



Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

NAO Spracherkennung

Studienprojekt

Name, Vorname:	Anna Stabel, Caroline Sarah Schäfer, Sofie Wagner,
Semester:	WiSe 2024/25
Fachbereich:	Duales Studium (FB 2)
Studiengang:	Duales Studium Informatik
Modul:	Studienprojekt II
Betreuer Hochschule:	Dagmar Monet Diaz
Anzahl der Wörter:	0

Unterschriften

Ort, Datum

Anna Stabel

Ort, Datum

Caroline Sarah Schäfer

Ort, Datum

Sofie Wagner

Ort, Datum

Ausbilder*in SAP

Ort, Datum

Ausbilder*in HZB

Abstract

Hier kommt das Abstract hin

Inhaltsverzeichnis

Abstract	iii
Akronyme	v
1 Einleitung	1
2 Problemstellung	1
2.1 Leistungsfähigkeit	1
3 NLP Suchalgorithmen	4
3.1 Knowledge Graphs	4
3.1.1 Beschreibung	4
3.1.2 Implementierung	4
3.2 Ontologie	5
3.2.1 Beschreibung	5
4 Fazit	6
4.1 Ergebnis	6
4.2 Ausblick	6
Bibliothek	7
Abbildungsverzeichnis	7

Akronyme

KG Knowlegde Graphs

HWR Hochschule für Wirtschaft und Recht

OpenIE Open Information Extraction

ML Maschinelles Lernen

TSP Traveling Salesman Probelem

1 Einleitung

Einleitung

2 Problemstellung

TODO

2.1 Leistungsfähigkeit

Die momentane Performance des Algorithmus wird mit Hilfe der Python-Bibliothek *time* gemessen. Diese Bibliothek bietet Funktionen, die es Entwicklern ermöglichen, mit Zeiten zu arbeiten und verschiedene Zeitoperationen durchzuführen. Um die Zeit zu messen, die seit den Funktionsaufrufen vergangen ist, wird die Differenz zwischen der Endzeit und der Startzeit berechnet. Dazu wird der Timer mit der Funktion `time.nc()` initialisiert. Diese Funktion gibt die Anzahl der Nanosekunden seit der Initialisierung des Timers als Integer zurück [2]. `time.nc()` wurde gewählt, um potentiellen Präzisionsfehlern, die aufgrund einer Floating Nummer passieren können, zu vermeiden. Die folgende Tabelle zeigt die Messergebnisse:

Frage	Antwort-Algorithmuszeit (ns)	Transkriptionszeit (ns)	Gesamtzeit (ns)
Wie oft muss man einen PTB schreiben?	33659000	2432785000	2474059000
Welche Fachbereiche gibt es in der HWR?	18816000	388357000	411529000
Wie kann ich mich für ein duales Studium bewerben?	14467000	360204000	379107000
Erzähl mir über den Informatik-Studiengang.	13036000	308227000	325151000

Frage	Antwort- Algorithmuszeit (ns)	Transkriptionszeit (ns)	Gesamtzeit (ns)
Wann wurde die HWR Berlin gegründet?	22216000	379835000	405591000
Erzähl mir was über die HWR.	17349000	316136000	337483000
Welche Voraussetzungen gibt es für ein Informatik-Studium?	12492000	405805000	422220000
Was ist eine Studienarbeit?	13245000	291517000	308290000
Was ist ein Studiengang?	12957000	305227000	320896000
Was bedeutet PTB?	14044000	307653000	323718000

Durchschnittszeiten:

- Antwort-Algorithmuszeit: 17228100 ns
- Transkriptionszeit: 549574600 ns
- Gesamtzeit: 570804400 ns

Im folgenden wird die Antwort-Algorithmuszeit analysiert, um Verbesserungspotential im Suchalgorithmus festzustellen. Die Funktionsabfolge wurde in ?? erläutert.

