UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

PALUNO
The Ruhr Institute for Software Technology

Institut für Informatik und Wirtschaftsinformatik (ICB)
Lehrstuhl für Software Engineering, insb. mobile Anwendungen
Prof. Dr. Volker Gruhn

Natural Language Processing zur Verbesserung von Suchfunktionen in der Software-Entwicklung

Bachelorarbeit

vorgelegt der Fakultät für Wirtschaftswissenschaften
der Universität Duisburg-Essen (Campus Essen) von

Leon Zimmermann
Laddringsweg 8
45219 Essen
Matrikelnummer: 3080384

Essen, den 11. Mai 2023

Betreuung:	Wilhelm Koop, Sascha Feldmann
Erstgutachter:	Prof. Dr. Volker Gruhn
Zweitgutachter:	???
Studiengang:	Angewandte Informatik - Systems Engineering (B. Sc.)
Semester:	10

Abstract

TODO

Zusammenfassung

TODO

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

NLP Natural Language Processing

1. Einleitung

Die Einleitung beschreibt die Struktur der Arbeit, sodass dem Leser gleich zu Beginn der rote Faden ersichtlich wird. Dazu wird im Kapitel Motivation zuerst erläutert, warum der Inhalt der Arbeit von Relevanz ist. Das Kapitel beschreibt das Problem, welches Softwareentwickler in der täglichen Arbeit mit Wissensdatenbanken haben. Das nächste Kapitel "Ziele der Arbeit" gibt eine Übersicht darüber, was diese Arbeit erreichen möchte. Das Kapitel erklärt, wie das Problem, welches in der Motivation beschrieben wurde, in dieser Arbeit systematisch gelöst wird. Dazu sollen vier Teilziele erreicht werden. Wie diese vier Teilziele erreicht werden wird in dem Kapitel "Vorgehensweise" erklärt. Das Kapitel beschreibt für jedes Teilziel kurz und knapp dessen Vorgehensweise. Im darauffolgenden Kapitel werden die Verwandten Arbeiten beleuchtet. Zuletzt wird die Arbeit in dem Kapitel "Abgrenzungen" inhaltlich abgegrenzt.

1.1. Motivation

Suchfunktionen sind in vielen Anwendungen vorhanden und können einen großen Einfluss auf die Erfahrung von Nutzern haben. Google ist ein Anbieter einer bekannten Suchfunktion. Sie wird verwendet, um im World Wide Web die passende Website für eine Suchanfrage zu finden. Die Suchanfrage besteht dabei meistens aus lediglich ein bis drei Keywords. Es ist nicht einfach für diese Keywords die Websites zu identifizieren, welche für einen Nutzer am passendsten ist.

Auch andere Arten von Anwendungen verwenden Suchfunktionen, darunter Wissensdatenbanken, wie beispielsweise Confluence. In Wissensdatenbanken werden Informationen auf Seiten gespeichert, welche über die Suchfunktion auffindbar sein sollen. Softwareentwickler haben bei der Arbeit mit Wissensdatenbanken, insbesondere bei der Arbeit mit Spezifikationen und Dokumentationen, oft das Problem, dass die gesuchten Inhalte nicht gefunden werden können. Das kann daran liegen, dass die gefundenen Ergebnisse zu spezifisch sind, z.B. wenn eine allgemeine Definition von Domänenobjekten gesucht wird, aber eine Spezifikation eines Use-Cases gefunden wird, in welchem das Domänenobjekt lediglich erwähnt wird. Außerdem haben manche Wissensdatenbanken Schwierigkeiten, die Semantik einer Suchanfrage zu verstehen. Wird also nach einem Suchbegriff, wie „deployment“ gesucht, dann werden keine Ergebnisse gefunden, welche Wörter, wie „rollout“ beinhalten, auch wenn es sich hierbei um den gleichen Prozess handelt. Softwareentwickler müssen also das genaue Wording in der Dokumentation kennen, um ein sinnvolles Ergebnis angezeigt zu bekommen, und es reicht nicht aus, ein thematisch ähnliches Wort in die Suche einzugeben. Ziel dieser Arbeit soll es sein, mithilfe von Wissen über Suchalgorithmen und Algorithmen aus dem

Natural Language Processing, diese Probleme zu lösen, sodass Softwareentwickler in Zukunft besser die Inhalte finden, nach denen sie tatsächlich gesucht haben.

1.2. Ziele der Arbeit

In der Motivation sind bereits Probleme genannt worden, die ein Softwareentwickler bei der Verwendung von Suchfunktionen in Wissensdatenbanken hat. Manchmal ist das Abstraktionslevel der gefundenen Informationen nicht das, welches sich der Softwareentwickler gewünscht hat. Manchmal kennt der Softwareentwickler nicht das genaue Wording, um die gewünschten Informationen zu finden. Ziel der Arbeit ist es diese Probleme zu lösen.

Zur Erreichung des Hauptziels der Arbeit werden folgende Teilziele definiert:

- **Teilziel 1:** Es werden Lösungsansätze zur Verbesserung von Suchfunktionen herausgearbeitet.
- **Teilziel 2:** Es werden Methoden für die Bewertung der Lösungsansätze herausgearbeitet. Damit wird die Frage beantwortet, wann eine Suchfunktion "gut" ist.
- **Teilziel 3:** Es wird die Theorie zur Implementierung der Lösungsansätze erläutert.
- **Teilziel 4:** Es werden die Lösungsansätze anhand der Evaluationsmethoden bewertet, mit einer bestehenden Suchfunktion verglichen und es wird ein Fazit gezogen.

