Automatisierte Zusammenfassung von Änderungen in der Online-Hilfe des Microsoft SQL Servers

Masterarbeit zu Erlangung des

Master of Advanced Studies ZFH in

Data Science

vorgelegt von

Georg Lampart

geboren am 01.02.1972

von Eschenbach, Luzern

eingereicht

Dr. Mark Cieliebak

Zürich, 29. Juni 2020

Management Summary

Ausgangslage und Ziele

Microsoft SQL Server ist eines der drei grossen, kommerziellen, relationalen Datenbankmanagement-Systeme. In der Ära von Cloud Services werden neue Funktionen monatlich oder sogar täglich ausgeliefert, auch beim Microsoft SQL Server. Die Online-Hilfe besteht aus ca. 23'000 Seiten und ist auf Github öffentlich zugänglich. Es gibt täglich viele Änderungen, davon entfällt ein Grossteil auf Formatierungs- oder stilistische Anpassungen. Für EntwicklerInnen ist es unmöglich, mit angemessenem Aufwand die relevanten Änderungen für ihre tägliche Arbeit zu ermitteln.

Diese Masterarbeit hat zum Ziel, relevante Modifikationen in der Dokumentation automatisiert zu extrahieren und in einer kompakten Form anzuzeigen. Die Zusammenfassung soll über einen Teilbereich der Online-Hilfe von ca. 360 Seiten erfolgen. Fünf verschiedene Zeiträume dienen als Entscheidungsgrundlage, ob z.B. ein wöchentliches oder monatliches Intervall sinnvolle Daten ergibt. Es werden viele verschiedene Methoden für die Relevanz der Änderungsinformationen angewandt und die Zusammenfassungen quantitativ/qualitativ verglichen. Nicht inbegriffen ist ein Automatismus, um die Daten aus dem Repository von Github herunterzuladen und für die Zeitintervalle aufzubereiten. Dies erfolgt manuell.

Datengrundlage

Die Hilfeseiten werden in Markdown-Dateien verwaltet. Es handelt sich um eine leichtgewichtige Form, Text mit möglichst wenig Formatierungsanweisungen zu speichern und in ein Zielformat, z.B. HTML, transformieren zu können. Jede Datei weist eine Dokumentstruktur auf mit einer Überschriftshierarchie und zugehörigen Abschnitten.

Für den Vergleich einer Datei (1 im Beispiel rechts) zu zwei unterschiedlichen Zeitpunkten wird die Python Library *difflib* eingesetzt. Diese vergleicht den Inhalt auf Zeilenbasis und gibt die Differenzen als Blöcke von Zeilen aus. Diese Blöcke werden nachfolgend *Änderungsblöcke* genannt (2). Innerhalb eines Änderungsblocks können einzelne Zeichen, Wörter oder Sätze verändert werden. Im Beispiel ist das Wort "new" eingefügt worden (3). Diese kleinen Änderungen werden *Differenzoperationen* genannt.

Toolbox

Für die Text- bzw. Differenzanalyse kommen verschiedene Konzepte von Natural Language Processing (NLP) zum Einsatz. Regular Expression ist das wichtigste Werkzeug, mit dem Textstellen anhand von Mustern (Pattern) gefunden oder auch ersetzt werden können.

Hilfreich ist auch das POS-Tagging (Part-of-speech), das Wörter als Nomen, Verben etc. kennzeichnet. Häufig zum Einsatz kommen Lemmas, die den Stamm eines Wortes wiedergeben. Weitere Instrumente sind (Minimum) Edit Distance für Zeichen oder Wörter, Word Error Rate, Word Similarity, Thesaurus für Synonyme, Spell Checking und Dependency Parsing. Alle Werkzeuge werden in Kapitel 2, Seite 3, beschrieben.

Vorgehen

In der ersten Phase werden die Änderungsblöcke bereinigt. Basis ist die Testdatei, die die meisten Änderungsblöcke aufweist. Die Markdown-Formatierungen werden entfernt, um den Rohtext auszuwerten. Die Änderungsblöcke werden manuell kategorisiert; die Testdatei enthält über 15

Kategorien. Um die kategorisierten Differenzen zu entfernen, werden Cleansing-Funktionen programmiert. Es gibt einfache Funktionen (z.B. um Leerzeichen zu entfernen) und komplexe (z.B. um ähnliche Wörter mit Word Similarity zu erkennen). Resultat der ersten Phase: Die Testdatei enthält für das Halbjahr-Intervall über 400 Änderungsblöcke, davon sind 71 als relevant (oder nicht bereinigbar) eingestuft (= Baseline). Die Bereinigung kommt mit 86 restlichen Änderungsblöcken nahe an diese Baseline.