Funktion `db_connector.get_generic_term`:

- Beschreibung: Für jedes relevante Wort wird eine Datenbankabfrage durchgeführt, um dessen generische Form zu finden. Dies ist der teuerste Schritt, da jede Abfrage Zeit beansprucht und die Abfragen in einer Schleife ausgeführt werden.
- Zeitaufwand: Dies hängt von der Anzahl der Wörter und der Geschwindigkeit der Datenbank ab, typischerweise im Bereich von Millisekunden.

- Laufzeitkomplexität: $O(n \cdot m)$, wobei n die Anzahl der Wörter und m die durchschnittliche Zeit für eine einzelne Datenbankabfrage ist. Falls für jedes Wort eine Abfrage ausgeführt wird, summieren sich die Datenbankoperationen linear zur Anzahl der Wörter.

Funktion `caseID = counter.count_ids`:

- Beschreibung: Dieser Schritt durchsucht die IDs basierend auf Gewichtungen der Wörter und sucht nach dem relevantesten caseID. Die Laufzeit hängt von der Implementierung von `count_ids` und der Anzahl der Wörter ab.
- Zeitaufwand: Variiert von Millisekunden bis Sekunden, abhängig von der Datenbank und der Anzahl der IDs in der Datenbank.
- Laufzeitkomplexität: $O(n \cdot k)$, wobei n die Anzahl der relevanten Wörter und k die Anzahl der verfügbaren caseIDs in der Datenbank ist.

Funktion `db_connector.get_answer_from_db`:

- Beschreibung: Die Funktion ruft die Antwort basierend auf einem einzelnen caseID ab. Da nur eine Datenbankabfrage erforderlich ist, ist die Laufzeit unabhängig von der Eingabelänge.
- Zeitaufwand: Typischerweise Millisekunden, aufgrund nur einer Abfrage.
- Laufzeitkomplexität: $O(1)$, da nur eine einzige Datenbankoperation ausgeführt wird, um die Antwort abzurufen.

Die zeitintensivsten Vorgänge sind demnach die Abfrage der generischen Form des Wortes und die Gewichtung der relevantesten caseIDs. Das Ziel dieser Studienarbeit besteht folglich darin, die Laufzeit der zuvor beschriebenen Funktionen zu mindern bzw. den Suchalgorithmus so zu modifizieren, dass diese obsolet werden.

3 NLP Suchalgorithmen

TODO: Intro für NLP Suchalgorithmen

3.1 Knowledge Graphs

3.1.1 Beschreibung

Knowledge Graphs (KG) stellen eine strukturierte Darstellungsform von Informationen dar, welche aus unstrukturierten Texten gewonnen werden. Sie setzen sich aus Informationsentitäten, welche Knoten genannt werden, und Beziehungen zwischen den Informationsentitäten, welche Kanten genannt werden, zusammen. Diese werden aus Textdaten abgeleitet. Dadurch wird die Integration, der Abruf und die Analyse von Informationen erleichtert [4]. Um einen KG aus einem Text zu konstruieren, werden verschiedene Methoden. Beispiele dafür sind Techniken wie Open Information Extraction (OpenIE), Maschinelles Lernen (ML) und semantische Analyse zum Einsatz [5]. Die strukturierte und semantische Darstellungsform von Informationen, wie sie in Knowledge Graphen erfolgt, ermöglicht eine präzisere und effizientere Beschaffung von textbasierten Informationen [3]. Dies wird durch folgende Faktoren begünstigt:

- Die verbesserte Textdarstellung ermöglicht die Rückgabe reichhaltiger semantischer Strukturen.
- Die automatische Strukturierung von Textinhalten wird durch KGs signifikant vereinfacht, da eine Kategorisierung von Textinformationen in kürzerer Zeit erfolgt [4].
- Die Berechnung der semantischen Ähnlichkeit, welche eine Steigerung der Effizienz und Genauigkeit von Suchergebnissen zum Ziel hat, kann mittels KGs ohne großen Aufwand durchgeführt werden [6].
- Die Integration von KGs in multimediale Modelle, wie beispielsweise Wikipedia, ermöglicht die Nutzung zusätzlicher Ressourcen, beispielsweise visueller Art, für die semantische Suche sowie die Beantwortung von Fragen [7].