1.3. Vorgehensweise

Die Vorgehensweise ist im Folgenden entsprechend der Teilziele in vier Schritte eingeteilt. Jeder Schritt in der Vorgehensweise addressiert ein Teilzeil.

Adressierung von Teilziel 1

Zunächst werden die Lösungsansätze herausgearbeitet indem die konkreten genannten Probleme weiter beleuchtet und unterteilt werden. Sie werden in mehreren Use-Cases formuliert.

Adressierung von Teilziel 2

Um die Lösungsansätze zu bewerten wird erörtert, wie die Performance einer Suchfunktion evaluiert werden kann. Die Lösungsansätze sollen auf ihre Effektivität geprüft werden. Die Effektivität zu bestimmen ist nur mit geeigneten Messmethoden möglich. Diese Effektivität soll mit einer bestehenden Suchfunktion verglichen werden.

TODO Was bedeutet Effektivität in diesem Kontext?

Adressierung von Teilziel 3

Die theoretische Implementierung der Lösungsansätze wird als nächstes erläutert. Dazu werden auch die theoretischen Grundlagen für die Implementierung behandelt. Außerdem werden verschiedene Arten von Suchfunktionen vorgestellt. Zusätzlich werden Methoden vorgestellt, um die Suche, unabhängig von der gewählten Methode zu verbessern. Und es werden Methoden zur Aufbereitung der Informationen vorgestellt. Damit werden die gesuchten Ergebnisse auf eine Art und Weise dargestellt, welche dem Softwareentwickler gleich die gewünschten Informationen liefert.

Adressierung von Teilziel 4

Um ein Fazit zu ziehen muss bewertet werden, ob die Lösungsansätze ihren Zweck erfüllen. Dazu wurden zunächst die Evaluationsmethoden erörtert. Diese werden in diesem Schritt angewandt. Die Evaluationsmethoden liefern Metriken anhand dessen die Performance der Lösungsansätze bewertet wird. Diese Performance wird verglichen mit einer bestehenden Suche. Damit kann im Fazit diskutiert werden, ob die Lösungsansätze ihren Zweck erfüllt haben.

1.4. Verwandte Arbeiten

Es gibt einige Arbeiten, welche die gleichen oder sehr ähnliche Probleme adressieren.

So wird in "Automatic Query Reformulations for Text Retrieval in Software Engineering" von Haiduc et. al. ein System zur Verbesserung von Suchanfragen vorgeschlagen. Ausgangspunkt für das Paper ist das Problem der Traceability zwischen Code und anderen Softwareentwicklungs-Artefakten. Traceability bedeutet, dass sich von einer Stelle im Code, auf die entsprechenden Stellen in anderen Artefakten zurückschließen lässt. Ein Anwendungsfall für eine solche Traceability-Funktionalität ist "Feature Location", also das finden der Spezifikation eines Features, wenn nur der Code vorhanden ist. Das System, welches von Haiduc et. al. vorgeschlagen wird, verwendet Query Reformulations, um die Traceability herzustellen. Query Reformulation bedeutet, dass das System den Softwareentwickler bei der Eingabe einer Suchanfrage zur Suche nach den passenden Artefakten unterstützt. Dazu gibt der Softwareentwickler zunächst eine Suchanfrage ein, und markiert diejenigen Ergebnisse, welche am relevantesten für ihn sind. Auf Grundlage der gewählten Ergebnisse und mithilfe eines Machine Learning Algorithmus werden nun Vorschläge für eine verbesserte Suchanfrage gemacht. Dabei gibt es verschiedene Strategien. Wenn der Softwareentwickler zu Beginn eine sehr lange Suchanfrage eingegeben hat, dann kann das System eine Reduktion der Suchanfrage vorschlagen. Hat der Softwareentwickler dagegen lediglich einen Suchbegriff angegeben, so kann das System eine Erweiterung der Suchbegriffe vorschlagen. Dazu greift das System auf Synonyme des eingegebenen Suchbegriffes zurück.

TODO Gegebenenfalls Anwendungsfälle aus den Verwandten Arbeiten ableiten: Feature Location, Bug Location, Traceability Link Recovery, Refactoring, Reuse.

In dem Paper "From Word Embeddings To Document Similarities for Improved Information Retrieval in Software Engineering" von Ye et. al. wird beschrieben, wie Word Embeddings dazu verwendet werden können, um Traceability zwischen Code und anderen Softwareentwicklungs-Artefakten herzustellen. Word Embeddings sind eine Datenstruktur, welche einem Wort einen Vektor in einem n-dimensionalen Raum zuweist. Anhand dieses Vektors kann die Ähnlichkeit zwischen Wörter beschrieben werden. Ähnliche Wörter haben eine geringe Distanz im n-dimensionalen Raum. Unähnliche Wörter haben eine hohe Distanz. Der Algorithmus, welcher die Ähnlichkeit der Wörter bestimmt, macht Gebrauch von der Distributional Hypthesis. Dieser besagt, dass Wörter, welche im gleichen Kontext verwendet werden, eine ähnliche Semantik besitzen. Hermit wird also die Ähnlichkeit der Wörter bestimmt. Dieses Verfahren wird nun sowohl auf den Code angewendet als auch auf die Softwareentwicklungs-Artefakte.

