nutzung eines Topic-Klassifizierungsmodelles Trainingsdaten in großen Mengen erforderlich sind, auf dass das Modell trainiert werden kann. Diese hätten aufwendig vorbereitet werden müssen, was den Zeitrahmen gesprengt hätte.

2.3.1 Wie hängen die unterschiedlichen Teilsysteme zusammen?

Zu Beginn steht die Gewinnung von Daten, welche von der Website der TH Köln stammen. Mithilfe der unter Punkt 3 näher ausgeführten Vorgehensweise konnten von einzelnen Webseiten der TH Köln Dokumente generiert werden, welche mittels Name Matching mit den dort genannten Personen verknüpft wurden.
Diese Dokumente wurden dann durch eine Indexierungspipeline geführt, welche die Inhalte der Dokumente für das LDA Topic Modelling vorbereiten sollte.
Die Topic mithilfe von LDA wurden nur auf den Dokumenten generiert, die auf der Website unter Projekte oder Forschung zu finden sind, oder aber aus PDF-Dateien stammen, die auf der Website verlinkt sind. Daraufhin wurden auch den restlichen Dokumenten Topics zugewiesen.

Nachdem mithilfe des LDA Moduls die unterschiedlichen Topics erstellt wurden, sollten diese intellektuell erschlossen werden. Hierzu wurden zuvor, unter der Berücksichtigung der einzelnen Lehrgebiete, verschiedene Unter- und Oberkategorien gebildet, welche anschließend in einen Thesaurus geordnet werden sollten. Aufgrund des Thesaurus entstand die gewünschte Hierarchie, welche für den Browsing Ansatz vonnöten ist. Die einzelnen Topics wurden den entsprechenden Kategorien zugeordnet, sodass jedes relevante Topic seinen Platz im Thesaurus finden kann.
Die erarbeiteten Oberkategorien sollten demnach innerhalb des Browsing Modells aufgelistet werden. Durch das Anklicken einer Kategorie gelangt man jeweils auf die nächste Unterkategorie. Auf der untersten Ebene der hier beschriebenen Hierarchie sollten dann alle Experten des Themas aufgelistet werden.
Dies sollte dadurch erfolgen, dass jedem Dokument eine Person zugeordnet sein muss. Da die Dokumente zusätzlich mit mindestens einem Topic verknüpft sein müssen, um in das Browsing Modell aufgenommen zu werden, waren demnach auch Personen mit Topics über die Dokumente verknüpft.

3. Erstellung des Datensatzes

Um einen passenden Korpus für ein Experten Retrieval erstellen zu können, wurden mehrere Quellen und Ansätze untersucht und in einem iterativen Prozess stetig verbessert. Als Grundlage diente ein Scrapy Scraper (Zyte, 2008) der Bachelorarbeit „Crawl Your Prof – Fact-Crawling von Hochschulseiten“ (Bischkopf, R., 2020), der die Daten aller im Personenverzeichnis der TH Köln Website stehenden Personen extrahierte.

3.1. BA Scraper

Die Bachelorarbeit von Ruben Bischkopf (2020) beschäftigt sich mit der Sammlung frei verfügbarer Informationen von Angestellten der TH Köln zur Erstellung von Expertenprofilen. Hierzu wird das Personenverzeichnis der Hochschule als Ausgangspunkt verwendet. Diese Seite listet alle Beschäftigten von alphabetisch geordnet, inklusive Links zu den jeweiligen Personen Detailseiten auf. Dabei werden zunächst die ersten 60 Personen angezeigt, wobei sich die Ansicht durch einen Klick auf einen “Mehr Anzeigen”-Button um jeweils zehn Personen erweitern lässt.
Im Rahmen der Bachelorarbeit (Bischkopf, R., 2020) wird der „Mehr Anzeige“-Button automatisiert so oft aufgerufen, bis alle Personen aufgelistet sind.

Parallel werden alle Links extrahiert und die entsprechenden Personen Detailseiten gescraped. Hierzu werden die einzelnen Informationsfelder ausgelesen und in Form von Scrapy Items gespeichert.

Die mithilfe des Python Framework Scrapy (Zyte, 2008) implementierten Methoden zur Extraktion der Personen Detailseiten der TH Köln wurden aus der Bachelorarbeit, für unser Projekt übernommen. Als Output wird eine CSV-Datei (persons.csv) erzeugt, welche eine Liste aller Angestellten, inklusive aller auf den jeweiligen Personen Detailseiten gefüllten Informationsfelder, enthält. Diese Datei dient als Ausgangspunkt für das Name-Matching, das für die weitere Datenbeschaffung implementiert wurde.

3.2. Interne Quellen

3.2.1. Ansatz Scrapy

Der zur Verfügung gestellte Scraper der Bachelorarbeit wurde mit dem Python Framework Scrapy erstellt, was als Grundlage der Erweiterung um einen Crawler diente, welcher die gesamte TH Seite crawlen und scrapen kann.

Datenqualität – Im ersten Durchlauf wurde der Crawler lediglich mit der Einschränkung auf die th-koeln.de Domain beschränkt. Das resultierte darin, dass der überwiegende Teil der gescrapten Seiten keinen oder irrelevanten Inhalt hatte. Dies zeigte sich daran, dass diese leer, Suchschlitz-URLs, Online-Formulare, Dateien oder Unterseiten der TH waren. Vor allem die Suchschlitz-URLs führten den Crawler schnell in eine Umgebung von nicht brauchbaren Seiten, da jegliche Kombination von Parametern sowie vorhandene Suche Unterstützungen als eigene URL gescrapt wurde.

(Beispiel: www.th-koeln.de/suche/index.php?query=prof&topic_de[]=Personen&file_type_de[]=Webseite&faculty_de[]=Wirtschafts-+und+Rechtswissenschaften)

Daraufhin wurde der Crawler auf URLs beschränkt, die:

- Mit https://www.th-koeln.de anfangen, um Unterseiten wie bibl.th-koeln.de auszuschließen
- Nicht th-koeln.de/en/ enthalten, um englische Seiten auszuschließen
- Nicht /download enthalten, um einen Teil von Dateien auszuschließen
- Nicht mit einer Dateiendung, wie z.B. .ppt oder .mp4, enden
- Keine ? beinhalten, um die Suchschlitz Problematik zu umgehen

Die Datenqualität verbesserte sich, jedoch waren weiterhin leere und irrelevante Seiten der überwiegende Teil der Dokumente.

Performance – Die Performance dieses Ansatzes war auch nach mehreren Versuchen der Optimierung schlecht. Der Crawler/Scraper war in vielen Varianten selbst nach 24 Stunden nicht fertig – wobei dies zu einem Teil auch auf die Performance des ersten Name Matchings zurückzuführen ist, welches im Scraper eingebettet war.

Letztlich wurde der Scrapy Scraper aufgrund mangelhafter Performance und der Ansatz „einfach alles zu scrapen“ aufgrund nicht zufriedenstellender Dokument Qualität verworfen.

3.2.2. Ansatz TH-Suche

Um eine bessere Dokumentqualität zu erhalten, wird in diesem Ansatz auf die Suche der TH-Seite zurückgegriffen. Diese hat gleich mehrere nützliche Eigenschaften, die man für eine bessere Dokumentqualität verwenden kann.

1. Fast alle Ergebnisse haben Inhalt
2. Man kann auf Dokumenttypen einschränken, sodass man beispielsweise nur Webseiten, nur PDFs, nur Word Dateien, etc. angezeigt bekommt
3. Alle potenziell relevanten Dokumente stehen gesammelt nacheinander, sodass ein Crawler im eigentlichen Sinne nicht mehr gebraucht wird.

Die grundlegende Überlegung war simpel: Man sucht in der TH Suche und scrap alle aufgeführten Dokumente. Zu Beginn beschränkte man sich auf Webseiten, da diese den überwiegenden Teil der Dokumente ausmacht und diese am einfachsten auslesbar waren. Der erste Ansatz suchte nach jedem vollständigen Namen, der durch den BA Scraper aus dem Personenverzeichnis extrahiert wurde und scrapte alle aufgeführten Dokumente. Sollten Dokumente bei mehreren Suchen aufgeführt werden, wurde diese nur einfach gescrapt. Beim Betrachten der pro Suche neu hinzukommenden Dokumente fiel auf, dass nach ca. der 500-sten Suche kaum noch neue gefunden werden und die restlichen 2000 Suchen im Grunde kaum einen Mehrwert liefern.

Da das Scrapen der ca. 2500 Suchanfragen ca. 3 Stunden dauerte und mit jeder Suche immer weniger neue Dokumente gefunden wurden, wurde ein Trick angewandt, um fast alle Dokumente mit einer Suchanfrage zu erhalten. Anstatt nach jedem Namen einzeln zu suchen, wurde die query einfach leer gelassen. Mit dieser query wurden bis auf wenige Ausnahmen alle Dokumente gefunden, die vorher durch die Namenssuchen gefunden wurden *(th-koeln.de/suche/index.php?query=).* Die Anzahl der Suchen reduzierte sich von 2500 auf 1, sodass der gesamte Prozess wesentlich vereinfacht wurde.

Zudem wurde die Scrape Logik angepasst, um die Dokumente asynchron zu beziehen. Es wurden zuerst alle URLs des Ergebnisses gescrapt und danach die Dokumente hinter den URLs Real Python. (o. D.). Durch die Verwendung von asynchronen Requests dauerte der gesamte Prozess nun nur noch wenige Minuten, wobei die Tageszeit entschied, ob der Prozess in 40 Sekunden (Nacht) oder 5 Minuten (Mittag) durchlief. Nach einiger Zeit wurden auch Bildergalerien gescrapt, die reguläre Webseiten mit einigen Bildern sind.

Neben Webseiten wurden auch PDFs mit der gleichen Herangehensweise gescrapt und mittels des Python Moduls PyPDF2 der enthaltene Text extrahiert. Die Inhalte der PDFs reichen von Bauplänen über Flyer und Formularen bis zu fachlichen Artikeln und mussten daher so gefiltert werden, dass möglichst nur die Dokumente mit fachlichem Bezug überbleiben. So wurden beispielsweise Dokumente aussortiert, die Wörter wie Formular, Lageplan, Antrag, Fragebogen oder Terminübersicht in der URL hatten, was die Anzahl der überbleibenden PDFs von ursprünglich 5580 auf 2700 reduzierte. Zudem wurden einige PDFs teilweise oder vollständig falsch eingelesen, da diese beispielsweise gescannte Bilder oder Texte enthielten. Dies resultiere darin, dass diese Dokumente entweder vereinzelnd oder vollumfänglich aus Anreihungen von zufälligen Unicode Zeichen bestanden.

3.2.2. Projekte

Die auf der TH Seite aufgeführten Projekte wurden ebenfalls als eine potenziell wertvolle Quelle für relevante Dokumente angesehen, da in diesen speziellere fachliche Themen behandelt und zu Personen zugeordnet werden. Die Projekte wurden ähnlich wie die anderen Dokumente gescrapt, wobei lediglich die Basis URL eine andere ist und man keine Suche konstruieren musste *(www.th-koeln.de/forschung/aktuelle-projekte_2418.php)*.