In der zweiten Phase werden die Differenzoperationen aus den Änderungsblöcken extrahiert. Weil einzelne Zeichen oder Wörter den Anwendern keine genügenden Informationen liefern, werden ganze Sätze aufbereitet. Diese Sätze werden auf relevante Änderungen geprüft. Die Analyse ergibt, dass Sätze anzuzeigen sind, die komplett neu hinzugefügt worden sind sowie Änderungen, die die Syntax eines Befehls betreffen. Kleine Wortanpassungen innerhalb eines Satzes beinhalten selten relevante Informationen. Alle auszugebenden Sätze werden pro Hilfeseite in eine Datei zwischengespeichert, damit sie für die Anzeige aufbereitet werden können.

In der letzten Phase werden die Daten für die Anwender aufbereitet. Eine statische Webseite ist ausreichend, da sich die zugrundeliegenden Daten nicht ändern. Drei Bestandteile erzeugen eine attraktive und einfach bedienbare Oberfläche: 1. erlauben Frameworks wie *Bootstrap* ein konsistentes und einfach zu erstellendes Design. 2. sind die zwischengespeicherten Dateien als Python Liste aufzubereiten. 3. kann die Webseite mit einem daten-getriebenen Template generiert werden. Das Template enthält einerseits das Gerüst mit Kopfzeile, Navigation und Hauptbereich basierend auf dem *Bootstrap* Design, und andererseits gibt es Platzhalter, die mit den Daten aus der Python Liste ersetzt werden. Das Resultat ist eine Webseite mit den relevanten Differenzen. Wöchentliche oder monatliche Intervalle sind zu bevorzugen, weil die anderen, grösseren Zeitbereiche zu viele und zu komplexe Änderungen für die Anwender aufweisen.

Diskussion und Ausblick

Natural Language Processing (NLP) liefert interessante Konzepte. POS-Tagging und Lemma fanden erfolgreich Anwendung, andere Werkzeuge wie Thesaurus und Word Similarity brachten erst nach manuellen Erweiterungen die gewünschte Wirkung. Die umgesetzten Algorithmen sind regelbasiert und liefern brauchbare Resultate. Unklar ist, ob sich die Regeln auf andere Gebiete übertragen lassen.

Kritisch zu hinterfragen ist das Verfahren, die Änderungsblöcke in der ersten Phase zu verändern, z.B. Unterschiede entfernen, Umwandeln in Kleinschreibung. Die Aufbereitung der Anzeige mit Originalsätzen wurde dadurch erschwert. Im Anschluss an das Projekt ist eine alternative Vorgehensweise zu prüfen. Überraschend war, dass nur komplette, eingefügte Sätze relevant sind. Kleine Änderungen (Zeichen, Wörter) innerhalb eines Satzes sind fast ausnahmslos unbedeutend.

Um mit der Webseite produktiv gehen zu können, muss die Qualität der Differenzaufbereitung verbessert werden, z.B. liefert die Library *difflib* ungenügende Resultate. Die Darstellung der Differenzen ist zu optimieren: Grossen Hilfeseiten haben viele Differenzen, kleine nur wenige; die Anzeige wird dadurch unausgewogen. Um alle 23'000 Seiten aufzubereiten, ist eine hierarchische Struktur notwendig, um die Informationen nach Modulen und Zeitintervallen aufzuteilen. Wichtig ist, die Anwender in der Übersicht hinzuweisen, wenn neue Hilfeseiten hinzugekommen sind. Als letztes ist ein Automatismus umzusetzen, der die Daten von Github herunterlädt und die Zeitintervalle vorbereitet. Dann können die Differenzen ermittelt und für die Benutzer aufbereitet werden.

Insgesamt ist ein Projekt entstanden, das die Basis für eine erfolgreiche Ausgabe von Differenzen für interessierte Benutzer bildet. Anpassungen und Verbesserungen sind noch notwendig, damit eine grosse Benutzerakzeptanz erreicht werden kann.

Inhaltsverzeichnis

Abbildungsverzeichnis

Logos auf der Titelseite für ZHAW[1] und Microsoft SQL Server[2].

[1] Das Logo *ZHAW* stammt von https://www.zhaw.ch/storage/engineering/ueber-uns/medien
[2] Das Logo *Microsoft SQL Server* stammt von https://worldvectorlogo.com/logo/microsoft-sql-server

Tabellenverzeichnis

Abkürzungsverzeichnis

CRLF ..... Carriage Return & Line Feed
CSS ..... Cascading Style Sheets
DWH ..... Data Warehouse
ETL..... Extract, Transform, Load
HTML..... Hypertext Markup Language
JSON..... JavaScript Object Notation
ML ..... Machine Learning
NLP..... Natural Language Processing
POS..... Part-of-speech
RDBMS..... Relationales Datenbankmanagement-System
SQL..... Structured Query Language
T-SQL..... Transact-SQL
UI..... User Interface
XML ..... Extensible Markup Language

1 Einleitung

1.1 Ausgangslage

In der Ära von Cloud Services nimmt die Zahl der Applikations- und Service-Releases rapide zu. Statt eines Releases alle 1-2 Jahre werden die Produkte in der Cloud monatlich oder sogar täglich neu ausgeliefert. Die zugehörigen Dokumentationen ändern sich entsprechend schnell und häufig.