In dem Paper "Information Retrieval Models for Recovering Traceability Links between Code and Documentation" verwenden Antoniol et. al. einen ähnlichen Ansatz, wie Ye et. al. Auch hier werden Word Embeddings verwendet um Softwareentwicklung Artefakte gegen den Code zu matchen. Hier durchlaufen die Artefakte und der Code zwei verschiedene Pipelines. Die Wörter der Artefakte in natürlicher Sprache werden in lowercase umgewandelt. Anschließend werden Stoppwörter entfernt. Zuletzt werden Flexionen entfernt. Aus dem Code werden zunächst Identifier extrahiert. Identifier, welche mehrere Wörter unter Verwendung von CamelCase oder snake_case beinhalten, werden in die einzelnen Wörter aufgeteilt. Anschließend werden die Identifier auf die gleiche Art und Weise normalisiert, wie die Wörter der Softwareentwicklungs-Artefakte. Dann erfolgt sowohl für die Identifier als auch für die Wörter aus den Artefakten die Indizierung, also die Umwandlung in Word Embeddings.

1.5. Abgrenzungen

Diese Arbeit behandelt keine Large Language Models, wie GPT.

2. Auswahl von Use-Cases

In diesem Kapitel sollen Use-Cases ausgewählt werden, für welche später Lösungsansätze entwickelt werden. Die Use-Cases beschreiben die tatsächlichen Probleme, die Softwareentwickler bei der Verwendung von Suchfunktionen in Wissensdatenbanken haben. Zur Identifikation von Use-Cases wurden in Confluence testweise Sucheingaben gemacht. Es wurde das Erwartete Ergebnis gegen das tatsächlich gefundene Ergebnis abgeglichen. Aus diesem Verfahren ließen sich die Use-Cases ableiten. So ließ sich ableiten, dass manchmal nicht die gewünschten Ergebnisse gefunden werden, weil nicht das korrekte Wort in der Suche verwendet wurde. Stattdessen wurde ein Synonym oder thematisch sehr ähnliches Wort verwendet. Grund dafür ist, dass der Softwareentwickler nicht das genaue Wording des gewünschten Dokuments kennt, sondern lediglich das Thema des gesuchten Dokuments. Es wäre also hilfreich, wenn die Suche auch Dokumente finden würde, welche thematisch ähnlich zu der Sucheingabe sind. Dieser Use-Case wird in dem Kapitel "Semantik der Suche verstehen" erläutert. Außerdem sich ableiten, dass der Scope, in welchem der Softwareentwickler sucht variieren kann. Dieser Use-Case wird in dem Kapitel "Rollenspezifische Suchfilter" beschrieben.

In den Verwandten Arbeiten wurden bereits Arbeiten genannt, welche ähnliche Probleme lösen sollen. Diese Fokussieren sich vorallem auf die "Feature Location", "Bug Localization" und die Traceability zwischen Code und anderen Artefakten. Bei der Entwicklung eines neuen Features greift der Softwareentwickler also auf die entsprechende Spezifikation zurück. Dazu muss ihm bekannt sein, wo die Spezifikation zu finden ist. Nun muss er bei der Entwicklung darauf achten, dass er Best-Practices und Konventionen einhält, sowie die Qualitätsanforderungen. Eine Qualitätsanforderungen könnte dabei eine vereinbarte Testabdeckung der Software sein. Der Softwareentwickler muss also bei der Entwicklung eines neuen Features auch diese Informationen einfach finden können. Und ihm muss klar sein, an welcher Stelle im Code er den neuen Code einbauen sollte. Das ist Feature Location.

Wenn der Softwareentwickler gerade kein neues Feature implementiert, dann korrigiert er gerade möglicherweise einen Fehler in der Software. Um einen Fehler überhaupt zu identifizieren, muss aber zuerst wieder die Spezifikation herangezogen werden. Denn in der Spezifikation wird, wie bereits erwähnt, die gewünschte Funktionsweise der Software beschrieben. Damit wird auch definiert, was ein fehlerhaftes Verhalten ist, und was ein korrektes Verhalten ist. Wenn der Softwareentwickler nun ein Fehlerticket erhält, dann muss er die entsprechende Spezifikation zu diesem Fehlerticket finden können. Und idealerweise wird ihm durch die Suche sogar gleich die betroffene Stelle im Code angezeigt. Das ist Bug Localization.

2.1. Wahl des Abstraktionslevels

Oft ist im Voraus nicht klar, was man genau sucht. Man gibt einen Suchbegriff ein und möchte "sinnvolle" Informationen zu diesem Suchbegriff bekommen. Sinnvoll könnte eine Definition des Begriffs sein, oder ein Überblick darüber, in welchen Bereichen und Kontexten dieser Begriff vorkommt.

TODO

2.2. Semantik der Suche verstehen

Oft sucht ein Software-Entwickler nach einem bestimmten Bereich in der Spezifikation, aber kennt nicht das genaue Wording, welches auf der Seite verwendet wird. Beispielsweise könnte ein Softwareentwickler sich in ein neues Projekt einarbeiten, und möchte nun verstehen, wie er die bestehende Anwendung überhaupt in einer Testumgebung starten kann. Dann gibt er in die Sucheingabe der Wissensdatenbank soetwas ein, wie "deployment". Wenn nun die Seite, welche das Deployment erklärt, aber nicht genau dieses Wort enthält, dann wird die gewünschte Seite durch die Suche nicht gefunden. Es sei denn, es handelt sich um eine Semantische Suche. Eine Semantische Suche versteht Zusammenhänge zwischen verschiedenen Wörtern. Wenn also auf der Seite das Wort rollout verwendet wird, anstelle von deployment, dann versteht eine Semantische Suche, dass diese beiden Begriffe sehr ähnlich zueinander sind, und zeigt auch Suchergebnisse von Seiten an, welche das Wort rollout verwenden, auch wenn nach dem Wort deployment gesucht wurde.