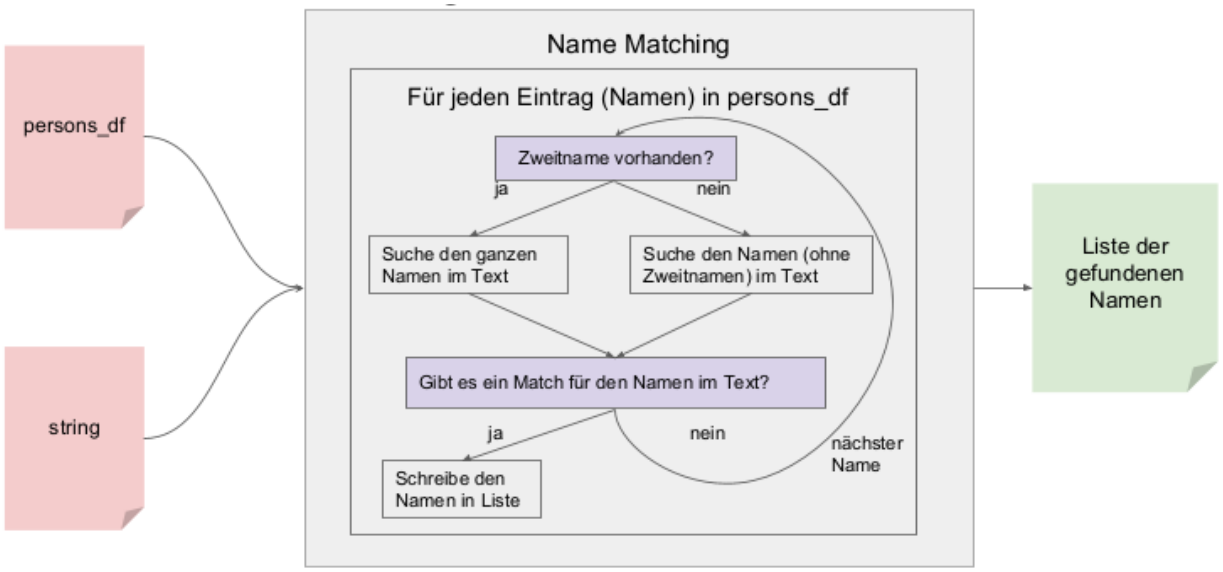
3.2.3. Name Matching

Im Rahmen der Datenbeschaffung wird ein Name-Matching durchgeführt, welches das Ziel hat, jedem Dokument die darin vorkommenden Personen zuzuordnen. Insbesondere sollen die folgenden zwei Aspekte erreicht werden:

Zum einen werden Dokumente aussortiert, die keiner Person zugewiesen werden können. Dadurch wird die zu speichernde Datenmenge vermindert.
Zum anderen stellt diese Vorgehensweise die Weichen für das Experten-Empfehlungssystem, da die Dokumente hier bereits mit entsprechenden Experten verknüpft werden.

Ursprünglich wurde ein Ansatz gewählt, der auf Lautschrift basierte. Hierzu wurden alle Namen und Texte mittels der Python Bibliothek epitran (Calzolari et al., 2018) in Lautschrift konvertiert, bevor das matching durchgeführt wurde. Dieser Ansatz bietet eine relativ zuverlässige Erkennung von Namen in Texten, da Rechtschreibfehler hier kein Problem darstellen. Dies ist besonders bei externen Quellen ein Vorteil. Jedoch wies die Konvertierung in Lautschrift relativ lange Laufzeiten auf.
Nach der Durchführung von Stichproben, ist davon auszugehen, dass keine bzw. kaum Rechtschreibfehler in den Namen von Personen mit Zugehörigkeit zur TH Köln innerhalb der internen Quellen auftreten. Darüber hinaus wurden im Rahmen unserer Arbeit schlussendlich keine externen Quellen hinzugezogen.
Aus den genannten Gründen haben wir uns dazu entschieden den Ansatz zu verwerfen.

Stattdessen nutzen wir einen Ansatz, der auf einer Regex basiert. Hierzu wird der zu prüfende Text als String und die persons.pq Datei verwendet, die innerhalb des „BA-Scrapers“ erzeugt nach Vorverarbeitung als DataFrame eingelesen wird. Die persons.pq enthält dabei ein DataFrame, welche die Namen aller Personen enthält, in Titel, Vorname, ggf. Zweitnamen und Nachnamen geteilt. Die Personenliste kann dabei entsprechend angepasst werden, um beispielsweise nur nach Personen mit dem Status Lehrbeauftragte/r zu suchen.



Mithilfe der entwickelten Regex-Muster können Namen mit oder ohne Titel gefunden werden. Dabei werden auch Namen erfasst, bei denen Vor- und Zweitnamen in Form von Initialen vorkommen. Die Zweitnamen sind dabei immer optional.
Als Ergebnis des Name-Matching werden die Personen IDs aller gefundenen Namen als Liste zurückgegeben. Aktuell findet das Name Matching Anwendung in der mapping.py, in der über das Name-Matching identifiziert wird, in welchem Teil des Dokuments der Name eines/r Hochschulangehörigen steht.

3.3 Externe Quellen

Um weiterführende Dokumente zu erhalten, wurde die Beschaffung externer Quellen angestrebt. Ausgangspunkt hierfür, waren die ausgehenden Links auf den Personenseiten der TH Köln.
Die hier gesammelten Quellen wurden anschließend analysiert und gruppiert, um einen Überblick über die Kategorien der Webseiten und die Verteilung dieser zu erlangen.
Darüber hinaus wurden rechtliche Aspekte in Bezug auf das Scrapen fremder Webseiten erörtert. Als Ergebnis zeigte sich, dass das Scrapen von Webseiten für wissenschaftliche Zwecke i.d.R. erlaubt ist. Hierbei sind jedoch bestimmte Auflagen zu beachten:

- Die robots.txt muss beachtet werden.
- Die Daten dürfen nur für ein konkretes Forschungsprojekt gescraped werden und nach Abschluss des Projektes gelöscht werden.
- Es dürfen nur Auszüge einer Webseite gescraped werden.

Bei der Entwicklung eines Scrapers für externe Quellen stellten sich die rechtlichen Vorschriften als wesentliches Problem dar, da diese nicht eindeutig sind.
Daher wurden zunächst innerhalb eines Vorverarbeitung Skripts (prepare_outlinks.py) alle Domains anhand einer Liste aussortiert, für die klar war, dass sie nicht gescraped werden dürfen. Allerdings war diese Liste nicht vollständig, sodass nach wie vor ein rechtliches Risiko bestand.
Des Weiteren wurden im Rahmen des Vorverarbeitung Skripts URLs aussortiert, die auf interne Webseiten verweisen oder die nicht erreichbar sind.
Um das rechtliche Problem zu lösen haben wir uns, statt der manuellen Filterung, entschieden das Scrapy Framework für die externen Quellen zu verwenden. Dieses lässt sich so konfigurieren, dass beim Scrapen automatisch die robots.txt der Webseite beachtet wird. Hierfür wird die vorverarbeitete Liste der Links zu externen Webseiten als Start-URLs eingelesen. Diese wird von Scrapy abgearbeitet und die entsprechenden Seiten mit einem Tiefenlimit von eins gescraped. Dabei sollen URLs mit Suchanfragen ausgeschlossen werden. Ungeachtet dessen traten weitere Probleme auf. Zum einen ist das HTML jeder Webseite anders aufgebaut, sodass es schwierig ist, festzulegen, welche Teile extrahiert werden sollen. Dies kann weitestgehend dadurch gelöst werden, dass das erste p-Tag innerhalb des HTML-Bodys ausgelesen wird.
Für uns war es nicht möglich die privaten Webseiten der Lehrenden von Webseiten wie LinkedIn o.a. zu unterscheiden. Dies gestaltet sich als wesentliches Problem, da Webseiten von Lehrenden sehr viele wichtige Informationen liefern können, sodass wir diese vollständig scrapen wollten. Wie bereits erwähnt ist dies für diverse Webseiten von Unternehmen untersagt. Als mögliche Lösung könnte hier vor der Speicherung eine Prüfung erfolgen, ob die Seite in der URL oder im Inhalt den Namen von mindestens einer der gelisteten Personen enthält. Die Implementation eines solchen Filters ist uns innerhalb von Scrapy jedoch während des Projektes nicht gelungen.
Aufgrund der beschriebenen Schwierigkeiten haben wir die externen Quellen nicht in unseren Prototypen einbezogen.

3.4 Der Datensatz

Der momentan vorliegende Datensatz von Dokumenten umfasst 3950 Webseiten, 1068 PDFs, 138 Bildergalerien und 407 Projekte, die über das Name Matching mindestens einer relevanten Person zugeordnet werden konnten, wobei Professoren, Lehrbeauftragte und Lehrkräfte mit besonderen Aufgaben als relevant eingestuft werden. Für Webseiten, Bildergalerien und Projekte liegen jeweils folgende Spalten vor:

- **document_id**: Eine eindeutige Dokument-ID
- **url**: Die URL
- **html**: Das gesamte HTML Script der Seite
- **title**: Den Titel der Seite, falls vorhanden
- **article**: Sämtlicher Text der Seite der zwischen <article> Tags steht
- **names_from_left_panel**: Die Namen die im linken Panel aufgeführt werden, falls vorhanden
- **type**: Der Dokumenttyp (website, gallery, project)

Für PDFs liegen folgende Spalten vor:

- **document_id**: Eine eindeutige Dokument-ID
- **url**: Die URL
- **article**: Der extrahierte Text
- **type**: Der Dokumenttyp (pdf)

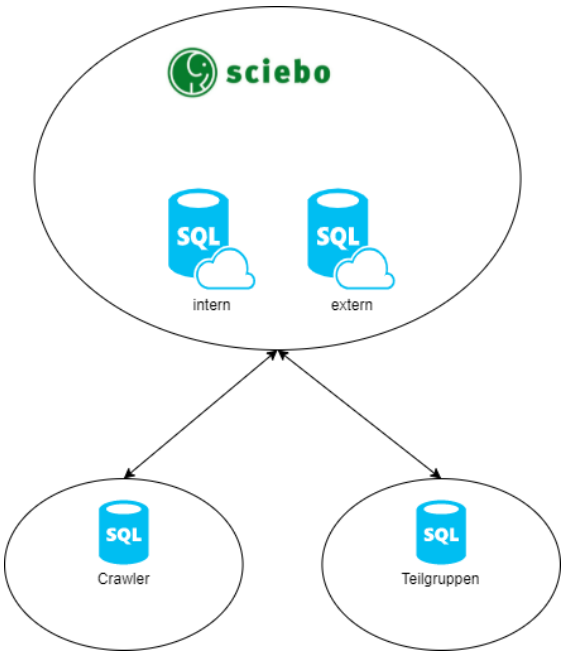
Die Dokumente werden über eine Mapping Tabelle zu den verschiedenen Personen gemappt. Diese Tabelle enthält folgende Spalten:

- **document_id**: Die Dokument-ID
- **person_id**: Die Personen ID
- **found_in_article**: True, wenn der Name im Text gefunden wurde
- **found_in_left_panel**: True, wenn der Name im linken Panel gefunden wurde

4. Vorverarbeitung des Datensatzes

In einer frühen Phase des Projekts sollten die gecrawlten Daten als SQL-Datenbankdateien auf einem sciebo Projektordner zentral zur Verfügung gestellt werden. Dazu wurde bei der Campus IT der TH Köln (*Campus IT*, o. D.) ein sciebo Projektordner mit einer Größe von 1 TB beantragt. Per pyocclient (*Owncloud - Pyocclient*, o. D.) konnten über Python Skripte Ordner und Dateien gelesen, sowie geschrieben werden. Mit dem SQLite Modul konnten im weiteren Verlauf die zuvor eingelesenen SQL-Dateien in einer Python Entwicklungsumgebung

aktualisiert werden. Der Crawler sollte Daten in SQL Files schreiben und diese auf Sciebo hochladen. Die anderen Teilgruppen konnten diese SQL Files zur weiteren Verarbeitung aus dem Sciebo Projektordner herunterladen, aktualisieren und anschließend wieder in den Sciebo Projektordner hochladen.