Dies trifft auch auf das Produkt "Microsoft SQL Server" zu. Es handelt sich um eines der drei grossen, kommerziellen, relationalen Datenbankmanagement-Systeme (RDBMS) auf dem Datenbankmarkt.

Für Entwickler und Datenbankadministratoren des SQL Servers ist es schwierig [geworden], sich einen Überblick zu verschaffen, was sich innerhalb eines Zeitraums, z.B. Woche, Monat, Quartal, geändert hat.

Microsoft verwaltet die Dokumentationen seit einiger Zeit in einem Repository auf Github[3], u.a. auch für SQL Server. Dadurch sind alle Änderungen öffentlich zugänglich. Jeder Interessierte kann anhand der Commits[4] die Anpassungen in den Dateien ermitteln.

Die Dokumentation wird mit Markdown-Dateien verwaltet (Dateiendung "md"). Es handelt sich um eine leichtgewichtige Form, Text mit möglichst wenig Formatierungsanweisungen zu speichern und nachher in ein Zielformat, z.B. HTML, zu transformieren. Bei bisherigen Formaten, wie Word, HTML, XML, war ein Vergleich erschwert, weil die vielen Formatierungsinformationen den eigentlichen Inhalt überblendeten.

Die Hilfe des Microsoft SQL Server umfasst ca. 23'000 Markdown-Dateien. Es gibt täglich sehr viele Änderungen in diesen Daten. Viele dieser Änderungen betreffen aber Rechtschreibkorrekturen und Formatierungsanpassungen, die keinen Informationsgehalt haben. Eine Entwicklerin müsste jede angepasste Datei einzeln vergleichen und auf relevante Änderungen auswerten.

Es bietet sich an, diesen Aufwand durch ein maschinelles Vorgehen zu vereinfachen.

1.2 Abgrenzung

Das RDBMS Microsoft SQL Server ist ein sehr umfangreiches Produkt und besteht aus vielen Komponenten und Funktionalitäten.

Für die Masterarbeit soll das Themengebiet auf die Online-Hilfe für Transact-SQL (T-SQL-)Statements[5] beschränkt werden, die Ende Januar 2020 ca. 360 Seiten umfasst. Technische Änderungen bei solchen Statements werden meistens nicht in den "What's New?"-Seiten bei einem neuen Release erwähnt, da sie für Marketingzwecke zu wenig hergeben. Jedoch sind die Änderungen von hoher Relevanz für Entwickler und Datenbankadministratoren.

---

[3] SourceCode-Verwaltung, es können neben Programmierprojekten aber auch andere Projekte (Repository) mit Dateien verwaltet werden (u.a. Dokumentationen)

[4] Ein "Commit" ist ein nachvollziehbarer Änderungsvorgang, der ein oder mehrere Dateien umfasst.

[5] Ein Beispiel für ein T-SQL-Statement ist ALTER TABLE; es gibt auch noch andere T-SQL-Befehle, z.B. Funktionen, die nicht Teil dieser Arbeit sein werden.

Eine weitere Abgrenzung gibt es bei älteren Versionen von Microsoft SQL Server vor SQL Server 2008. Deren Dokumentation wird nicht auf Github verwaltet. Die Historie ist nicht verfügbar. Deshalb werden die älteren Versionen nicht berücksichtigt.

Für die Masterarbeit werden der Ausgangszustand der Hilfeseiten (Zeitpunkt t_0) sowie der neue Zustand (Zeitpunkt t_1) manuell ermittelt, d.h. es wird kein Automatismus umgesetzt.

1.3 Ziele

Die Masterarbeit hat zum Ziel, relevante Änderungen in der Dokumentation zu erkennen und in einer geeigneten kompakten Form für die Anspruchsgruppen zu präsentieren.

Die Arbeit soll aufzeigen, wo Kompromisse in der Aufbereitung notwendig sind, weil die maschinelle Verarbeitung nicht die Qualität einer mensch-erstellten Auswertung erreichen kann.

Für einen produktiven Einsatz soll aufgezeigt werden, wie die Aufbereitung automatisiert werden kann (wo sind die Source-Daten verfügbar, wie wird das Subset "Transact-SQL-Statements" extrahiert und wie erhält man die Historie für den Ausgangs- und Endzustand, damit ein definierter Zeitbereich verglichen werden kann).[6]

Die Zusammenfassung soll über die ca. 360 Seiten erfolgen. Es sollen (mindestens) drei verschiedene Zeiträume ausgewertet werden, z.B. eine Woche, ein Monat sowie ein halbes Jahr als Entscheidungsgrundlage, ob z.B. ein wöchentlicher oder monatlicher Vergleich sinnvolle Daten ergibt.

Es werden mindestens zwei verschiedene Methoden für die Relevanz der Änderungsinformationen geprüft und die Zusammenfassungen quantitativ/qualitativ verglichen.

[6] Die Informationen für die automatisierte Aufbereitung sind in Anhang D beschrieben.