2.3. Aktualität und Relevanz des Dokuments

Manchmal werden Blog-Einträge von unbekannten Benutzern angezeigt, welche sehr kurz sind. Die Überschrift ist zwar passend, aber der Inhalt des Textes nicht. Und der Text ist nur sehr kurz. Es scheint offensichtlich, dass für allgemeine Suchanfragen längere Texte als Ergebnis sinnvoller sind, da eine allgemeine Suchanfrage der Versuch ist, sich mit einem Thema vertraut zu machen. In diesem Fall sollten viele Informationen priorisiert werden, gegenüber kurzen Texten, welche die Lösung eines sehr spezifischen Problems erläutern.

Manchmal werden alte Seiten gefunden oder unfertige Seite. Es sollte erkannt werden, wenn eine Seite veraltet ist (ist markiert), oder wenn sie noch nicht fertig ist (durch Markierung oder Länge/Struktur des Textes)

TODO

2.4. Rollenspezifische Suchfilter

Eine Möglichkeit relevante Dokumente einfacher zu finden ist die Verwendung von Suchfiltern. Im Kontext einer Suche, welche auf Software-Entwickler zugeschnitten ist, ist es denkbar einen Suchfilter zu verwenden, welcher nach Software-

Informationen filtern. Also die Spezifikation und Dokumentation der Software, sowie How-To-Guides, Best-Practices und Code Konventionen. Außerdem ist ein Filter für Informationen zur Infrastruktur denkbar. Also Informationen darüber, wie die Software deployt wird, und wohin sie deployt werden kann. Außerdem Informationen über häufige Probleme beim Deployment. Dann ist auch ein Filter für Projektmanagement-Informationen sinnvoll. Dieser kann verwendet werden, um Informationen zum nächsten Software-Release zu finden, über die Teamaufteilung und Verantwortlichkeiten.

2.5. Information Extraction

Eine Suchfunktion ist im allgemeinen ein Information Retrieval System. Es wird eine Eingabe gemacht, und relevante Dokumente werden identifiziert und dem Nutzer angezeigt. Neben dem Information Retrieval gibt es Information Extraction. Information Extraction entnimmt aus den gefundenen Dokumenten genau die Informationen, welche für den Nutzer relevant sind. Es werden also nicht nur Dokumente gefunden, sondern gleich aufbereitet. Das ist vor allem dann praktisch, wenn der Nutzer nach einer Information sucht, welche in einem sehr großen Dokument zu finden ist. In diesem Fall ist es zwar schön, dass das System das Dokument findet, trotzdem muss der Nutzer sich mit der Suche nach der gewünschten Information innerhalb vom Dokument quälen. Information Extraction löst genau dieses Problem. Die Inhalte des Dokuments, welche relevant sind, können bei der Darstellung des Suchergebnisses als Kurztext dargestellt werden, um dem Nutzer gleich die gewünschten Informationen zu liefern.

TODO Erläuterung von Named Entity Recognition, Entity Linking, Parser etc. für die Umsetzung solcher Use-Cases

3. Evaluationsmethoden und -Kriterien

3.1. Features

Relevanz

Eine Suchfunktion muss nicht nur Ergebnisse für eine Such-Query finden, sondern diese auch darstellen. Bei der Darstellung der Ergebnisse ist die Reihenfolge besonders wichtig. Manchmal hat der Nutzer keine genaue Vorstellung davon, was genau er sucht. Er möchte Informationen zu einem Thema finden, aber ohne dabei bestimmte Seiten oder Inhalte im Hinterkopf zu haben. Nichtsdestotrotz möchte der Nutzer mit minimalem Aufwand so viele wichtige Informationen, wie möglich erhalten. Die Sortierung von Ergebnissen der Suche soll genau das ermöglichen. Nun ist zu klären, wie die Bestimmung der Relevanz von Suchergebnissen überhaupt funktioniert. Es muss bestimmt werden, anhand welcher Kriterien die Suchfunktion die Relevanz bestimmt. Für diese Kriterien gibt es wiederum unterschiedliche Quellen. Eine Quelle für Kriterien der Relevanz ist das Nutzerverhalten. Seiten, die besonders häufig besucht werden und auf denen lange verweilt wird, oder welche sehr häufig bearbeitet werden, sind gute Kandidaten für eine hohe Relevanz. Seiten, welche bei Nutzern praktisch in Vergessenheit geraten sind, scheinen nicht so wichtig zu sein, sodass die Relevanz für die Suche niedrig ist.

Clustering

Thematisch ähnliche Dokumente tendieren dazu relevant für die gleiche Suche zu sein. Aus diesem Grund ist es oftmals sinnvoll ähnliche Dokumente zu clustern, also zu gruppieren, und sie dem Nutzer als ein solches Cluster darzustellen. Alternativ kann auch ein Suchergebnis dargestellt werden und ähnliche Suchergebnisse werden angezeigt als „Ähnliche Suchergebnisse“.

Autovervollständigung

Autovervollständigung bedeutet, dass während der Eingabe von Suchbegriffen bereits Ergänzungen für die gesamte Eingabe vorgeschlagen werden. Das ist vor allem ein hilfreiches Feature in Fällen, wo der Nutzer nur eine grobe Ahnung davon hat, was er eigentlich sucht. Mithilfe von Vorschlägen durch die Suchmaschine selbst kann der Nutzer seine Eingabe spezifischer fassen als er es ohne Vorschläge schaffen würde, sodass die Suchergebnisse besser zu dem passen, was er eigentlich sucht.

Best-Prefix-Highlighting

Das ähnlichste Ergebnis zur Eingabe des Nutzers soll gehighlightet werden.

Synonyme

Synonyme bei Suche mitbeachten.