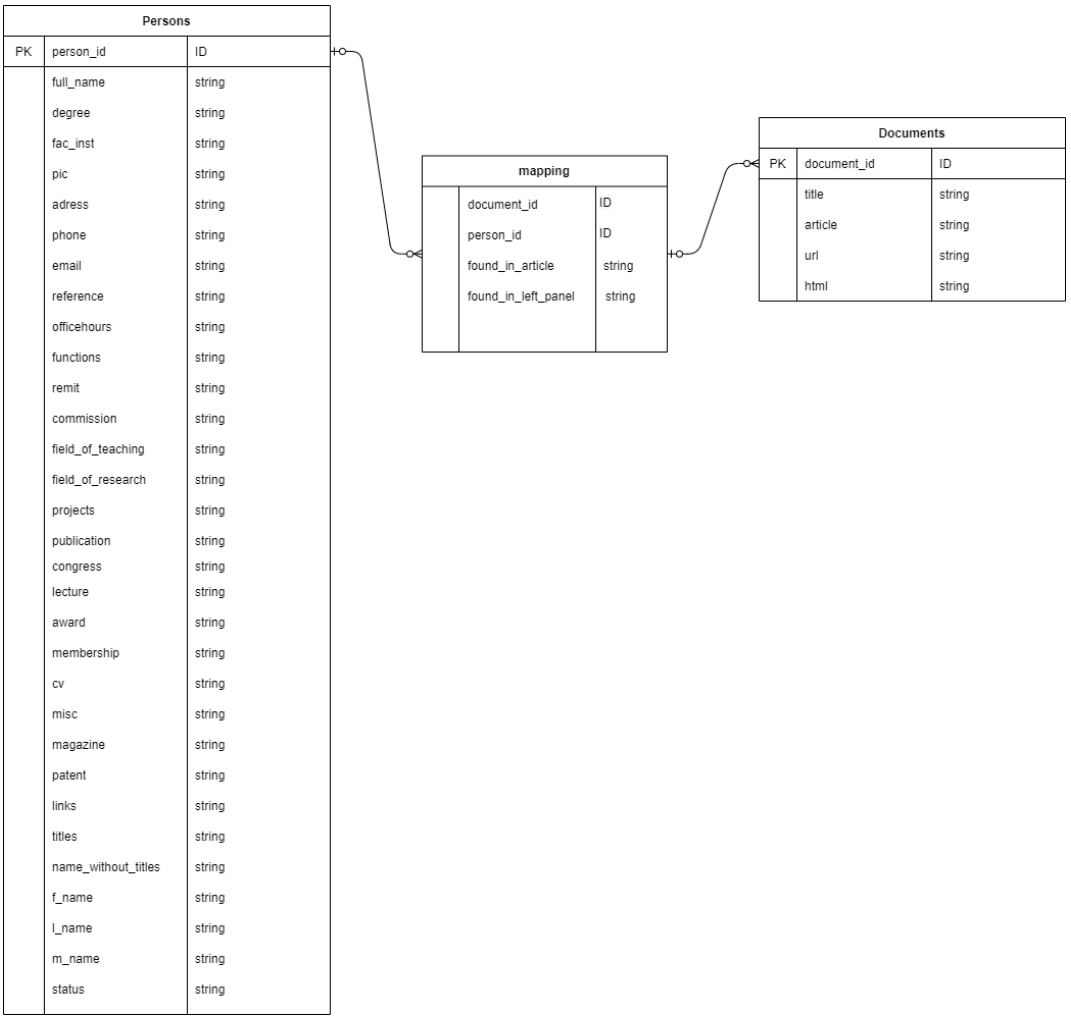
Dieser Prozess erwies sich als unpraktikabel und so wurde entschieden eine im Internet zugängliche MariaDB als zentralen Speicherort einzurichten. Da nicht abzusehen war, wann über die TH Köln eine entsprechende MariaDB zur Verfügung gestellt werden konnte, wurde zunächst eine private MariaDB verwendet. Später wurde die Datenbank auf eine von der TH Köln gehostete MariaDB migriert.

Über die Python Module mariadb (Hedgpeth, 2020) und sqlalchemy (*SQLAlchemy - The Database Toolkit for Python*, o. D.) wird direkt eine Verbindung zu der gewünschten Tabelle auf der Datenbank aufgebaut und es können Daten gelesen, gelöscht oder aktualisiert werden.

Das Datenbankschema basiert auf drei Tabellen:

- Persons
    - Die Persons Tabelle enthält die resultierenden Daten aus dem BA Crawler und umfasst die Informationen der Personenseiten der TH Köln.
- Documents
    - Die Documents Tabelle speichert die Informationen der internen Quellen, welche zuvor gecrawlt wurden.

- mapping
    - Über die Mappingtabelle werden die beiden Tabellen miteinander verknüpft.



Es wurden mehrere Beispielfunktionen zur Implementierung der Datenbank in die Skripte der Teilgruppen zur Verfügung gestellt. Die Zugangsdaten und Serveradresse werden über Systemvariablen in die Funktionen eingebracht. In bestimmten Kombinationen von gecrawlten Daten und der Art und Weise wie die Verbindung zur Datenbank aufgebaut worden ist, keine Daten in die Tabelle geschrieben werden konnten. Bei manchen Daten mussten diese z.B. erst alle Daten in Strings konvertiert werden und bei anderen Daten wurden bestimmte Sonderzeichen per RegEx entfernt. Deswegen werden unterschiedliche Funktionen für den Verbindungsaufbau zur und das Schreiben in die Datenbank zur Verfügung gestellt. Das Abrufen der Daten funktionierte problemlos.

4.2. Pre Processing Pipeline

Um mittels *Latent Dirichlet Allocation* Themen für die einzelnen Dokumente bestimmen zu können, sollten die generierten Daten bereinigt, tokenisiert und mittels Gewichtung gefiltert werden.

Um dies zu bewerkstelligen wurde eine Klasse bestehend aus 18 Funktionen erstellt, die sich grundlegend in 4 Schritte aufteilen lässt: Download der Dokumente, Säuberung und Tokenisierung der Dokumente, berechnen der TF-IDF Gewichtung und IDF Gewichtungen für einzelne Tokens und die Speicherung der bereinigten Dokumente in einem geeigneten Format. Im Folgenden wird der Ablauf der preprocessing Klasse *Pipeline* beschrieben.

Im ersten Schritt werden die Daten durch die Funktion *read_sql_to_df* mittels *sqlalchemy* (*SQLAlchemy - The Database Toolkit for Python*, o. D.) und der Pandas Funktion read_sql_table aus der SQL-Datenbank heruntergeladen und via *Pandas* (*pandas - Python Data Analysis Library*, o. D) in einem Dataframe gespeichert. Dieses Dataframe dient als strukturelle Grundlage der weiteren Verarbeitung. Da im vorhergehenden Schritt des scrapens, die Inhalte aus dem Titel-Feld im Artikel-Feld zusammengefasst wurden, bezieht sich die Verarbeitung der Dokumente in den folgenden Schritten nur auf das „article“-Feld und auf die daraus neu entstehenden Spalten des Dataframes.

Im nächsten Schritt werden mittels der Funktion *regex_pre_cleaning* verschiedene Zeichen, die nicht im lateinischen Alphabet enthalten sind mithilfe der Unicode Struktur herausgefiltert. Ebenso werden Sonderzeichen und spezielle Zeichenketten herausgefiltert um im Folgenden, mit der Funktion *tokenization,* Tokens aus den einzelnen Worten zu erzeugen. Bei diesem Schritt wird eine gesonderte Liste mit Wortkompositionen die eine Leerstelle oder Bindestriche enthalten abgefragt, um diese Wortkompositionen als einzelne Tokens zu erhalten. Diese Liste wurde intellektuell erschlossen.

Aus den generierten Tokens werden alle Vor-, Mittel- und Nachnamen sowie die akademischen Grade die in der Datenbank *persons* gespeichert sind, entfernt. Hierzu kommen die Funktionen *generate_name_list* und *remove_names* zum Einsatz.

Um die Sprache der einzelnen Dokumente bestimmen zu können wurde ein hybrider Ansatz gewählt. In der Funktion *lang_by_stop* wird einmal, mittels der Häufigkeit der vorkommenden Stoppwörter der jeweiligen Sprache die Sprache der Spalte bestimmt, als auch durch die Python Bibliothek *langdetect* (*Langdetect*, o. D.). Das Ergebnis beider Methoden wird abgeglichen. Wenn der Abgleich übereinstimmt, wird die Sprache als deutsch oder englisch vermerkt.

Durch die Funktion *remove_stopwords_from_list* werden alle vorkommenden Stoppworte mittels der Python Bibliothek *nltk* (Natural Language Toolkit — NLTK 3.6.2 documentation, o. D.) entfernt.

Im letzten Schritt der Datenbereinigung werden durch die Funktion *filter_locations_from_tokens* Länder-, Städte und Ortsnamen in deutscher und englischer Sprache herausgefiltert. Hierzu wird anhand von drei Listen ein Matching auf die generierten Tokens durchgeführt und diese entfernt. Die Listen unterteilen sich in eine weltweite Städteliste mit 28050 Einträgen sowie eine Liste mit 249 englischsprachigen Länderbezeichnungen und eine Liste mit 277 deutschsprachigen Länderbezeichnungen (*World Cities Database | Simplemaps.Com*, o. D.). Zusätzlich wurden die Einträge sowohl in Unicode als auch in ASCII-Codierung codiert und mit den Tokens abgeglichen.

Die bereinigten Daten werden im Anschluss durch die Funktion *lemmatization* mittels der Python Bibliotheken *HanTa (HanTa, o. D.)* für deutsche Tokens und *nltk für englischsprachige Tokens* lemmatisiert. Die Wortkompositionen wurden dabei vom Lemmatisierungsprozess ausgeschlossen.

Mit der Funktion *calculate_tf_idf* wird dann die Gewichtung der einzelnen Tokens auf den gesamten Korpus berechnet und das Token mit entsprechendem Gewicht, in einer separierten Spalte des Dataframes abgelegt. Hierbei wird für jedes Wort eines Dokuments sowohl der tf_idf Wert und das zugehörige Wort in einer nested List in einer Spalte hinterlegt.

Mittels dieser Gewichtung ist es nun möglich ein Threshold für eine Gewichtung anzugeben und die Auswahl aus dem Data Frame zu entfernen. Es stellte sich bei der Anwendung heraus, dass sich der IDF-Wert als Gewichtung zum filtern von Token besser eignete als der tf_idf, da einige irrelevante Terme durch eine hohe Termfrequenz in einigen Dokumenten nicht herausfielen.

Der gesamte Korpus eignet sich nicht gut für die Erstellung von Topics im LDA Topic Modelling, da ein Großteil der Dokumente kaum fachspezifischen Bezug beinhaltet. Für die Erstellung eines fachspezifischen Korpus wurde die Funktion *create_full_corpus_df_and_selected_corpus_df* implementiert. Diese erstellt sowohl ein Dataframe für den kompletten Korpus als auch einen Korpus der nur die gescrapten PDF Dokumente, die Projektseiten und die Forschungsseiten der TH Seite beinhalten. Die PDF-Dokumente werden dabei zusätzlich mit den zugeordneten Autoren gemappt und gefiltert, wenn einem Dokument mehr als 10 Autoren zugeordnet waren. Ziel war es nur die PDFs zu nutzen, die spezifisch einem oder mehreren Autoren zuordnen zu können. Auf den

reduzierte Datensatz wurde die komplette Pipeline inklusive TF-IDF Berechnung angewendet und anschließend für die Topic Generierung genutzt, während auf den kompletten Korpus nur die Schritte bis einschließlich Lemmatisierung angewandt wurden. Die auf Basis des reduzierten fachspezifischen Korpus generierten Topics werden anschließend wieder dem kompletten Korpus zugeordnet.

Im letzten Schritt des pre processings können die Daten durch die Funktion *create_parquet_file* in einem Parquet-File gespeichert und weiterverarbeitet werden.

4.2.1 Designentscheidungen

Zur besseren Implementierung in eine gesamt übergreifende Struktur wurde die preprocessing *Pipeline* als Klasse implementiert. Weiter wird dadurch die Integration einzelner Skripte vereinfacht und die Übersichtlichkeit gesteigert sowie eine Vererbung der Inputparameter ermöglicht.

4.2.2 Problematik Implementierung

Die Arbeit an der Pipeline konnte nur auf die zur Verfügung stehenden Dokumente optimiert werden. Das führte dazu, dass durch neue Dokumente und damit einhergehenden Sonderzeichen die Pipeline teilweise an Funktionalität einbüßen musste. Besonders stark konnte dies bei der späteren Ergänzung des Korpus durch die PDFs beobachtet werden. Hier fanden sich Bindestriche unabhängig von Silben und andere Artefakte, die in der kürze der Zeit nicht mehr herausgefiltert beziehungsweise angepasst werden konnten. Weiter wurde durch das Hinzufügen der PDF-Dokumente die Laufzeit der Berechnung des TF-IDF-Werts deutlich erhöht und stellt nun ein Problem dar. Dies könnte durch die Implementierung von Pandarallel optimiert werden.

Weiter wäre es von Vorteil gewesen, die Zeichen die nicht dem lateinischen Alphabet angehören mittels eines Whitelist Ansatzes zu Filtern, anstelle des von uns gewählten Blacklist Ansatzes.

Auch das Filtern von irrelevanten Zeichenketten stellte sich als Herausforderung dar. Beispielsweise Links zu Social Media Diensten, diese sind meist irrelevant für die weitere Verarbeitung, jedoch bei Experten für Social-Media wiederum relevant. Viele Irrelevante Zeichenketten mussten manuell in den Dokumenten gesucht und in eine jeweils spezielle Filterfunktion implementiert werden. Auch das Filtern von Vor-, Mittel- und Nachnamen, die nicht in der *persons* Datenbank hinterlegt wurden, stellt eine Herausforderung dar. Diese Probleme konnten nicht generisch gelöst werden.