2 Datengrundlage

Dieses Einstiegskapitel liefert Basisinformationen über das Produkt "SQL Server", den Aufbau einer Hilfeseite, die Herkunft der Daten sowie über die zwei Arten von Differenzen.

2.1 Microsoft SQL Server

Das Produkt "Microsoft SQL Server" ist in den letzten Jahren einerseits im Umfang der Funktionalitäten gewachsen und andererseits kann das Produkt im Cloud-Zeitalter auf vielen Plattformen betrieben werden.

2.1.1 Funktionalitäten

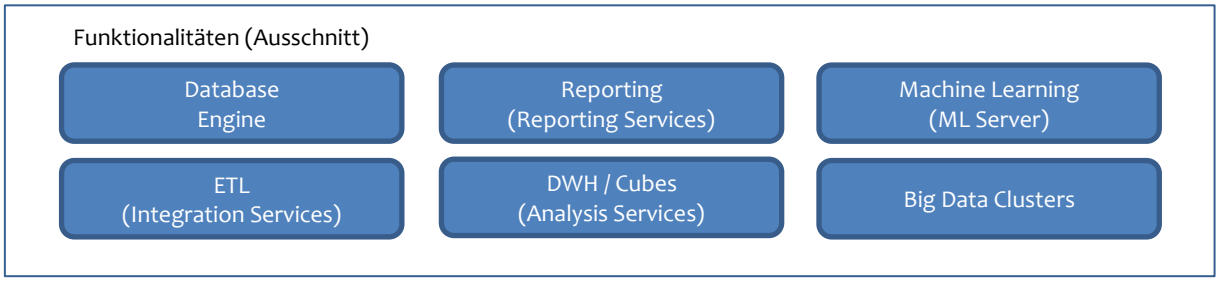


Abbildung 1: Microsoft SQL Server – Funktionalitäten (Ausschnitt)

Microsoft SQL Server ist nicht nur ein relationales Datenbankmanagement-System, sondern umfasst u.a. auch Integration Services fürs Extrahieren, Transformieren und Laden von Daten (Extract, Transform, Load [ETL]); die Reporting Services für Endbenutzer-Reporting; die Analysis Services fürs Data Warehousing und Aufbereiten von Cubes; den Machine Learning Server für die Daten-nahe Ausführung von ML-Modellen mit Python, R oder Java; den Big Data Clusters für die Integration von Big Data aus anderen Daten-Systemen wie Hadoop.

2.1.2 Angebote

War früher Microsoft v.a. für die Windows-Betriebssysteme und Office bekannt, so hat sich dieses Bild in den letzten Jahren geändert. Microsoft hat das Cloud-Geschäft stark ausgebaut, Office 365 wurde in Microsoft 365 umbenannt und läuft auch auf Android- und iOS-Betriebssystemen.

Diese Wandlung ist auch beim Produkt "SQL Server" zu sehen. SQL Server läuft nicht mehr nur auf dem Windows-Betriebssystem, sondern es gibt nun auch eine Linux-Version sowie Unterstützung für Containers. Und für die Cloud sind drei spezifische Angebote verfügbar.

Abbildung 2 zeigt die aktuellen Angebote (englisch: offerings) von SQL Server. Auf der linken Seite sind die traditionellen Angebote, mit denen ein Unternehmen den SQL Server im eigenen Rechenzentrum (On Premises) betreiben kann, bisher v.a. auf *Windows*-Betriebssystemen, seit SQL Server 2016 auch auf *Linux*-Betriebssystemen.

Die rechte Seite zeigt die Cloud-Angebote. Es handelt sich um Managed Services. Dies bedeutet, dass Wartung, Patching, Backups durch den Cloudbetreiber (Microsoft) übernommen werden. Ein Unternehmen, das den Service nutzt, kann sich auf die Nutzung der Funktionalität konzentrieren.

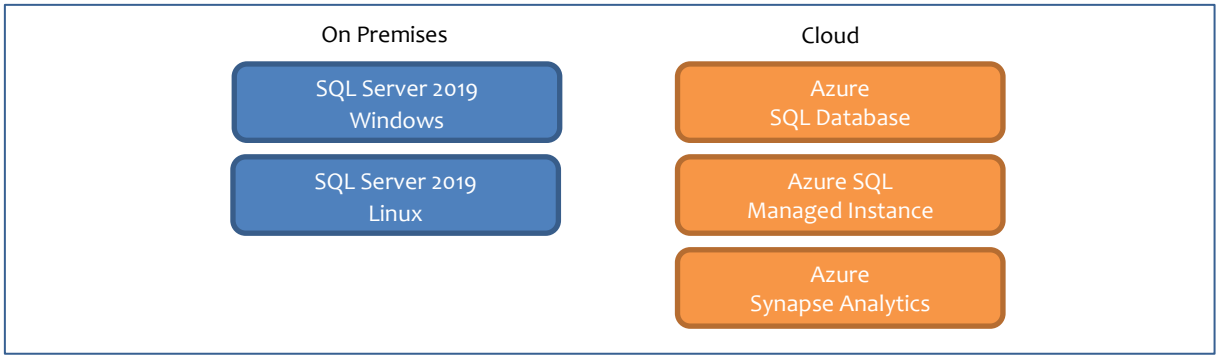


Abbildung 2: Microsoft SQL Server – Angebote

Alle diese Angebote basieren zu grossen Teilen auf der gleichen Technologie, aber im Detail gibt es viele Unterschiede, entweder dass Funktionalitäten nicht bei jedem Angebot verfügbar sind oder dass sich die Syntax unterscheidet. Diese Unterschiede haben auch einen Einfluss auf die Dokumentation.