3.2. Qualitätskriterien

Interaktivität

Klassische Suchfunktionen sind so konzipiert, dass der Nutzer einen Suchbegriff eingibt, anschließend auf einen Submit-Button klickt und erst dann Suchergebnisse angezeigt bekommt. Der Nachteil dieser Vorgehensweise ist, dass der Nutzer erst spät Feedback darüber bekommt, ob er mit den eingegebenen Suchbegriffen auch Ergebnisse bekommt, die für ihn gerade sinnvoll sind. Das Qualitätskriterium Interaktivität gibt an, wie kurz die Zeit zwischen dem Eingeben von Suchbegriffen und der Anzeige von Ergebnissen ist. Eine Suchfunktion muss nicht unbedingt einen Submit-Button verwenden. Die Suche kann auch mit jedem ausgeschriebenen Keyword die Suche starten und Ergebnisse dafür anzeigen, oder auch nach jedem eingegebenen Buchstaben. Mit einer kürzeren Zeit zwischen Eingabe und Ergebnis, also einer höheren Interaktivität, weiß der Nutzer schneller, ob seine Eingaben sinnvoll sind, und bekommt damit schneller gute Suchergebnisse.

Geschwindigkeit

Die Geschwindigkeit einer Eingabe ist die Zeit zwischen dem Beginn der Suche und der Anzeige der ersten Antwort. Die Geschwindigkeit hat nichts mit der Interaktivität zu tun.

Darstellung

Wie gut kann man anhand der Zusammenfassung der Ergebnisse den Inhalt bestimmen.

Benutzerfreundlichkeit

Das System findet auch Ergebnisse für ähnliche Wörter. So sind bspw. Tippfehler kein Problem. Fuzzy-Search.

3.3. Precision, Recall und F-Maß

Zur Evaluation werden die Werte Precision und Recall gemessen. Der Precision-Wert ist das Verhältnis zwischen allen gefundenen Dokumenten und den gefundenen Dokumenten, die tatsächlich relevant sind. Der Recall-Wert gibt an, wie

viele von den tatsächlich relevanten Dokumenten auch gefunden wurden. Es ist schwierig beide Werte zu optimieren, da der Precision-Wert versucht die Anzahl der gefundenen Dokumente einzugrenzen und der Recall-Wert versucht die Anzahl der gefundenen Dokumente zu erweitern. Das F-Maß fasst beide Werte zu einem neuen Wert zusammen.

3.4. Befragungen

TODO

4. Such- und NLP-Algorithmen

4.1. Suchalgorithmen

4.1.1. Keyword Search

TODO

4.1.2. Phrase Search

Eine Phrase Search ist die Suche nach Textausschnitten in Dokumenten. Es werden also nicht mehrere Keywords einzeln betrachtet, sondern die Eingabe in das Suchfeld als Ganzen. Anschließend wird in Dokumenten nach genau diesem Textausschnitt gesucht.

4.1.3. Boolean Search

TODO

4.1.4. Field Search

Eine Field Search sucht Dokumente anhand von Attributen. Der Nutzer kann diese Attribute auswählen. Wenn der Nutzer beispielsweise ein Dokument sucht, welches am 01.01.2005 erstellt wurde, dann kann die Eingabe der Suche so aussehen: "erstelldatum: 01.01.2005". Es können beliebig viele Attribute verwendet werden, um die Suche einzugrenzen.

Neben der Verwendung der Attribute für die Suche selbst, können die Attribute komplementär zu einer anderen Art von Suche verwendet werden. So kann eine Suchfunktion Buttons bereitstellen, über welche Filter festgelegt werden. Nun kann eine Keyword Search durchgeführt werden, aber die gefundenen Dokumente werden mithilfe der Filter weiter eingeschränkt.

4.1.5. Structured Search

Eine klassische Suchfunktion verwendet Keywords, um relevante Dokumente für eine Sucheingabe zu ermitteln. Der Vorteil dieser Art von Suche ist, dass die technische Struktur, in der die Daten vorliegen und gespeichert sind, nicht bekannt sein müssen. Der Nachteil ist auf der anderen Seite, dass die Suchergebnisse unpräzise sein können. Wenn beispielsweise der Suchbegriff Kamera eingegeben wird, dann werden Kameras als Ergebnisse zurückgegeben. Soll die Kamera nun bestimmte Eigenschaften besitzen, dann müssen diese Eigenschaften ebenfalls als Keywords

angegeben werden. Nun wird aber nicht die Suche auf Ergebnisse eingegrenzt, bei denen eine Kamera diese bestimmten Eigenschaften besitzt. Stattdessen werden Suchergebnisse angezeigt, bei denen einige dieser Keywords vorkommen. Demgegenüber stehen Datenbankabfragen, beispielsweise mithilfe von SQL. Bei einer Datenbankabfrage können Objekte abgefragt werden, dessen Eigenschaften ganz bestimmte Werte haben. Die Ergebnisse, die eine solche Abfrage zurückgibt, sind dabei vollkommen genau. Es werden keine Objekte zurückgegeben, welche diese Kriterien nicht erfüllen. Voraussetzung für eine solche Suchabfrage ist allerdings, dass die Struktur der Datenbank a priori bekannt ist. Dem Nutzer muss der Name der Datenbank, der relevanten Tabellen, sowie der relevanten Properties bekannt sein, damit er das passende SQL für die Datenbankabfrage schreiben kann. Strukturierte Suchen sollen die Vorteile beider Vorgehensweisen kombinieren. Die Struktur der Daten soll a priori nicht bekannt sein müssen, aber trotzdem sollen die Suchergebnisse vollkommen präzise sein.