5. Extraktion der Topics aus den Dokumenten

Nachdem die Dokumente preprocessed wurden, bestand der nächste Schritt in der Extraktion der Topics aus den Dokumenten. Hierfür wurde der LDA-Ansatz gewählt, welcher im Abschnitt 2 Methoden/Projektpipeline vorgestellt wurde.
Zur Findung geeigneter Themenfelder wurde der gesamte verfügbare Korpus auf Projektseiten, Forschungsseiten und PDFs der TH Köln beschränkt, deren Inhalte am ehesten wissenschaftliche Themenbereiche der TH Köln abbilden. Im Folgenden wird dieser “Selected Korpus” genannt. Die verwendeten Funktionen sind im Skript lda_functions.py (DIS-KD-Project, 2021) enthalten, welches alle wichtigen Funktionen für die Erstellung und Evaluation der Topics (welche nachfolgend erläutert ist) notwendig sind.

5.1. Ermittlung der optimalen Anzahl der Topics

Für die Erstellung von fachbezogenen Topics ist es zunächst relevant, die optimale Anzahl der Topics für den gegebenen Korpus zu bestimmen, was im Skript optimal_num_topics.py (DIS-KD-Project, 2021) definiert wird. Hierfür wurde zunächst ein pickle eingelesen, welche die vorverarbeiteten Tokens des Selected Korpus enthält. Im nächsten Schritt wurde der Selected Korpus anhand der IDF-Spalte und der Anzahl von Tokens gefiltert. Alle Tokens, dessen IDF-Score unter dem Mindestwert von 3,5 liegen, wurden aussortiert. Dieser Wert erwies sich als am besten geeignet, nachdem verschiedene Wertebereiche ausprobiert und stichprobenartig geprüft wurde, welchen Score die jeweiligen Tokens haben und welche bei einem gewissen Score herausgefiltert würden. Dabei war wichtig, dass möglichst keine fachspezifischen Terme wie „Erneuerbare Energien“ und möglichst viele sehr allgemeine Begriffe aus dem Korpus entfernt werden. Außerdem wurden nur Dokumente mit mindestens 10 Tokens berücksichtigt.
Nach der Vorselektion wurde mithilfe der Funktion create_corpus_and_textdata eine Liste mit dem reinen textlichen Inhalt, ein gensim Dictionary, sowie der Korpus gebaut. Diese werden benötigt, um nachfolgend mithilfe der Funktion compute_coherence_values verschiedene Modelle mit unterschiedlichen Anzahlen an Topics zu bauen, welche anschließend anhand der Kohärenz miteinander verglichen werden können:
Es wird zunächst festgelegt, was die minimale und maximale Anzahl an Topics sein soll und in welchen Abständen neue Modelle gebaut werden sollen. Anschließend werden die unterschiedlichen Kohärenten ausgegeben. Zur Festlegung der Anzahl wird empfohlen die Topicanzahl mit der höchsten Kohärenz zu wählen (Röder et al., 2015). Für den Selected

Korpus wurden folgende Settings verwendet: start=25, limit=501, step=25. Damit wurde der sehr umfangreiche Topicanzahlbereich von 25 bis 500 Topics in 25er-Schritten abgedeckt und dessen Kohärenz ermittelt. Das Modell mit 500 hatte in diesem Fall die höchste Kohärenz, weshalb sich auf diese Anzahl an Topics geeinigt wurde.

5.2. Trainieren des Modells

Nachdem mithilfe des vorherigen Skripts die optimale Anzahl an Topics bestimmt wurde, wird anschließend mithilfe des Skripts build_model.py (DIS-KD-Project, 2021) das Model mit 500 zu erstellenden Topics gebaut. Vor der Modellerstellung wurde der Korpus zunächst wieder auf die zuvor genannten Settings (IDF, Anzahl Tokens) gefiltert. Anschließend wurden erneut die Liste mit dem reinen textlichen Inhalt, das gensim Dictionary sowie der Korpus erstellt. Unter Verwendung der Funktion lda_on_corpus wurde anschließend das Modell erstellt und gespeichert. Als Modell wurde das MultiCore Modell von Gensim verwendet (Řehůřek, 2021). Hierbei haben sich 15 passes als hilfreich und ausreichend erwiesen und wurden daher standardmäßig eingestellt. Mithilfe der Funktion get_topics können nachfolgend die einzelnen Topics in Form einer CSV-Datei ausgegeben werden.

Einige aus dem LDA-Modell resultierenden Topics erweisen sich als durchaus sinnvoll und decken wichtige Themenbereiche der TH Köln ab. Im Großen und Ganzen sind die Topics jedoch nicht so aussagekräftig wie erwartet. Da die Topics nicht direkt beeinflussbar sind, sind beispielsweise zwei oder mehrere verschiedene Themenbereiche in einem Topic vereint. Manchmal kommt es aber auch dazu, dass Terme in einem Topic enthalten sind, welche überhaupt nicht miteinander zu tun haben oder keinem Fachgebiet zugeordnet werden können.

Zwei aussagekräftige Topics sehen wie folgt aus:

Topic ID	Terme
322	mediumen, weißabgleich, lichtfarbe, photokina, mischlicht, bildaufnahme, ungenauen, camera, kamera, lter
326	cloud, schema, sensorcloud, db, fdbs, dbms, datenbanksysteme, Ã, förderiert, hersteller

Tabelle X: Ausgewählte gute Topics

Das Topic mit der ID 322 beinhaltet überwiegend Terme aus dem Bereich Medien und gehört zu den sehr aussagekräftigen Topics. Lediglich die Terme “ungenauen” und “lter” fallen aus dem Raster und sollten grundsätzlich in keinem Topic enthalten sein, da diese keinem

Fachbereich zugeordnet werden können. Es ist auch nicht ganz klar, woher der Term “Iter” überhaupt stammt.
Die Terme aus dem Topic mit der ID 326 lassen sich zum Großteil dem Bereich Datenbanken zuordnen. Die Terme “Ã”, “förderiert”, “hersteller” passen allerdings nicht zu diesem Bereich, wobei der Term “Ã” wahrscheinlich durch die Hinzunahme der PDF-Dokumente bei der Modellerstellung entstanden ist. Aufgrund von Zeitproblemen konnte dies jedoch nicht vom Preprocessing-Team berücksichtigt werden.

Im Gegensatz dazu sehen zwei schlechte Topics wie folgt aus:

Topic ID	Terme
94	mediennutzung, lv, #, israelisch, societies, rückversicherung, turnus, hochschuldidaktik, whatsapp, nadev
428	forschungsstrategie, stärkung, forum, vision, forschungsprofil, forschungsförderung, funk, weiter-, technologietransfer, organisationsentwicklung

Tabelle X: Ausgewählte schlechte Topics

Das Topic mit der ID 94 enthält ein paar wenige fachspezifische Terme, nämlich “mediennutzung” aus dem Bereich Medien und “rückversicherung” aus dem Fachgebiet Versicherungswesen. Die restlichen Begriffe sind jedoch unspezifisch oder ergeben grundsätzlich keinen Sinn, wie z.B. “#”. Insgesamt passen die Terme überhaupt nicht zusammen, weshalb das Topic unbrauchbar ist.

Das Topic mit der ID 428 enthält zwar zusammenpassende Terme, jedoch sind diese nicht fachspezifisch, weshalb auch Begriffe wie “forschungsstrategie” oder “forschungsprofil” während des Preprocessings herausgefiltert werden sollten.

5.3. Erstellen der Topic-Dokument-Relation des gesamten Korpus

Die Ermittlung der Topic-Dokument-Relation bzw. die Zuordnung der Topics zu den Dokumenten für den gesamten Korpus wurde im Skript unseen_docs.py realisiert. Zunächst wurde das resultierende LDA-Modell aus dem Skript build_model.py (DIS-KD-Project, 2021) eingelesen. Anschließend wird, ähnlich wie in den Skripten zuvor, auch hier wieder eine Liste mit dem reinen textlichen Inhalt, ein gensim Dictionary, sowie der Korpus gebaut. Allerdings erfolgt dies in diesem Fall zweimal, wobei zwischen Selected Korpus und dem gesamten Korpus unterschieden wird. Der Selected Korpus wird wie zuvor mit der Funktion create_corpus_and_textdata verarbeitet. Das resultierende gensim Dictionary, welches auch

für die Erstellung des LDA-Modells verwendet wurde, wird anschließend als Basis verwendet, um den gensim Korpus für den gesamten Dokumentenkorpus zu erstellen. Anschließend wird mit der Funktion topic_document_relation die Topic-Dokument-Relation realisiert. Das Ergebnis ist ein DataFrame, welches jeweils maximal die Top 3 Topics inklusiver Perc_Contribution der einzelnen Dokumente umfasst. Hierbei ist zu erwähnen, dass nicht zwangsweise zu jedem Dokument 3 Topics zugewiesen wurden, aber maximal 3. Die Perc_Contribution sagt dabei aus, wie sehr das Topic zu dem entsprechenden Dokument passt (Prabhakaran, 2020).

Die stichprobenartige Analyse der Topic-Dokument-Relation zeigte, dass die zugeordneten Topics vieler Dokumente nicht die fachbezogenen Dokumenteninhalte wiedergeben. Das liegt vor allem damit zu begründen, dass nur wenige der resultierenden Topics, wie zuvor erwähnt, einen Fachbereich abdecken und dadurch überhaupt aussagekräftig sind. Für ausgewählte Dokumente konnte jedoch durchaus sinnvolle Topics gefunden werden, welche im Folgenden beispielhaft beschrieben werden:

Dokument Nr. 40:
https://www.th-koeln.de/angewandte-sozialwissenschaften/oeffnung-des-wohnquartiers-fuer-das-alter_17816.php

Topic 1	**Topic 2**	**Topic 3**
Topic ID 301: vermittler, öffna, sozialraum, seniorenberatung, wohnquartier, öffnung, beratungsstelle, beratungsangebot, silqua, kontaktaufnahme	Topic ID 66: beratungsbedarf, unerkannt, informationsangebot, überreichen, überbrückung, wohnquartier, notsituation, projektlaufzeit, friseure, ehrenfeld	Topic ID 331: Mehrfach, sterberaten, unsicherheit, mirko, letztlich, stromnetz, wei-, nahe, koordinieren, ordnen

Tabelle X: Topic-Dokument-Relation für Dokument Nr. 40

Im Dokument Nr. 40 geht es um ein Buch über die Öffnung des Wohnquartiers für das Alter und die Entwicklung einer kommunikativen Informationsinfrastruktur zur Überbrückung struktureller Löcher im Sozialraum. Das beste Topic mit der ID 301 enthält wichtige Terme wie “sozialraum”, “seniorenberatung” und “wohnquartier”, die den Inhalt des Dokuments wesentlich wiedergeben. Allerdings kann aus diesem Topic kein Expertenwissen generiert werden, da die Terme unpassend für eine solche Zuordnung sind. Das Topic mit der ID 66 enthält ebenfalls Terme, die den Inhalt des Dokuments ausreichend wiedergeben, jedoch ist

auch dieses für die Zuordnung von Experten nicht geeignet. Das Topic mit der ID 331 ist nicht aussagekräftig und gibt nicht den Inhalt des Dokumentes wieder. Es ist nicht erklärbar, warum dieses Topic überhaupt dem Dokument zugeordnet wurde, da keiner der Terme im Dokument vorkommt.