2.2 Online-Hilfe – Herkunft

Microsoft speichert die Online-Dokumentation für den Microsoft SQL Server auf der Plattform Github im Repository "Microsoft Docs / sql-docs"[7]. Die Dateien und die zugehörigen Änderungen sind öffentlich zugänglich.

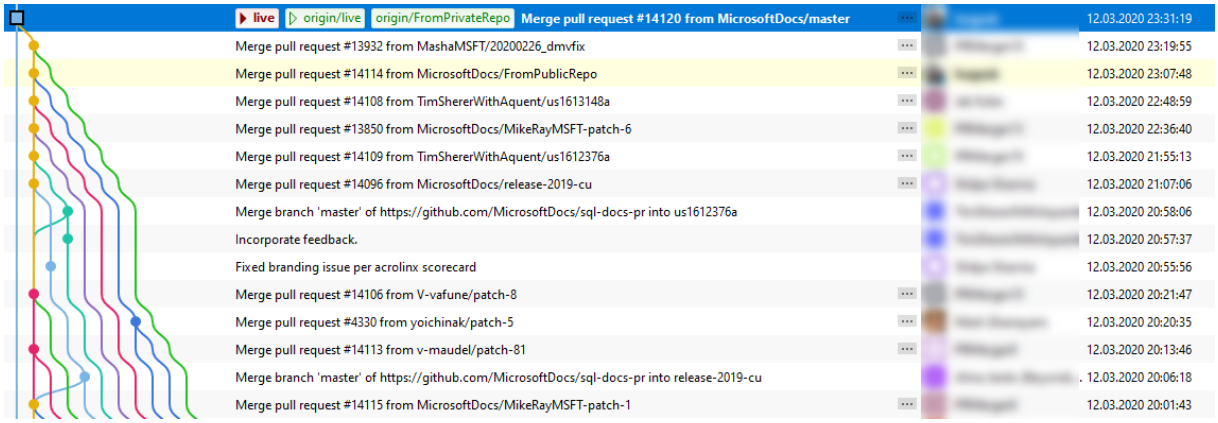


Abbildung 3: Ausschnitt Änderungshistorie im Repository "Microsoft Docs / sql-docs"

Die Hilfeseiten werden in Markdown-Dateien[8] gespeichert mit der Dateiendung "md". Dies ist ein leichtgewichtiges Format, um Text mit minimalen Formatierungsangaben schreiben zu können. Markdown kann automatisiert in andere Formate konvertiert werden, z.B. HTML oder PDF.

Per 12. März 2020 gibt es knapp 23'000 Markdown-Dateien. Täglich werden Dutzende von Änderungen im Repository hinzugefügt. Die Abbildung 3 zeigt im Programm "Git Extensions" einen Ausschnitt der Änderungshistorie.

7 https://github.com/MicrosoftDocs/sql-docs
8 Eine kurze Einführung in die Markdown Syntax findet sich z.B. auf https://commonmark.org/help/

2.3 Online-Hilfe – Dokumentstruktur

Die Hilfeseiten verwenden normale Dokumentstrukturen wie Überschriften und Abschnitte.

2.3.1 Überschriften

Jedes Dokument hat eine Überschrift 1. Dies entspricht dem Titel des Dokuments. Je nach Grösse gibt es Unterabschnitte auf Level 2, 3 oder 4. Die nachfolgende Abbildung 4 zeigt die Überschrift 1 "ALTER DATABASE SET options (Transact-SQL)" und den Abschnitt "Select a product" als Überschrift 2.[9]

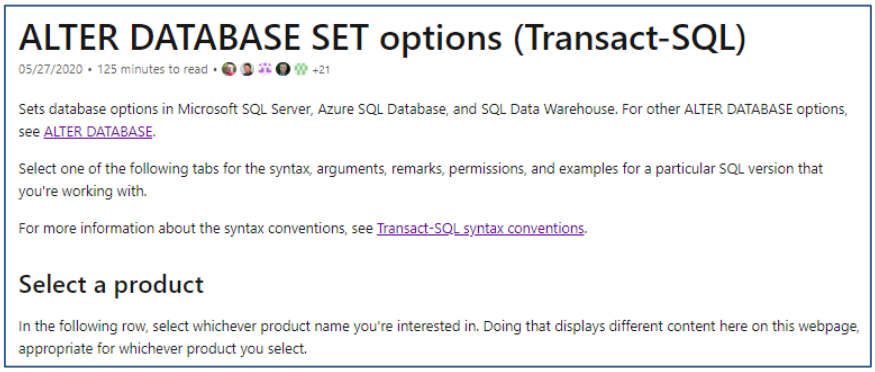


Abbildung 4: Dokumentstruktur - Beispiel für Überschrift 1 und 2

Die Struktur kann für die Relevanz der Daten benutzt werden. Z.B. hat ein neu eingefügter Text im Abschnitt "Syntax" eine hohe Relevanz. Hingegen sind Änderungen im Abschnitt "See also" unter Umständen weniger wichtig.