4.1.6. Lexical Search

TODO

4.1.7. Semantic Search

Eine Semantische Suche arbeitet nicht anhand von Keywords, sondern anhand von Bedeutungen von Wörtern. Sie versteht, dass einige Wörter sehr ähnlich sind, so wie "rollout" und "deployment", und dass diese Wörter oft im gleichen Kontext verwendet werden. Um zu verstehen, welche Wörter kontextuell zusammengehören, werden die Wörter in einem n-dimensionalen Raum positioniert. Wörter, die sich sehr ähnlich sind, also im gleichen Kontext verwendet werden, haben in diesem n-dimensionalen Raum eine geringe Distanz. Wörter, die sich eher unähnlich sind, wie "rollout" und "API", haben eine größere Distanz.

Um eine semantische Suche zu implementieren, werden die Technologien von Transformern und Vektordatenbanken verwendet. Ein Transformer bekommt als Input eine große Menge an Text und mappt die einzelnen Wörter auf einen Vektor einer beliebigen Länge. Der Vektor, der am Ende herauskommt, beschreibt die Position des Wortes in dem n-dimensionalen Raum. Der Vektor beschreibt gewissermaßen, wie stark ein Wort in eine abstrakte Kategorie einzuordnen ist. Jeder Wert im Vektor entspricht einer Kategorie. Mithilfe der Vektoren können verschiedene Wörter hinsichtlich ihrer Ähnlichkeit analysiert werden. Ähnliche Wörter habe eine große räumliche Nähe, während zwei Wörter, die in vollkommen unterschiedlichen Kontexten verwende werden eine sehr große Distanz im Raum besitzen. Nehmen wir für ein Beispiel einen dreidimensionalen Raum an. Die X-Achse ist beschriftet mit dem Wort „Tier“, die Y-Achse ist beschriftet mit dem Wort „Computer“ und die Z-Achse ist beschriftet mit dem Wort „Mensch“. Nun geben wir einem Transformer das Wort „Katze“, und der Transformer berechnet einen dreidimensionalen Vektor, welcher das Wort „Katze“ im Raum positioniert. Weil eine

Katze ein Tier ist, ist der X-Wert des Vektors eins. Der Wert eins bedeutet, dass das Wort vollständig zu dieser Kategorie gehört. Da eine Katze überhaupt nichts mit einem Computer zu tun hat, ist der Y-Wert des Vektors 0. Nun ist eine Katze zwar kein Mensch, aber eine Katze ist ein Haustier von Menschen. Es ist denkbar, dass die Wörter Katze und Mensch oft im gleichen Kontext verwendet werden, sodass der Wert bei 0,3 liegen könnte. Damit der Transformer einen Vektor berechnen kann, braucht er eine Menge Daten. Diese Daten erhält er aus vielen Texten. Werden zwei Wörter oft im gleichen Text genannt oder kommen zwei Wörter in vielen Texten sehr nahe beieinander vor, dann geht der Transformer davon aus, dass die beiden Wörter ähnlich sind, und berechnet ähnliche Vektoren. Zuvor müssen die Texte allerdings bereitgestellt werden. Dazu kann beispielsweise das Internet gecrawlt werden. Die Ergebnisse des Transformers werden in einer Vektordatenbank gespeichert. Eine Vektordatenbank ist eine Datenbank, welche Vector Embeddings, also ein Objekt als Key und dessen Vektor als Value speichert. Bei dem Objekt kann es sich um Wörter handeln, dann wird auch von Word Embeddings gesprochen. Es können aber auch Daten andere Daten, wie Bilder, Videos oder Audio gespeichert werden. Der Zweck von Vektordatenbanken ist es, Daten nicht einfach linear zu speichern, sondern in einem Raum. Die Distanz zwischen zwei Einträgen in diesem Raum beschreibt dessen Ähnlichkeit. Genau diese Informationen machen sich semantische Suchen zu Nutze.

4.2. NLP-Algorithmen

TODO

5. Vergleich der Suchfunktionen

6. Erläuterung des Ergebnisses

7. Implementierung

Die Architektur von Suchmaschinen besteht aus drei Komponenten. Aus dem Crawling, der Indexierung und der Suchfunktion selbst. Das Crawling ist zuständig für das Finden von Websites. Die Indizierung ist zuständig für das optimale Speichern der Informationen der Websites, und die Suche ist zuständig für das Verstehen der Nutzeranfrage und die Abfrage der relevantesten Informationen aus dem Index, sowie dessen Verarbeitung und Darstellung. Im Folgenden wird der Begriff Dokument verwendet, um die Dateien zu beschreiben, welche durch einen Crawler gesucht und durch den Index verarbeitet werden. Unter Dokumenten können hierbei auch eine Website verstanden, welche durch einen Webcrawler durchsucht werden.

7.1. Crawling

Bei der Implementierung eines Systems für eine Suchfunktion benötigt das System zunächst einen Datensatz von Dokumenten, welche überhaupt grundsätzlich über die Suchfunktion gefunden werden können. Dieser wird mithilfe eines Crawlers aufgebaut. Ein Crawler ist ein Algorithmus, welcher ein Dokument als Startpunkt bekommt, und anhand dessen neue Dokumente findet. Der Algorithmus analysiert dazu das Dokument auf Links, welchen der Algorithmus anschließend folgt. Die neuen Dokumente werden durch den Index verarbeitet und wiederum auf neue Links analysiert. Dieses Verfahren kann beliebig lange und beliebig rekursiv durchlaufen werden, um den Index zu erweitern. Neben dem Crawling können Indizes befüllt werden, indem eine Liste von Dokumenten übergeben werden, welche dem Index hinzugefügt werden sollen.