Dokument Nr: 15074
https://www.th-koeln.de/anlagen-energie-und-maschinensysteme/ki-mobil_74226.php

Topic 1	**Topic 2**	**Topic 3**
Topic ID 81: künstliche intelligenz, intelligenz, künstlich, big, algorithmen, maschinell, handschrift, fahre, telematik, rezept	Topic ID 181: duroplasten, prozessparameter, projektbeschreibung, präsenzveranstaltung, studierendenzentriert, ipk, lfk, kurzfassung, materialeigenschaft, einsparung	Topic ID 428: forschungsstrategie, stärkung, forum-, vision, forschungsprofil, forschungsförderung, funk, weiter-, technologietransfer, organisationsentwicklung

Tabelle X: Topic-Dokument-Relation für Dokument Nr. 15074

Das Dokument 15074 thematisiert das Forschungsprojekt “KI: Mobil” und ist übergeordnet dem Bereich Künstliche Intelligenz (KI) zuzuordnen. Das LDA-Modell detektiere hierfür als das beste Topic das Topic mit der ID 81. Dieses beinhaltet Terme rund um das Gebiet Künstliche Intelligenz, wodurch der Dokumenteninhalt relativ gut erfasst wird. Die vier letzten Terme passen jedoch thematisch nicht zu den anderen Termen. Die Topics 2 und 3 mit der ID 181 und 428 passen thematisch nicht zum Dokumenteninhalt. Diese wurden zugeordnet, weil sowohl die Terme “projektbeschreibung”, “ipk” und “lfk” aus dem Topic 181 als auch die Teilterme “forschung” und “technologie” aus dem Topic 428 im Dokument enthalten sind. Diese sind jedoch in keiner Weise aussagekräftig, sowohl als Topic an sich als auch als Repräsentation das Dokument 15074.

5.4. Herausforderungen, Probleme und Zwischenergebnisse

Bereits nach dem ersten Durchlauf wurde schnell klar, dass an den Topics gearbeitet werden muss. Die erste Herausforderung war, die optimale Anzahl an Topics zu bestimmen, da diese sonst zu Allgemein oder zu durcheinander wären. Dieses Problem konnte gelöst werden, indem die Topic Coherence berechnet wurde und entsprechend die Anzahl an Topics mit dem höchsten Wert genommen wurde.

Hierbei entstand ein weiteres Problem: Die Laufzeit der Modellierung der einzelnen LDA-Models dauerte zu lang, um den Prozess auf einen privaten Rechner zu realisieren. Jedoch bestand die Möglichkeit den Prozess auf einer Virtuellen Maschine (VM) seitens der Hochschule laufen zu lassen, was zu einer erheblichen Verbesserung der Performance führte. Eine weitere Verbesserung der Performance konnte durch das Verwenden eines Multicoreansatzes (Řehůřek, 2021) erzielt werden. Dennoch waren die Ergebnisse, auch nach der Bestimmung der optimalen Anzahl an Topics, nicht zufriedenstellend, was daraus resultierte, dass viele Orte, aber auch Namen, sowie typische Hochschulbegriffe oft in den Dokumenten vorgekommen sind und die Topics entsprechend beeinflussten. Mithilfe des Preprocessing-Teams wurden anschließend Filterlisten erstellt, welche beispielsweise den Großteil aller Städtenamen herausgefiltert haben. Außerdem wurden tf-idf und idf-Werte für die einzelnen Terme berechnet. Diese sollten dabei helfen, die Terme weiter zu filtern. Allerdings erwies sich diese Filterung als schwierig, da die Bestimmung des Scores händisch erfolgen musste.
Aber auch nach der erfolgten Filterung waren die Topics noch nicht fachspezifisch. Aus diesem Grund wurde sich darauf geeinigt, dass fortan mit zwei verschiedenen Korpussen gearbeitet werden sollte. Der Korpus für die Erstellung der LDA-Topics sollte fachspezifischer sein und nur noch passende Dokumente enthalten. Demnach wurde der Korpus so angepasst, dass dieser nur Projekte, Forschungsseiten und PDF-Dateien berücksichtigt. Die Hinzunahme der PDF-Dateien führte allerdings zu neuen Schwierigkeiten, da diese Sonderzeichen beinhalten, welche zuvor nicht beachtet wurden. Das beeinflusste einige Topics negativ.
Die hieraus entstandenen Topics wurden jedoch nicht wie zuvor auf denselben Korpus angewandt, sondern auf den Gesamtkorpus.
Abschließend sollte festgehalten werden, dass die erstellten Topics immer noch nicht ideal sind. Da die Topics nicht beeinflussbar sind kommt es vermehrt vor, dass zwei Fachgebiete innerhalb eines Topics vorkommen. Dies müsste beim Weiterführen des Projektes entsprechend berücksichtigt werden und die Dokumente müssen im Nachhinein händisch gefiltert werden.

6. Erschließung der Themengebiete

6.1. Automatisch

Mit Blick auf den Browsing Ansatz wurde versucht, die Domänen inhaltlich einzuordnen und zu hierarchisieren. Dazu wurde für eine bessere “Aktualisierbarkeit” ein algorithmischer Ansatz verfolgt. Hierzu sollten Fachgebiete die der TH Köln internen Struktur zugeordnet wurden. Das heißt es wurden Lehrgebiete den Fakultäten und Instituten zugeordnet. Das Browsing

Interface besteht auf der ersten Ebene aus einer Darstellung der Fakultäten. Diesen sind Institute zugeordnet und als Letztes findet sich eine Auswahl an Lehrgebieten. Diese Lehrgebiete sind aussagekräftige Begriffe über die Fachbereiche der Experten und stammen aus den Personenseiten der TH Köln. Bei diesem Ansatz ist es möglich, auch interdisziplinäre Lehrgebiete zu ermitteln. Die Terme aus den Lehrgebieten können über ein Matching der Experten um die erstellten (LDA) Topics erweitert werden. So können die Fachbereiche an den Fakultäten und Instituten genauer dargestellt werden. Außerdem wird die Auswahlmöglichkeit in der Web-User Interface erweitert und füllt eventuelle Lücken bei den Fachgebieten der Personen.

Als Datenbasis dient die CSV-Datei „personen_th.csv“. Sie umfasst den gesamten Inhalt aller Personenseiten der TH-Köln Website. Über die Git-Shell wurden dann mittels ‚cut‘ Befehl nur gewisse Spalten extrahiert und in eine neue ‚gefilterte‘ CSV-Datei geschrieben (persons_new.csv). Zum Darstellen der Fachbereiche wurden pro Person die Kategorien: „Aufgabe“, „Lehrgebiet“, „field of research“, „Projekte“, „Veröffentlichungen“, „Vorlesungen“ und „Mitgliedschaft“ extrahiert. Weitere Möglichkeiten sind „Patente“, „Auszeichnungen“ und „Kommissionen“. Für das Ergebnis wurde nur der Inhalt der Spalte “Lehrgebiet” ausgewählt, da die anderen Spalten keine oder sehr wenige aussagekräftige Terme beinhalten. Zur besseren Modellierung wurde ein Skript (“categorize_name_faculty.py”) erstellt, um die gefilterte ‚persons_new.csv‘ in ein JSON-Format zu konvertieren (Person_fot.json). Da in den Rohdaten die Informationen zu Fakultät und Institut in derselben Spalte stehen, mussten diese getrennt werden. Dazu wurde im Skript händisch eine Liste mit allen 12 Fakultäten der TH-Köln angelegt. Die 53 Institute wurden dann mittels Regular Expressions erschlossen und automatisch in eine weitere Liste eingetragen. Das Ergebnis ist eine JSON Datei, die alle Personen der TH-Köln einer Fakultät, einem Institut und einem Lehrgebiet zuordnet, wenn entsprechende Inhalte vorhanden sind. Daraufhin hat eine Modellierung der Datei stattgefunden. Es wurde ein Dictionary erzeugt, das einer Fakultät ein Institut zuweist, wenn eine Person an diesem tätig ist. Das erzeugte Dictionary hat folgende Struktur:

{

Fakultät: {

Institut: [Lehrgebiete d. Kandidaten]}

}

Über diesen Ansatz ist es möglich, die Lehrgebiete den Fakultäten und Instituten zuzuordnen. Die JSON Struktur wurde gewählt, weil sie sich gut für die Darstellung des Browsing Ansatzes in der UI eignet. Diese Methode wurde allerdings verworfen, da die Domänen an der TH-Köln

ausschließlich über das LDA-Modelling aus den Dokumenten ermittelt werden und die LDA-Topics intellektuell erschlossen werden.

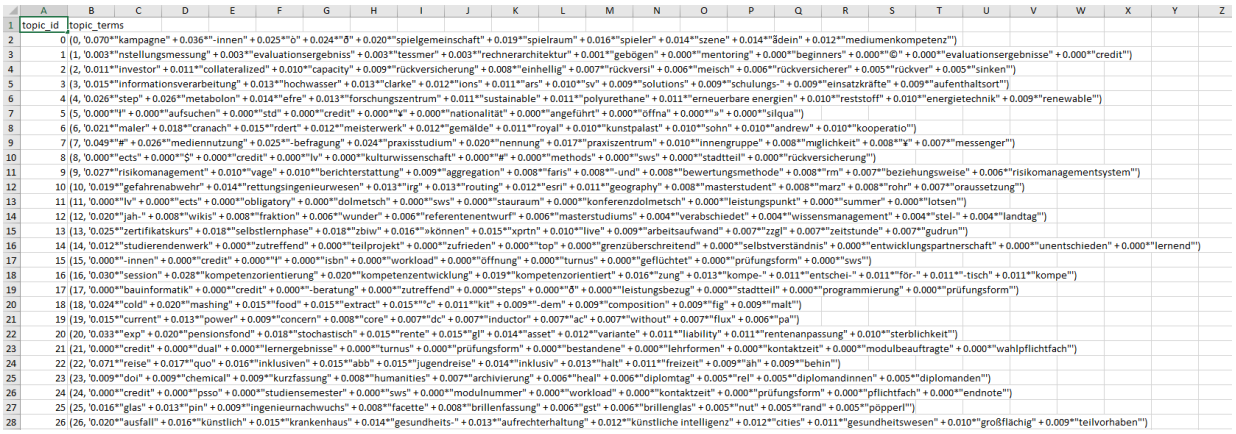
Alternativ kann das Browsing Interface auf der Website durch die Dewey-Decimal-Classificaton (DDC) dargestellt werden. Die DDC ist ein Klassifikationsmodell zur Sacherschließung von Bibliotheksbeständen. Der Aufbau eignet sich gut zum Einordnen Fachspezifischer Terme in eine vorgegebene Hierarchie die Fachbereiche inhaltlich erschließt. Die Lehrgebiete können dann in diese Struktur eingebunden werden.

6.2. Intellektuell

Wie im letzten Abschnitt schon beschrieben, ist eine Hierarchisierung der Lehrgebiete unvorteilhaft, da man nicht immer den Experten einem bestimmten Lehrgebiet zuordnen kann und das Empfehlungssystem folglich ungenau werden kann. Darüber hinaus verliert man die Möglichkeit, Topics miteinzubeziehen, die nichts mit einem bestimmten Lehrgebiet zu tun haben. Ein solches Topic ist beispielsweise „Mentoring“: Dieses Topic kommt im Datensatz sehr häufig vor, kann aber keinem bestimmten Lehrgebiet an der TH zugeordnet werden. Dadurch würde dieses Topic bei einer Erschließung mit dem ersten Ansatz nicht berücksichtigt werden. Außerdem wäre eine solche Erschließung nicht zielführend, da man sich auf der TH-Seite im Prinzip auch schon über die Lehrgebiete zu den Professoren durchklicken kann und unser Recommender-System dadurch redundant wäre.

Aufgrund dieser Tatsache musste ein neues Erschließungskonzept entwickelt werden. Wir haben uns dazu entschieden, die aus dem LDA-Modell extrahierten Topics inhaltlich zu erschließen.