2.3.2 Monikers

Im Abschnitt 2.1.2 Angebote, Seite 3, ist am Ende beschrieben, dass sich Funktionalitäten für die verschiedenen Angebote unterscheiden können, v.a. auch die Syntax.

Dies wirkt sich auf die Dokumentation aus, weil pro Angebot ein anderer Text konzipiert wird.

Die Benutzerin kann auf der Online-Hilfeseite auswählen, für welches Angebot der Hilfetext angezeigt werden soll. Abbildung 5 zeigt ein Beispiel für die Auswahl. Wenn die Benutzerin "SQL Database single database/elastic pool" anklickt, dann ändert sich die Textausgabe.

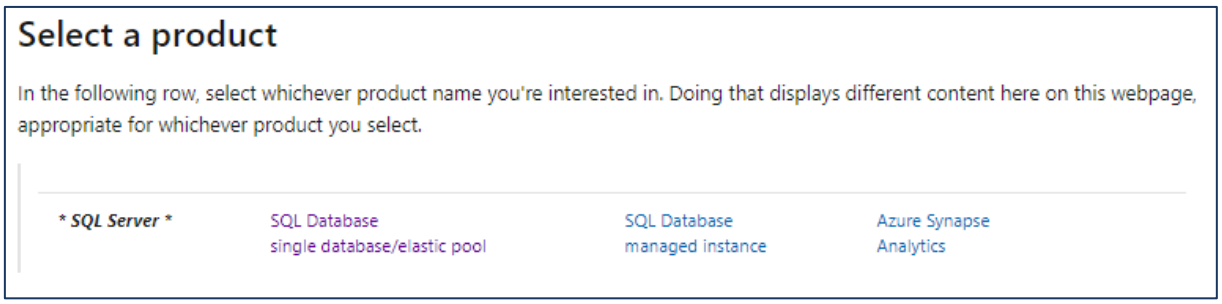


Abbildung 5: Beispiel für die Aufteilung via Moniker

[9] https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-database-transact-sql-set-options

Für die Online-Hilfeseite aus Abbildung 5 gibt es Texte für vier verschiedene Angebote (SQL Server, SQL Database single database/elastic pool, SQL Database managed instance, Azure Synapse Analytics). Gespeichert wird der Text aber *genau in einer* Markdown-Datei. Innerhalb der Datei wird unterschieden, welcher Text für welche Angebote gültig ist. Microsoft verwendet die Bezeichnung "moniker", um die Angebote zu unterscheiden.

Abbildung 6 zeigt die Vorgehensweise. Auf der linken Seite ist die Markdown-Datei aufgeführt. Der blaue Text (für "Alle monikers") gilt für alle Angebote. Der nächste Block ist spezifisch gekennzeichnet für das Angebot "SQL Server" (On Premises). Der Textblock wird technisch mit "moniker-range <SQL Server>" und "moniker-end" gekennzeichnet. Diese Blöcke können in beliebiger Reihenfolge vorhanden sein.

Wenn ein Benutzer nun die Online-Hilfeseite für das Angebot "SQL Server" anschaut, dann werden der blaue Textblock "Alle monikers" und der rote "SQL Server" angezeigt[10]. Die Anzeige ändert sich entsprechend bei Auswahl von "Single database".