7.2. Indizierung

Bei der Indexierung werden Wörter und Tokens mit Dokumenten assoziiert. Dazu werden Wörter aus gecrawlten Dokumenten extrahiert und in die Datenbank geschrieben. Den Dokumenten werden IDs zugeordnet und diese IDs werden den Wörtern zugeordnet. Dem Index werden nun weitere Informationen hinzugefügt, wie die Wortfrequenz. Die Wortfrequenz gibt an, wie oft ein Wort in einem Dokument vorkommt. Es wird auch gespeichert, an welchen Stelle des Dokuments das Wort vorkommt, und auch in wie vielen Dokumenten ein Wort vorkommt. Diese Art der Indizierung wird auch als invertierter Index bezeichnet, weil den Wörtern bzw. Tokens die Dokumente zugeordnet werden, und nicht umgekehrt. Wenn der Nutzer nun ein Keyword in die Suche eingibt, dann sind diese Keywords oft bereits im Index vorhanden, sodass die Dokumente, welche diese Keywords bein-

halten einfach dem Index entnommen werden können. Aufgabe der Suche ist es anschließend die Ergebnisse aufzubereiten.

Volltext-Indizierung

Bei der Indizierung der Wörter besteht die Problematik, dass gleiche Wörter in unterschiedlichen Formen existieren können. So stammen „Heizung“ und „heizen“ beide von dem gleichen Wortstamm „heiz“ ab. Um bei der Indizierung Speicherplatz zu sparen, können Wörter auf diesen Wortstamm reduziert werden, damit sie als ein einziges Wort betrachtet werden können. Die Bildung des Wortstamms wird auch als Stemming bezeichnet. Beim Stemming kann es jedoch zu Overstemming und Understemming kommen. Overstemming bedeutet, dass zwei Wörter, die eigentlich nichts miteinander zu tun haben, also nicht semantisch gleich sind, den gleichen Wortstamm besitzen und als ein Wort betrachtet werden. Ein Beispiel hierfür sind die Wörter „Wand“ und „wandere“, wie in „ich wandere“. Beide besitzen den Wortstamm „wand“ und werden entsprechend als ein Wort betrachtet. Understemming bedeutet, dass zwei Wörter, die eigentlich etwas miteinander zu tun haben, also semantisch gleich sind, nicht den gleichen Wortstamm besitzen und dadurch als zwei verschiedene Wörter betrachtet werden. Ein Beispiel hierfür sind die Wörter „absorbieren“ und „Absorption“, welche die Wortstämme „absorb“ und „absorp“ besitzen. Es gibt Techniken zur Vermeidung solcher Probleme, wie der Einsatz vollständiger morphologischer Analysekomponenten. Hierauf soll aber nicht weiter eingegangen werden. Zur Implementierung eines Volltext-Index werden Dokumente und Wörter in einer m x n Matrix angeordnet, wobei m die Anzahl der Dokumente ist und n die Anzahl der Wörter. Die Werte in dieser Matrix werden anhand einer ausgewählten Metrik bestimmt. Eine oft verwendete Metrik ist das Verhältnis der Wortfrequenz mit der invertierten Dokumentfrequenz. Die Wortfrequenz gibt dabei an, wie häufig das in dem Dokument vorkommt. Die Dokumentfrequenz gibt an, in wie vielen Dokumente ein Wort vorkommt. Das Verhältnis gibt damit an, wie charakteristisch das Wort für ein bestimmtes Dokument ist. Wenn das Wort charakteristisch für ein Dokument ist, dann kommt es in diesem Dokument häufig vor, aber in anderen Dokumenten nur selten. Wenn das Wort nicht charakteristisch für das Dokument ist, dann kommt es in anderen Dokumenten genauso häufig oder häufiger vor als in diesem Dokument. Die Metrik wird auch als tf-idf-Wert bezeichnet. TODO Rechnung besser verstehen Wir können uns nun einen m-dimensionalen Raum vorstellen, in dem jedes Dokument eine Dimension darstellt. Jedes Dokument liegt räumlich auf der Achse der eigenen Dimension. Wenn der Nutzer nun Die Matrix lässt sich konzeptionell so verstehen, dass sie aus n Merkmalsvektoren besteht. Verwenden wir die eben beschriebene Metrik, dann gibt jeder Merkmalsvektor eine Gewichtung an, wie charakteristisch ein Wort für die verschiedenen m Dokumente ist. Nun kann der Nutzer Keywords in die Suche eingeben. Diese Keywords

7.3. Suffix-Trees

TODO

7.4. N-Gramm

Ein N-Gramm ist das Ergebnis der Spaltung eines Textes in N Fragmente. Fragmente können dabei Buchstaben, Wörter, Phoneme, Morpheme etc. sein. N-Gramme können bei der Indizierung verwendet werden und werden auch in Algorithmen von Stemming und Information Retrieval verwendet, außerdem werden durch N-Gramme Rechtschreibprüfungen oder die Suche nach ähnlichen Wörtern ermöglicht. Besteht ein Text aus einem Fragment, dann wird von einem Monogramm geredet. Besteht es aus zwei, wird es Bigramm genannt.