Das Vorgehen sieht folgendermaßen aus: Der Output des LDA-Modells (in der oberen Abbildung zu sehen), in dem jede Zeile für ein Topic steht, wird genau intellektuell analysiert. Wenn es sich für ein Topic anbietet, wird ein „Oberbegriff“ für das Topic gewählt, was den Inhalt dieses Topics genau beschreiben kann (basierend auf den Begriffen des Topics).



topic_id	topic_terms
0	(0, '0.070*"kampagne" + 0.036*"-innen" + 0.025*"ö" + 0.024*"ð" + 0.020*"spielgemeinschaft" + 0.019*"spielraum" + 0.016*"spieler" + 0.014*"szene" + 0.014*"ädein" + 0.012*"mediumenkompetenz"')
1	(1, '0.003*"nstellungsmessung" + 0.003*"evaluationsergebniss" + 0.003*"tessmer" + 0.003*"rechnerarchitektur" + 0.001*"gebögen" + 0.000*"mentoring" + 0.000*"beginners" + 0.000*"©" + 0.000*"evaluationsergebnisse" + 0.000*"credit"')
2	(2, '0.011*"investor" + 0.011*"collateralized" + 0.010*"capacity" + 0.009*"rückversicherung" + 0.008*"einhellig" + 0.007*"rückversi" + 0.006*"meisch" + 0.006*"rückversicherer" + 0.005*"rückver" + 0.005*"sinken"')
3	(3, '0.015*"informationsverarbeitung" + 0.013*"hochwasser" + 0.013*"clarke" + 0.012*"ions" + 0.011*"ars" + 0.010*"sv" + 0.009*"solutions" + 0.009*"schulungs-" + 0.009*"einsatzkräfte" + 0.009*"aufenthaltsort"')
4	(4, '0.026*"step" + 0.026*"metabolon" + 0.014*"efre" + 0.013*"forschungszentrum" + 0.011*"sustainable" + 0.011*"polyurethane" + 0.011*"erneuerbare energien" + 0.010*"reststoff" + 0.010*"energietechnik" + 0.009*"renewable"')
5	(5, '0.000*"ł" + 0.000*"aufsuchen" + 0.000*"std" + 0.000*"credit" + 0.000*"¥" + 0.000*"nationalität" + 0.000*"angeführt" + 0.000*"öffna" + 0.000*"ıı" + 0.000*"silqua"')
6	(6, '0.021*"maler" + 0.018*"cranach" + 0.015*"rdert" + 0.012*"meisterwerk" + 0.012*"gemälde" + 0.011*"royal" + 0.010*"kunstpalast" + 0.010*"sohn" + 0.010*"andrew" + 0.010*"kooperatio"')
7	(7, '0.049*"#" + 0.026*"mediennutzung" + 0.025*"-befragung" + 0.024*"praxisstudium" + 0.020*"nennung" + 0.017*"praxiszentrum" + 0.010*"innengruppe" + 0.008*"miglichkeit" + 0.008*"¥" + 0.007*"messenger"')
8	(8, '0.000*"ects" + 0.000*"$" + 0.000*"credit" + 0.000*"lv" + 0.000*"kulturwissenschaft" + 0.000*"#" + 0.000*"methods" + 0.000*"sws" + 0.000*"stadtteil" + 0.000*"rückversicherung"')
9	(9, '0.027*"risikomanagement" + 0.010*"vage" + 0.010*"berichterstattung" + 0.009*"aggregation" + 0.008*"faris" + 0.008*"-und" + 0.008*"bewertungsmethode" + 0.008*"rm" + 0.007*"beziehungsweise" + 0.006*"risikomanagementsystem"')
10	(10, '0.019*"gefahrenabwehr" + 0.014*"rettungsingenieurwesen" + 0.013*"irg" + 0.013*"routing" + 0.012*"esri" + 0.011*"geography" + 0.008*"masterstudent" + 0.008*"marz" + 0.008*"rohr" + 0.007*"oraussetzung"')
11	(11, '0.000*"lv" + 0.000*"ects" + 0.000*"obligatory" + 0.000*"dolmetsch" + 0.000*"sws" + 0.000*"stauraum" + 0.000*"konferenzdolmetsch" + 0.000*"leistungspunkt" + 0.000*"summer" + 0.000*"lotsen"')
12	(12, '0.020*"jah-" + 0.008*"wikis" + 0.008*"fraktion" + 0.006*"wunder" + 0.006*"referentenentwurf" + 0.006*"masterstudiums" + 0.004*"verabschiedet" + 0.004*"wissensmanagement" + 0.004*"stel-" + 0.004*"landtag"')
13	(13, '0.025*"zertifikatskurs" + 0.018*"selbstlernphase" + 0.018*"zbiw" + 0.016*"können" + 0.015*"xprtn" + 0.010*"live" + 0.009*"arbeitsaufwand" + 0.007*"zzgl" + 0.007*"zeitstunde" + 0.007*"gudrun"')
14	(14, '0.012*"studierendenwerk" + 0.000*"zutreffend" + 0.000*"teilprojekt" + 0.000*"zufrieden" + 0.000*"top" + 0.000*"grenzüberschreitend" + 0.000*"selbstverständnis" + 0.000*"entwicklungspartnerschaft" + 0.000*"unentschieden" + 0.000*"lernend"')
15	(15, '0.000*"-innen" + 0.000*"credit" + 0.000*"ł" + 0.000*"isbn" + 0.000*"workload" + 0.000*"öffnung" + 0.000*"turnus" + 0.000*"geflüchtet" + 0.000*"prüfungsform" + 0.000*"sws"')
16	(16, '0.030*"session" + 0.028*"kompetenzorientierung" + 0.020*"kompetenzentwicklung" + 0.019*"kompetenzorientiert" + 0.016*"zung" + 0.013*"kompe-" + 0.011*"entschei-" + 0.011*"för-" + 0.011*"-tisch" + 0.011*"kompe"')
17	(17, '0.000*"bauinformatik" + 0.000*"credit" + 0.000*"-beratung" + 0.000*"zutreffend" + 0.000*"steps" + 0.000*"ð" + 0.000*"leistungsbezug" + 0.000*"stadtteil" + 0.000*"programmierung" + 0.000*"prüfungsform"')
18	(18, '0.024*"cold" + 0.020*"mashing" + 0.015*"food" + 0.015*"extract" + 0.015*"°c" + 0.011*"kit" + 0.009*"-dem" + 0.009*"composition" + 0.009*"fig" + 0.009*"malt"')
19	(19, '0.015*"current" + 0.013*"power" + 0.009*"concern" + 0.008*"core" + 0.007*"dc" + 0.007*"inductor" + 0.007*"ac" + 0.007*"without" + 0.007*"flux" + 0.006*"pa"')
20	(20, '0.033*"exp" + 0.020*"pensionsfond" + 0.018*"stochastisch" + 0.015*"rente" + 0.015*"gl" + 0.014*"asset" + 0.012*"variante" + 0.011*"liability" + 0.011*"rentenanpassung" + 0.010*"sterblichkeit"')
21	(21, '0.000*"credit" + 0.000*"dual" + 0.000*"lernergebnisse" + 0.000*"turnus" + 0.000*"prüfungsform" + 0.000*"bestandene" + 0.000*"lehrformen" + 0.000*"kontaktzeit" + 0.000*"modulbeauftragte" + 0.000*"wahlpflichtfach"')
22	(22, '0.071*"reise" + 0.017*"quo" + 0.016*"inklusiven" + 0.015*"abb" + 0.015*"jugendreise" + 0.014*"inklusiv" + 0.013*"halt" + 0.011*"freizeit" + 0.009*"äh" + 0.009*"behin"')
23	(23, '0.009*"doi" + 0.009*"chemical" + 0.009*"kurzfassung" + 0.008*"humanities" + 0.007*"archivierung" + 0.006*"heal" + 0.006*"diplomtag" + 0.005*"rel" + 0.005*"diplomandinnen" + 0.005*"diplomanden"')
24	(24, '0.000*"credit" + 0.000*"psso" + 0.000*"studiensemester" + 0.000*"sws" + 0.000*"modulnummer" + 0.000*"workload" + 0.000*"kontaktzeit" + 0.000*"prüfungsform" + 0.000*"pflichtfach" + 0.000*"endnote"')
25	(25, '0.016*"glas" + 0.013*"pin" + 0.009*"ingenieurnachwuchs" + 0.008*"facette" + 0.008*"brillenfassung" + 0.006*"gst" + 0.006*"brillenglas" + 0.005*"nut" + 0.005*"rand" + 0.005*"pöpperl"')
26	(26, '0.020*"ausfall" + 0.016*"künstlich" + 0.015*"krankenhaus" + 0.014*"gesundheits-" + 0.013*"aufrechterhaltung" + 0.012*"künstliche intelligenz" + 0.012*"cities" + 0.011*"gesundheitswesen" + 0.010*"großflächig" + 0.009*"teilvorhaben"')

Das Ergebnis ist, wie in der unteren Abbildung zu sehen, eine Liste von Topics mit der dazugehörigen Topic ID. Diese Liste steht repräsentativ für die vorhandenen Inhalte in unseren Dokumenten. Nun fehlt nur noch der Schritt, die vorhandenen Topics den Dokumenten zuzuordnen.

	Topic ID	Term
2	1	Mentoring
3	2	Kapital
4	3	Unternehmen
5	7	Animation
6	9	Thermodynamik
7	11	Handel, Globalisation
8	12	Vermessung
9	13	Dispersion
10	14	Forschungsdaten
11	16	Immersion
12	22	Medien
13	23	Inklusion,Bildung
14	26	Pharmazeutik
15	27	Architektur, Philosophie
16	29	Hybrid,Kinetik,Energiespeicher,Fahrzeugentwicklung
17	30	Statistik
18	40	Ökologie
19	41	Mechanik
20	43	Restaurierung
21	46	Textil
22	48	Informationswissenschaften,Kommunikationswissenschaften
23	55	Politik
24	57	Elektronik
25	62	Denkmal

Dazu wurde die Datei „topics_distr_selected_whole_500.xlsx“ verwendet, die ebenfalls durch das LDA-Team zur Verfügung gestellt wurde. In diesem Dokument stehen alle Verbindungen von Topics und Dokumenten. Um die Arbeit zu verringern wurde ein kleines Pandas-Skript geschrieben, bei dem man einfach die gewünschte Topic ID in das Skript einfügt und anschließend die Dokument-IDs von allen Dokumenten erhält, in denen das Topic vorkommt. Das Skript ist in unserem Github-repository im Ordner „Finaler Ansatz“ (TopicDocConnection.ipynb) abgelegt.

Im nächsten Schritt würde die Hierarchisierung der Topics behandelt werden, um sie am Ende für das Frontend-Team zur Verfügung zu stellen. Die Hierarchisierung in Form eines Thesaurus könnte dann verwendet werden, um auf der UI durch die verschiedenen Themengebiete zu klicken und so zu den Experten zu gelangen. Diesen Schritt haben wir leider nicht abschließend behandeln können.

6.3. Herausforderungen, Probleme und Zwischenergebnisse

Durch den automatischen Ansatz ist es nicht möglich die Terme inhaltlich zu erschließen. Das heißt es sind maximal drei Hierarchisierungsstufen möglich, was nur einen sehr begrenzten Browsingansatz entspricht. Für eine tiefere Hierarchisierung ist eine inhaltliche Erschließung der Lehrgebiete notwendig. . Außerdem ist das Browsing über Fakultäten und Institute nicht besonders intuitiv, wodurch die Usability eingeschränkt ist. Das heißt das Browsing Interface grenzt sich aktuell nicht deutlich von der TH Köln Website ab. Auch dort ist eine Suche von Kandidaten möglich, indem über die TH Köln Website nach einer passenden Fakultät gesucht wird. Auf der Seite der Fakultät ist dann eine Liste an Instituten vorhanden, die wiederum eine Liste an Kandidaten beinhaltet. Aus diesen Gründen wurde der Ansatz verworfen und durch eine intellektuelle Erschließung der LDA Topics ersetzt.

Bei der intellektuellen Erschließung mithilfe der LDA Topics ist es ebenfalls zu Herausforderungen gekommen. Das erste Problem besteht darin, den Topics aus dem LDA-Modell inhaltlich korrekte und in sich konsistente Bezeichnungen zu geben. Mit dem darauffolgenden Schritt der Hierarchisierung musste also eine Homogenität der Begriffe vorhanden sein. Da der Dokumente Korpus und die daraus resultierenden LDA-Topics so vielfältig sind, ist eine Heterogenität der erschlossenen Topics nicht möglich, ohne dabei wichtige Informationen zu verlieren.

7. Expertenempfehlungen

7.1. Expert Ranking

Bevor das Expertenranking durchgeführt werden kann, müssen zunächst die Experten den Dokumenten und damit den Topics, zugeordnet werden. Sämtliche hier beschriebenen Vorgänge sind im Jupyter Notebook https://github.com/DIS-KD-Project/UI/blob/main/experts_for_topics/experts_for_topicid.ipynb zu finden.
Hierfür wurden unterschiedliche Datenquellen als Pandas Dataframe eingelesen:

- die LDA-generierten Topics
- die Topic-Dokument-Zuordnung
- die Dokument-Personen Zuordnung
- weitere Personen-Informationen

Durch die vorliegende Dokument-Personen und Topic-Dokument-Zuordnungen konnten wir mit Hilfe einfacher Joins der Dataframes eine Topic-Personen-Zuordnung generieren.