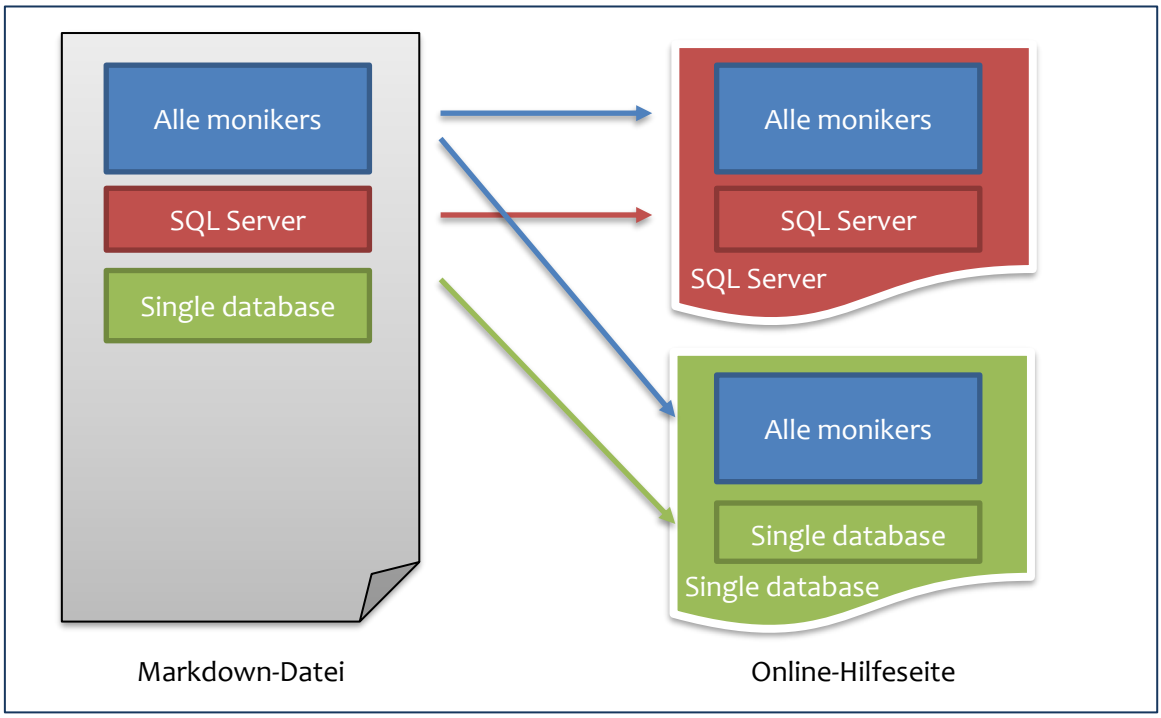


Abbildung 6: Ausgabe Markdown nach Online-Hilfeseite mit Monikers

2.4 Differenzen

Eine Aufgabe in dieser Masterarbeit ist es, innerhalb eines Zeitintervalls die Differenzen in den zugrundeliegenden Dateien zu finden. Für die maschinelle Auswertung kommen spezialisierte Libraries zum Einsatz. Bei der Programmiersprache Python gibt es die Standardlibrary *difflib*. Zu beachten ist, dass jedes Programm und jede Library einen eigenen Algorithmus implementiert, wie die Differenzen ermittelt werden. Deshalb werden die Differenzen bei jedem Programm anders ausgegeben, v.a. wenn es komplizierte Änderungen betrifft (z.B. das Verschieben von Text).

[10] Es ist anhand des HTML-Codes nicht erkennbar, ob Microsoft vier verschiedene Versionen der Online-Hilfeseite abspeichert oder ob die Anzeige anhand einer einzigen Online-Hilfeseite zur Laufzeit generiert wird.

Für die optische Darstellung der Differenzen gibt es Programme, wie Winmerge, Visual Studio, BeyondCompare etc., die zur Nachkontrolle genutzt werden können.

2.4.1 Arten von Differenzen

Als Resultat eines Dateivergleichs werden ganze Zeilen oder sogar Blöcke von unterschiedlichen Zeilen zurückgegeben. Innerhalb einer Zeile oder eines Blocks kann es dann mehrere kleine Änderungen geben.

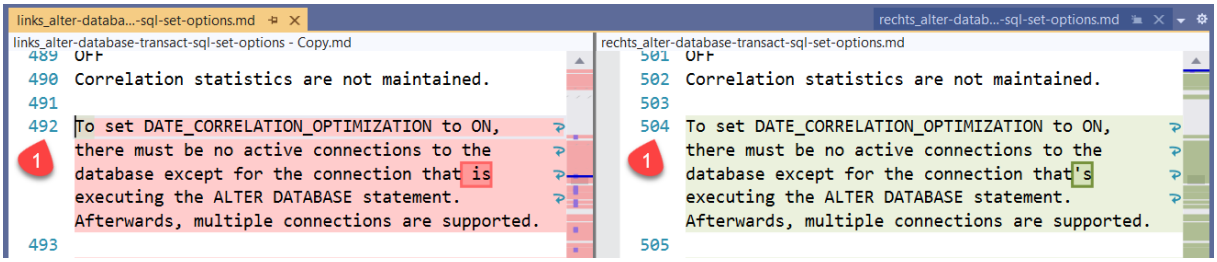


Abbildung 7: Optische Darstellung von Differenzen (hier angezeigt in Visual Studio)

In Abbildung 7 sind die Zeilen 492 (links) und 504 (rechts) als Block sichtbar; innerhalb des Blocks gibt es eine Änderung, nämlich das Ersetzen von "… is" nach "…'s" (technisch ein Replace). Im gleichen Block können aber auch Wörter oder ganze Sätze hinzugefügt oder gelöscht werden.