Um Algorithmen wie eine Rechtschreibprüfung zu ermöglichen, wird neben N-Grammen auch Ähnlichkeitsmaß benötigt, welches beschreibt, wie ähnlich sich zwei Terme sind. Das Wort Term ist an dieser Stelle synonym mit Wort zu verstehen. Der Dice-Algorithmus ist ein Ähnlichkeitsmaß für Terme. Er zerlegt zwei Terme in dessen N-Gramme und bestimmt anhand dieser die Ähnlichkeit der beiden Terme. Der Dice-Algorithmus lautet wie folgt:

TODO Equation

Nehmen wir beispielsweise die beiden Terme a=“Wort“ und b=“Wirt“. Verwenden wir in diesem Beispiel Bigramme, dann ergeben sich für a folgende Bigramme: „$W“, „Wo“, „or“, „rt“, „t$“. Für b ergeben sich: „$W“, „Wi“, „ir“, „rt“, „t$“. Die Schnittmenge der beiden Mengen, also die Bigramme, welche sowohl in a als auch in b vorhanden sind, lauten „$W“, „rt“, „t$“. Es sind also drei Stück. Das bedeutet der Zähler der Funktion ergibt 2*3 also 6. Die Anzahl der Bigramme in a und b sind jeweils 5, also ist der Nenner 5+5, also 10. Dann ist das Ähnlichkeitsmaß der beiden Wörter bei der Verwendung von Bigrammen 6/10 (60%). Bei der Verwendung von Trigrammen besitzt a „$$W“, „$Wo“, „Wor“, „ort“, „rt$“, „t$$“ und b „$$W“, „$Wi“, „Wir“, „irt“, „rt$, „t$$“. Es gibt drei gemeinsame Trigramme. Das Ergebnis ist (2*3)/(6+6) also 50%. An diesem Beispiel lässt sich zeigen, dass sich das Ergebnis des Ähnlichkeitsmaß verändert, wenn ein anderes N-Gramm verwendet wird.

7.5. Information Extraction

Information Extraction ist der Prozess des Extrahierens von Informationen aus Dokumenten. Dazu müssen zunächst die Dokumente gefunden werden, welche die passenden Informationen enthalten. Dieser Schritt wird Information Retrieval genannt. Das Extrahieren von Informationen mit einer Vielzahl von Algorithmen und erfolgt in mehreren Schritten. Der erste Schritt von Information Extraction

ist das Einteilen des Textes in Sätze. Dieser Schritt wird Sentence Segmentation genannt. Anschließend werden aus jedem Satz die einzelnen Wörter extrahiert (Word Tokenization) und kategorisiert als Nomen, Verb, Objekt, Pronomen etc. (Part-of-Speech Tagging). Für das Part-of-Speech Tagging gibt es zwei Implementierungsansätze, rule-based anhand von Grammatikregeln und statistic-based. Es folgt Syntactic Parsing , also das Generieren eines Syntaxbaumes, welcher den Text repräsentiert. Zuletzt wird mithilfe von Coreference Resolution identifiziert, welche Stellen im Text sich auf die gleichen Entitäten beziehen. Ein Beispiel für Coreference Resolution ist, dass ein System versteht, dass mit „Angela Merkel“ und „Merkel“ oder auch „A. Merkel“ die gleiche Person gemeint ist. Mithilfe dieser Schritte können Key Phrase Extraction, Named Entity Recognition, Entity Disambiguation, Relation Extraction und Event Extraction implementiert werden. Key Phrase Extraction ist das Identifizieren von Schlüsselwörtern im Text, welche für diesen Text charakteristisch sind und welche den Text klassifizieren. Ein Rezept für das Backen von Schokoladenkeksen könnte beispielsweise mithilfe von Schlüsselwörtern, wie „Backen“, „Kekse“, „Schokolade“ beschrieben werden. Named Entity Recognition ist das Identifizieren von Personen, Orten und Datumsangaben. Entity Disambiguation ist das Auseinanderhalten von Entitäten, welche die gleichen oder ähnliche Wörter besitzen. Unter dem englischen Wort „apple“ lässt sich nicht nur ein tatsächlicher Apfel verstehen, sondern es könnte sich auch um das Unternehmen apple handeln. Welche Bedeutung des Wortes die korrekte ist, erschließt sich aus dem Kontext. Das ist die Aufgabe von Entity Disambiguation.

TODO Image

Teilbereiche von Informationsextraktion sind Named Entity Recognition, Named Entity Linking, Relation Extraction, Knowledge Base Reasoning. Informationsextraktion ist für viele Anwendungsfälle der erste Schritt in einer größeren Pipeline. Das könnte z.B. ein Chatbot sein oder ein Übersetzer. Named Entity Recognition ist die Erkennung von Eigennamen. Named Entity Linking ist der Prozess, eine bestimmte Entität eine einzigartige Identität zu geben. Die Entität wird auf eine Target Knowledge Base gemappt, welche weitere Informationen über ebendiese Entität enthält. Relation Extraction erkennt Beziehung zwischen Entities. Die Beziehung kann durch ein Subjekt Prädikat Objekt Triple repräsentiert werden. Abgrenzung von Information Extraction und Information Retrieval. Information Retrieval selektiert relevante Dokumente. Information Extraktion extrahiert Informationen aus einem gegebenen Dokument. Die Klassifikation von Texten wird auch als Topic Modeling bezeichnet. Es wird die Häufigkeit von Wörtern aus bestimmten Kategorien bestimmt und damit ein Thema zugeordnet. Dieses Verfahren könnte zur Ermittlung von Labels verwendet werden. Die Labels gehen dabei über die Inhalte des Textes hinaus. Ein Algorithmus zum Topic Modeling ist Explicit Semantic Analysis (ESA). Language Modelling ist ein Task von NLP, welcher es ermöglicht Texte zu übersetzen, Spellchecking, Spracherkennungen und Handschrifterkennung zu implementieren.

TODO Image

8. Zusammenfassung und Ausblick

8.1. Zusammenfassung

TODO

8.2. Ausblick

TODO

Anhang

A. Anhang

A.1. TODO

TODO

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Ich habe alle Stellen, die ich aus den Quellen wörtlich oder inhaltlich entnommen habe, als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Essen, den 11. Mai 2023