Das Expertenranking basierte dann auf der (ebenfalls aus dem LDA Modelling) prozentualen Zuordnung eines Topics zu den Dokumenten (Scores zwischen 0 und 1). Basierend auf der Beobachtung, dass dort bei Projekten beispielsweise die Projektleiter aufzufinden sind, wurden die Scores mit einer 1,5-fachen Gewichtung multipliziert, sollte die Person im Dokument in der linken Infobox aufgetaucht sein.
Darüber hinaus musste eine Gruppierung vorgenommen werden, da eine Person mehrfach als Experte zu einem Topic aufgeführt werden kann. Dies passierte immer dann, wenn einem Topic mehrere Dokumente zugeordnet sind, in denen diese Person auftaucht. Hier wurde dann der jeweils höchste Score genommen.
Zu guter Letzt werden die einzelnen Scores jeweils durch den höchsten auftretenden Score dividiert, um Scores zwischen 0 und 1 zu erhalten, welche dann vereinfacht gerankt werden können.

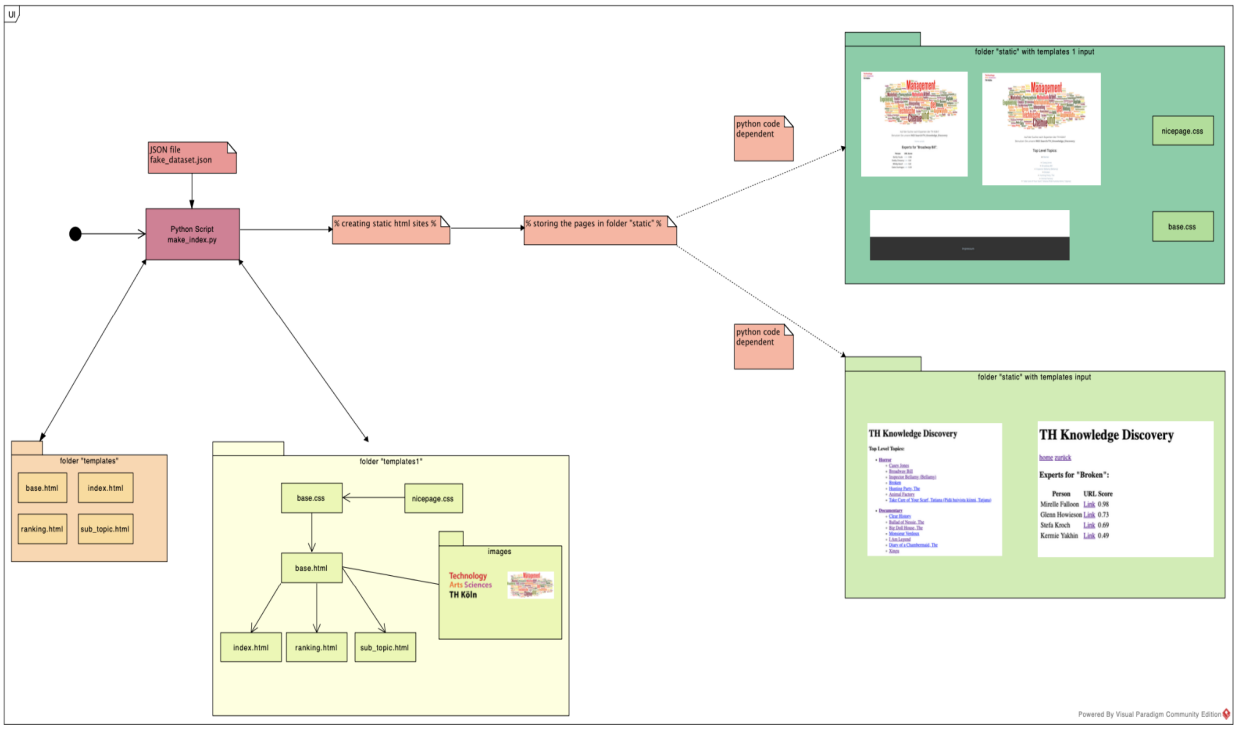
Diese Vorgehensweise wurde für jedes einzelne Topic wiederholt und die Ergebnisse in ein Dataframe geschrieben. Da nicht alle Personen die auf Topics gematcht werden auch als Experten aufgeführt werden sollen, wurde ein Threshold von 0.4 eingeführt. Nur Personen, die ein höheres Ranking besitzen, werden als Experte in den Output übergeben.
Schlussendlich wurde der Output als JSON gespeichert und an die UI übergeben.

Mögliche Optimierungsansätze sind hier insbesondere das Ranking an sich, welches deutlich professionalisiert werden sollte. Hier könnten noch weitere Informationen zu den Personen einfließen, um bspw. Professoren und Professorinnen höher zu ranken. Zusätzlich ist auch die

“Normalisierung”, mithilfe einer Division durch den höchsten Score pro Topic, nicht optimal. Hier sollte beispielsweise auf eine Min-Max-Normalisierung ausgewichen werden.
Sobald die Topics mithilfe des intellektuellen Ansatzes auch unter einem Begriff zusammengefasst sind, müssen auch diese Begriffe in den Output geschrieben werden.

7.2. Das Frontend-System

7.2.1. Aufbau und Funktionsweise



Mit Blick auf Kapitel 2.2 wurde bei der UI das gewünschte Browsing Modell zur visuellen Umsetzung gebracht.

Die UI bzw. deren Generierung besteht aus mehreren Bestandteilen, welche im Folgenden beschrieben und anhand Abbildung X nachvollzogen werden können:

Das Pythonscript make_index.py ist für die Generierung der statischen HTML-Seiten verantwortlich und erhält sowohl festen Input in Form einer JSON-Datei **(fake_dataset.json)**, als auch dynamischen, durch die Umbenennung eines Ordnernamens innerhalb des Codes. Das Script basiert auf der Bibliothek jinja2 und ist als Web Template Engine zur Erstellung von Vorlagen bekannt.[1] Damit die Entwicklung der UI bereits zu einem frühen Zeitpunkt starten konnte, wurde bei den unechten Daten der **fake_dataset.json** bereits auf eine Datenstruktur geachtet, die sich als Browsing Modell eignet und zu einem späteren Zeitpunkt durch die Daten der finalen Expertensuche ersetzt werden kann. Diese bestehen aus Genres auf der

[1] Jinja, 2021. [online]. [Zugriff am: 10.06.2021]. Verfügbar unter: https://jinja.palletsprojects.com/en/2.11.x/

ersten Ebene, Filmtitel auf der zweiten Ebene und auf der dritten Ebene die Schauspieler mit den berechneten Scores.

```
{% extends "base.html" %}
{% block title %}Knowledge Discovery{% endblock %}
{% block page_content %}

<a href="../index.html">home</a>
<a href="{{topic}}.html">zurück</a>
    <h3>Experts for "{{sub_topic}}":</h3>

<table>
  <tr>
    <th>Person</th>
    <th>URL</th>
    <th>Score</th>
  </tr>

    {% for person in ranking %}
    <tr>
      <td>{{person.name}}</td>
      <td><a href="{{person.url}}">Link</a></td>
      <td>{{person.score}}</td>
    </tr>
    {% endfor %}
</table>
{% endblock %}
```

Der Ordner „templates“ enthält vier verschiedene HTML-Dateien, welche zur Darstellung des JSON-Datasets benutzt werden. Die ***„index.html“***, die ***„ranking.html“*** und die ***„sub_topic.html“*** erben die HTML-Elemente der ***„base.html“*** und werden im Code, mit den Daten aus dem Script gefüllt. Wie in Abbildung X ersichtlich, wird mit ***{% extends "base.html" %}*** der Code von ***„base.html“*** geerbt und mit weiteren HTML-Elementen und den Daten versehen.

Der Ordner „templates1“ enthält die gleichnamigen HTML-Dateien, wie der ***„templates“*** Ordner. Der größte Unterschied liegt dabei darin, dass die ***„base.html“*** mit weiteren Code-Zeilen aufbereitet ist und Elemente einer CSS-Datei enthält, sodass dem User eine bessere UX verschafft werden kann. Die ***„base.css“*** wird als aktives Stylesheet benutzt, während die ***„nicepage.css“*** einzelne Elemente an diese vererbt.

Das Pythonscript make_index.py kann über die Konsole/Terminal ausgeführt werden und erstellt die Seiten. Es werden drei verschiedene Ebenen mit den statischen Seiten dargestellt bzw. generiert. Eine generierte Seite ***(„index.html“)*** umfasst alle Genres mit den dazu eingeteilten Filmen. Mehrere andere Seiten ***(sub_topic_horror.html/sub_topic_comedy.html etc..)*** umfassen nur ein Genre und die dazu eingeteilten Filme. Zuletzt gibt es zu jedem einzelnen Film eine statische HTML-Seite, in der den Schauspielern ein Score gegeben wird, sowie eine Verlinkung zur TH Köln Seite. Unabhängig von der Ordnerauswahl im Script für die Generierung, wird die ***„index.html“*** auf der ersten Ebene im Projektordner abgelegt, während die anderen Seiten in ***„static“*** gespeichert werden.

Je nach Veränderung des Codes im Script, werden entweder statische Seiten ohne Design generiert ***(“templates“)*** oder es wird auf die visuell bessere Lösung mit CSS-Dateien und verschiedenen Bildern zurückgegriffen ***(„templates1“).***

7.2.2. Alles im Auge des Betrachters – Die Designentscheidung

Wie in 6.2.1 herausgestellt, werden durch das Script zwei verschiedene Modelle der Anwendung angeboten. Eine einfache Lösung ohne weiteren Gebrauch von Bildern und CSS-Dateien, sowie eine Lösung mit eben diesen Inhalten.

Die Abbildung X zeigt nochmal deutlich, dass die linke Version mit CSS- und Bildanbindung, eine deutlich lebhaftere Nutzung verspricht, als die rechte Version. Wie erwähnt, werden von der “nicepage.css” Elemente in die “base.css” vererbt, welche unter anderem dafür zuständig sind, dass sich das TH Köln Logo in der linken oberen Ecke befindet und sich die Wordcloud an der angegebenen Position bewegt. Die Hintergrundfarben, sowie der schwarze Footer der Seite entstammen aus der nicepage.css. Die weiße Hintergrundfarbe, sowie die grauen/schwarzen Farbtöne der Texte, soll dem User eine software-ergonomisch gute Lösung bereitstellen, ohne dass zu viele CSS-Elemente die User-Experience während der Expertensuche stören.

Neben den Anpassungen beim Design wurden noch hilfreiche Verlinkungen im Code hinzugefügt. Das TH Köln Logo verlinkt auf die Startseite der TH Köln, während im Footer eine Verlinkung zum offiziellen Impressum der TH Köln gegeben ist. Wie gehabt, werden diese im HTML mit href angesprochen und sprechen für sich selbst.

7.2.3. Herausforderungen, Probleme und zukünftige Arbeitspakete

Die größte Herausforderung im Front-End bestand in erster Linie daraus, aus den verschiedenen Techniken für diesen Bereich eine passende für die Anwendung zu finden. Mit der von Herrn Keller eingebrachten Idee mit jinja2 und dem Imput als JSON-Format konnte ein großer Meilenstein erreicht werden, schließlich gab es eine lauffähige Anwendung mit randomisierten Daten. Das Einbringen einer JSON-Datei war bedacht und zukunftsweisend, schließlich ist die Key-Value-Eigenschaft von JSON, bei diesem Projekt gefüllt mit Themenfeld und Experten, sehr gut zu vereinigen.

Problematische Situationen gab es im Projektverlauf nicht. Die UI konnte z.T. vernachlässigt werden, da andere Teileinheiten des Projektes vorrangig fokussiert wurden, damit sich im Front-End etwas bewegen konnte. Etwaige Probleme lassen sich jedoch mit den zukünftigen Arbeitspaketen erahnen.

Die zukünftigen Arbeitspakete sehen wie folgt aus:

-Aktualisierung der Wortwolke

Damit die Wortwolke auf den HTML-Seiten einen aktuellen Stand hat, sollte zu gegebenem Zeitpunkt eine neue generiert werden, unter Berücksichtigung einer aktuellen Liste der kompletten Topics.
Wie? Wordle wurde für die erste Wortwolke genutzt und ist am Markt frei verfügbar.