3.1.2 Implementierung

Die Implementierung von KGs erfolgte unter Zuhilfenahme der Python-Bibliothek NetworkX in Kombination mit spaCy, einer bereits zuvor im Code verwendeten Python-

Bibliothek. Auf spaCy wurde bereits zuvor eingegangen. NetworkX ist ein Python-Paket, welches die Erstellung, Bearbeitung und Untersuchung der Struktur, Dynamik und Funktionen komplexer Netzwerke ermöglicht. Die Python-Bibliothek wurde als Werkzeug zur Umsetzung der vorliegenden Anforderungen gewählt. Das Paket ermöglicht es Entwicklern verschiedene Arten von Graphen (bspw. Diagraphen und Multigraphen) aus diversen Datenstrukturen wie Text oder XML zu erstellen oder zu generieren. Zudem können Operationen wie das Löschen von Knoten an Graphen durchgeführt, Graphen analysiert (beispielsweise die Anzahl der Knoten gezählt) oder Algorithmen wie z.B. zum Lösen des Traveling Salesman Problem (TSP) implementiert werden [1].

3.2 Ontologie

3.2.1 Beschreibung

1. **Fett 1** Nummerierte liste 1
2. *Kursiv 1* Nummerierte liste 1
3. Nummerierte liste 1

Eine Matheformel (Satz des Pythagoras):

$$a^2 + b^2 = c^2$$

wobei a und b die Längen der Katheten eines rechtwinkligen Dreiecks sind, und c die Länge der Hypotenuse.

1 Python Code Section 1

Listing 1: Pip Update

""Deutsche Anführungszeichen"" ?? referenz Zu Label1 „Anführungszeichen unten, Anführungszeichen oben“

?? Bild Referenz

Column1	Column2
row1	row2

Tabelle 2: Your table caption here

4 Fazit

Fazit

4.1 Ergebnis

Ergebnis

4.2 Ausblick

Ausblick

Bibliothek

- [1] NetworkX Community. *NetworkX Dokumentation*. 2024. URL: <https://networkx.org/documentation/stable/index.html>. (Besucht am 26.10.2024).
- [2] Python Community. *time — Time access and conversions*. 2024. URL: <https://docs.python.org/3/library/time.html#time.time>. (Besucht am 02.11.2024).
- [3] Laura Dietz, Alexander Kotov und E. Meij. „Utilizing Knowledge Graphs in Text-centric Information Retrieval“. In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (2017). DOI: 10.1145/3018661.3022756.
- [4] Wenny Hojas-Mazo u. a. „A Concept-Based Text Analysis Approach Using Knowledge Graph“. In: (2018), S. 696–708. DOI: 10.1007/978-3-319-91476-3_57.
- [5] José-Lázaro Martínez-Rodríguez, I. Lopez-Arevalo und Ana B. Ríos-Alvarado. „OpenIE-based approach for Knowledge Graph construction from text“. In: *Expert Syst. Appl.* 113 (2018), S. 339–355. DOI: 10.1016/J.ESWA.2018.07.017.
- [6] Ce Wang, Hongzhi Yu und F. Wan. „Information Retrieval Technology Based on Knowledge Graph“. In: (2018), S. 291–296. DOI: 10.2991/ICAMMCE-18.2018.65.
- [7] Meng Wang u. a. „Richpedia: A Large-Scale, Comprehensive Multi-Modal Knowledge Graph“. In: *Big Data Res.* 22 (2020), S. 100159. DOI: 10.1016/j.bdr.2020.100159.

Abbildungsverzeichnis

AI-Verzeichnis

- ChatGPT 4o und o1-preview
- Consensus AI
- Perplexity
- DeepL

Ehrenwörtliche Erklärung

Wir erklären hiermit ehrenwörtlich:

1. dass wir unsere Studienarbeit selbstständig verfasst habe,
2. dass wir die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe,
3. dass wir unsere Studienarbeit bei keiner anderen Prüfung vorgelegt habe.

Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Ort, Datum

Anna Stabel

Ort, Datum

Caroline Sarah Schäfer

Ort, Datum

Sofie Wagner