-Hinterfragung der statischen HTML Seiten

Je nach Fall sollte evaluiert werden, ob man sich auch weiterhin auf statische Seiten berufen möchte oder eine dynamische Methode wählt. Ein aktuelles Zugreifen auf die Datenbasis (MYSQL) wäre hier möglich und könnte durch POST/GET gelöst werden.
Die Aktualisierung der statischen Seiten würde entfallen, somit wären Zeit- und Platzprobleme nicht gegeben.
Wie? Einbindung von PHP mit POST und GET. In Bezug auf den Studiengang wäre auch Python Flask möglich (Frontend: HTML+Python, Backend: Python, SQL)

-Entfernung von CSS- und Bilderredundanzen

Wie in Kapitel 6.2.1 deutlich wurde, wird die ***„index.html“*** auf der ersten Ebene angezeigt, während die anderen generierten Seiten im ***„static“-***Ordner landen. Da sich alle generierten Seiten auf das gleiche CSS, sowie die gleichen Bilder berufen, sind diese sowohl einmal auf der ersten Projektebene vorhanden, als auch im genannten Ordner.
Wie? Veränderung der Pfadangabe zu einem relativen Pfad, wobei die verschiedenen Ebenen berücksichtigt werden müssen. Ggf. kann der Index auch im Ordner mit den anderen Seiten gespeichert werden.

-XML Midos Transfer

Die im Projektordner hinterlegte XML-Datei “thes_test” wurde vom Team X aus Kapitel X erstellt. Die Datei stammt aus dem Export von Midos Thesaurus und enthält eine erste kleine Struktur eines intellektuell angefertigten Thesaurus.
Die Ober- bzw. Unterbegriffe sind an die Themen der TH Köln angelehnt und sollen für den Output auf den HTML-Seiten benutzt werden, somit also die JSON-Datei ersetzen.
Wie? Veränderung des Scripts, damit XML-Input akzeptiert wird. Alternativen wären die Umwandlung der XML zu JSON und daraufhin eine Bereinigung.

9. Reflektion und Ausblick

Im Rahmen des Projektes Research Knowledge Discovery (RKD) wurde ein Recommender-System aufgebaut, dass eine Alpha Version mit Ausbaupotential darstellt.

Es wurden Topics generiert, die noch viel Optimierung benötigen und ein Expertenranking, dass durch Hilfe eines intellektuellen Erschließungsansatz ebenfalls weiter ausgebaut werden kann. Unter dem Browsing-Ansatz wurde ein Frontend gestaltet, dass ein erstes, beispielhaftes Ergebnis für die Nutzung des Recommender-Systems zeigt.

Retrospektiv startete das Projekt mit einem geringen Rechercheaufwand, da die ersten wissenschaftlichen Artikel (im Folgenden Paper genannt) mit grundlegenden Inhalten zum Projekt bereits zur Verfügung gestellt wurden. Der Vorteil daran war, dass die Projektgruppe potenziell auf dem gleichen Wissensstand war, aber bei der Durchführung wurde sich rückblickend zu wenig an den Papern orientiert.

Bei der Ansatz Entscheidung wurde das Browsing-Modell gewählt, was auf der TH Köln Website ohne Data Science Methoden in ähnlicher Form abgebildet ist. Hier sind die verschiedenen Kategorien wie Fakultät, Institut und weitere zu finden. Über die Fakultäten und Lehrgebiete könnten so manuell Experten herausgefiltert werden. Unser Browsing-Modell gestaltet sich durch die automatisch erschlossenen Topics der LDA-Gruppe (Latent Dirichlet Allocation). Während des Pre Processings war die Parallelisierung von Programmcode unbekannt und daraus resultierte, dass das Bereinigen der Daten deutlich mehr Zeit als erwartet in Anspruch nahm. Ebenfalls waren die Grundlagen für die Topicgenerierung mittels LDA noch nicht im Studium behandelt. Es stellte sich im Nachhinein heraus, dass LDA als Topic-Identifizierung der Dokumente nur in einigen Fällen sinnvoll war und sich für das RKD-Projekt wohl als der falsche Ansatz zur Generierung der Topics zeigte.

Neben dem LDA-Ansatz wurden Topics über eine intellektuelle Erschließung erfasst. Diese Form war präziser als eine automatische Erschließung aber auch deutlich zeitaufwendiger. Da dieser Ansatz erst sehr spät behandelt wurde, war eine erfolgreiche Umsetzung zeitlich nicht mehr möglich. Außerdem konnte hier ein Missverständnis festgestellt werden, da diese Erschließung immer auf den durch LDA generierten Topics beruhte und nicht über die Rohdaten des Pre Processings erfolgte.

Es wurde ein Expertenranking beruhend auf einer prozentualen Zuordnung von einem Topic zu den Dokumenten erstellt. Der Output des Rankings diente dem Frontend als Datengrundlage, für den dort visualisierten Browsing-Ansatz.

Zu Beginn des Projektes verfolgte die Projektgruppe noch einen weiteren Ansatz, der auch externe Webseiten berücksichtigen sollte. Die hierfür notwendige Modifizierung des Scrapers nahm sehr viel Arbeitszeit in Anspruch und wurde zum Ende verworfen. Drei weitere Aspekte sollten noch reflektiert werden:

Der Wechsel der Kommunikations- und Arbeitsplattformen stand im Konflikt zum gesamten Zeitfaktor des Projektes und der gewünschte Effekt eines vermehrten Austauschs innerhalb der Projektgruppe konnte allein dadurch nicht optimiert werden.

Mit Start des Projektes war noch keine entsprechende Infrastruktur durch die TH Köln bereitgestellt, um das Projekt vollumfänglich umzusetzen. Das sehr mühselige Einfordern raubte der Projektgruppe ebenfalls viel Arbeitszeit.

Zuletzt fehlte im Rahmen des Projektmanagements die übergeordnete Leitung durch die Dozierenden. Auch wenn sehr wertvolle Kenntnisse durch die eigenständige Arbeit geschaffen worden, so hätte sich eine klare Linie zielführender für das Projekt gezeigt.

Schlussendlich schaffte das Projekt ein Bewusstsein über die Wichtigkeit von gut generierten Topics und deren Erschließungsmethoden und vermittelte uns erste Erfahrungen und Expertise in einem Data Science Projekt.

Mit Blick auf die Möglichkeiten zur Weiterentwicklung, sollte das Vorhandene vorerst ausgeblendet werden. Noch ist nicht geklärt welche Anfragen genau gestellt werden und wie genau angefragt wird. Es könnte durch Interviews bzw. das Befragen der entsprechenden Personen die zu Experten an der TH Köln angefragt werden eine Analyse erfolgen, aus der die Anfrage-Terme gesammelt werden. Zur bereits vorhandenen Struktur, in der aktuell über “Lehrgebiete” gesucht werden kann (Personenverzeichnis der TH Köln) entsteht so eine neue Struktur der thematischen Suche über einen “TH Köln Experten Thesaurus”.

Parallel könnte auch ein AD-Hoc Modell entwickelt werden, um die dort eingehenden Anfragen genauer zu verstehen und sichtbar zu machen. Das Suchverhalten der Anfragenden müsste also interpretiert und analysiert werden. Diese Interpretation würde auch die Grundlage für einen “TH Köln Experten Thesaurus” bieten.

Mit dem Wissen aus den Interviews und dem AD-Hoc-Modell könnte dann eine entsprechende Anpassung unseres vorhandenen Systems vollzogen werden. Daraus würde eine optimierte Indexierungspipeline resultieren, welche folglich verbesserte Terme nutzt.

Nach erfolgreichem Bearbeiten der vorher beschriebenen Schritte müsste die erstellte Website (Frontend) in die TH Köln Website implementiert werden. Ein wichtiger Aspekt wäre hier ein zusammenhängendes Backend im Sinne von automatisch ausgeführten Aufgaben, welche die zuvor beschriebenen einzelnen Schritte der Gruppen systematisch abarbeitet, zu erstellen. Das heißt, es würde beispielsweise immer Montags das Scraping durchlaufen, Dienstags das Preprocessing usw. Das Frontend würde so personenunabhängig gefüllt werden.

10. Literaturverzeichnis

Bischkopf, Ruben. (2020). Crawl Your Prof – Fact-Crawling von Hochschulseiten (Bachelorarbeit). Technische Hochschule Köln.

Blei, D., Ng, A. & Jordan, M. (2003). Latent Dirichlet Allocation. University of California, Berkley. https://ai.stanford.edu/~ang/papers/nips01-lda.pdf

Calzolari, N., European Language Resources Association, Mortensen, D. R. D. S., Dalmia, S. & Littell, P. (2018). LREC 2018. European Language Resources Association.

Campus IT. (o. D.). TH Köln. Abgerufen am 23. Juni 2021, https://www.th-koeln.de/hochschule/campus-it_3866.php

DIS-KD-Project. (2021).Miro KD-Projektmanagement Board. https://miro.com/welcomeonboard/VzVKaWhOd1pqWHk4a1JITWc2VVR0MzdnTVdIOGM2WEN0R2JIdzQ1aWhwS1h4MllVbnVNazlZbHZ4SnBhZk04bHwzMDc0NDU3MzUxODg1MTUxMTU

DIS-KD-Project. (2021). Kanban Board. GitHub. https://github.com/orgs/DIS-KD-Project/projects/2

DIS-KD-Project. (2021). LDA_Topics/lda_functions. GitHub. https://github.com/DIS-KD-Project/LDA_Topics/blob/main/lda/lda_functions.py

DIS-KD-Project. (2021). LDA_Topics/optimal_num_topics. GitHub. https://github.com/DIS-KD-Project/LDA_Topics/blob/main/lda/optimal_num_topics.py

DIS-KD-Project. (2021). LDA_Topics/build_model. GitHub. https://github.com/DIS-KD-Project/LDA_Topics/blob/main/lda/build_model.py

DIS-KD-Project. (2021). LDA_Topics/unseen_docs. GitHub. https://github.com/DIS-KD-Project/LDA_Topics/blob/main/lda/unseen_docs.py

Dua, S. (2020, 18. Mai). Text Classification using K Nearest Neighbors - Towards Data Science. Medium. https://towardsdatascience.com/text-classification-using-k-nearest-neighbors-46fa8a77acc5

HanTa. (o. D.). HanTa. https://pypi.org/project/HanTa/

Hedgpeth, R. H. (2020, 25. Juni). How to connect Python programs to. MariaDB. https://mariadb.com/de/resources/blog/how-to-connect-python-programs-to-mariadb/

ILX Group. (2021). Prince2. https://www.prince2.com/de/what-is-prince2#:~:text=PRINCE2%20ist%20eine%20prozessbasierte%20Methode,wird%20weltweit%20anerkannt%20und%20verwendet.

langdetect. (o. D.). PyPI. https://pypi.org/project/langdetect/

Lin, S., Hong, W., Wang, D. & Li, T. (2017). A survey on expert finding techniques. Journal of Intelligent Information Systems, 49(2), 255–279. https://doi.org/10.1007/s10844-016-0440-5

Mattermost (2021). https://mattermost.com/

Natural Language Toolkit — NLTK 3.6.2 documentation. (o. D.). nltk. https://www.nltk.org/

owncloud - pyocclient. (o. D.). GitHub. https://github.com/owncloud/pyocclient

Prabhakaran, S. (2020, 17. September). Topic modeling visualization - How to present results of LDA model? | ML+. Learn Applied Data Science. https://www.machinelearningplus.com/topic-modeling-visualization-how-to-present-results-lda-models/#6.-What-is-the-Dominant-topic-and-its-percentage-contribution-in-each-document

Řehůřek, R. (2021, 29. April). Gensim: topic modelling for humans. Radimrehurek. https://radimrehurek.com/gensim/models/ldamulticore.html

Real Python. (o. D.). Async IO in Python: A Complete Walkthrough. https://realpython.com/async-io-python/

Röder, M., Both, A. & Hinneburg, A. (2015). Exploring the Space of Topic Coherence Measures. Agile Knowledge Engineering and Semantic Web. https://svn.aksw.org/papers/2015/WSDM_Topic_Evaluation/public.pdf

SQLAlchemy - The Database Toolkit for Python. (o. D.). SQLAlchemy. https://www.sqlalchemy.org/

Technische Hochschule Köln. (2021).Personen von A-Z. https://www.th-koeln.de/hochschule/personen_3850.php

World Cities Database | Simplemaps.com. (o. D.). Simplemaps. https://simplemaps.com/data/world-cities

Zoom Inc. (2021). https://zoom.us/de-de/meetings.html

Zyte (formerly Scrapinghub). (2008, 26. Juni). Scrapy. GitHub. https://github.com/scrapy/scrapy