Für die Ausführungen in den nachfolgenden Kapiteln sind diese unterschiedlichen Differenzarten von Bedeutung. Deshalb werden hier zwei – nicht offizielle – Begriffe eingeführt, nämlich *Änderungsblock* und *Differenzoperation*:

1. Als Basis dient eine Datei mit zwei Zuständen in einem Zeitintervall t_0 und t_1 (hier Sample.md).
2. **Änderungsblock:** Die Python Library *difflib* analysiert die Datei und gibt auf Basis der Zeilen einen oder mehrere Änderungsblöcke aus. Ein solcher Block kann auch mehrere Sätze umfassen.
3. **Differenzoperation:** Ein Änderungsblock wiederum kann mehrere kleine Änderungsvorgänge bzw. -operationen aufweisen, nämlich Einfügen, Ändern und Löschen von Textstellen (im Beispiel rechts das Wort "new"). Diese Operationen werden nachfolgend als **Differenzoperationen** bezeichnet.

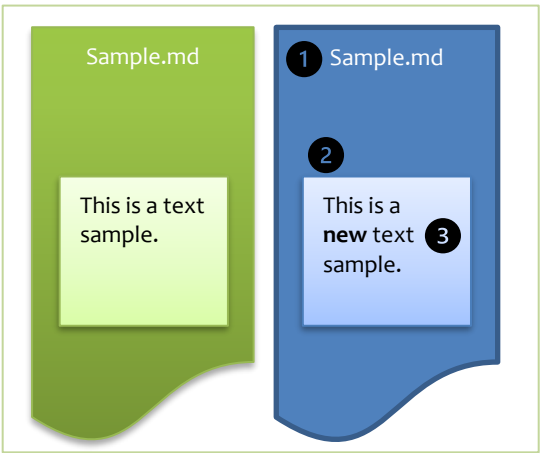


Abbildung 8: Arten von Differenzen

In den kommenden Abschnitten werden in einem ersten Schritt die *Änderungsblöcke* ausgearbeitet. Ob ein Änderungsblock relevant ist, wird anhand der *Differenzoperationen* ermittelt.

2.5 Daten für Masterarbeit

Wie in Abschnitt 2.1 Microsoft SQL Server, Seite 3, beschrieben, ist Microsoft SQL Server ist ein sehr umfangreiches Produkt. Für die Masterarbeit bietet es sich an, mit einem kleinen Ausschnitt der verfügbaren Dateien zu arbeiten.

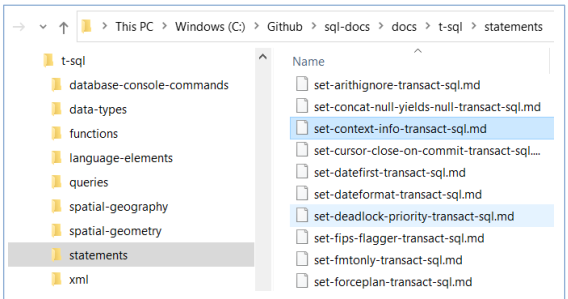


Abbildung 9: Git Repository Ausschnitt t-sql/statements

Das Themengebiet "T-SQL Statements" stammt aus dem Modul "Datenbank-Engine", umfasst 359 Seiten und wird als Basis für die Masterarbeit übernommen. Abbildung 9 (rechts) zeigt einige dieser Dateien im Unterordner t-sql\statements.

Um Aussagen zur Anzahl Differenzen machen zu können, werden mehrere Zeiträume vorbereitet. Die Dateien werden für ein Intervall von einer Woche, einem Monat, einem Quartal, einem Halbjahr sowie einem Jahr aus der Änderungshistorie extrahiert.

Die nachfolgenden Tabelle 1 zeigt einen Ausschnitt der Entwicklung der Änderungen bei jeder betroffenen Datei. Die Änderungsblöcke wurden mit der Python Library *difflib* ermittelt. Die komplette Tabelle ist in Anhang A nachvollziehbar.

Zuerst wird das kleinste Intervall "Woche" aufgelistet, die Daten sind eine Woche alt; danach kommen die weiteren grösseren Intervalle (die Zeit läuft rückwärts von links nach rechts, die Datenbasis wird immer älter). Grüne Farben bedeuten wenig Differenzen, rötliche und rote Differenzen viele Differenzen. Blau weist auf eine aussergewöhnliche hohe Anzahl hin und weisse Zellen zeigen an, dass die Datei erst nach diesem Intervallzeitpunkt erstellt wurde.

Dateiname	2020			2019	
Originalzustand per 13.03.2020	06.03.	13.02.	13.12.	13.09.	13.03.
Intervall rückwärts zum Originalzustand	Woche	Monat	Quartal	Halbjahr	Jahr
add-sensitivity-classification-transact-sql.md	0	0	0	8	9
alter-application-role-transact-sql.md	0	0	0	3	4
alter-authorization-transact-sql.md	0	0	0	3	4
alter-certificate-transact-sql.md	0	0	1	2	17
alter-column-encryption-key-transact-sql.md	0	0	0	9	9
alter-database-scoped-configuration-transact-sql.md	0	8	10	51	52
alter-database-transact-sql-compatibility-level.md	0	1	3	38	36
alter-database-transact-sql-file-and-filegroup-options.md	0	0	1	7	10
alter-database-transact-sql-set-options.md	0	1	406	405	275
alter-database-transact-sql.md	0	0	22	36	82
alter-external-data-source-transact-sql.md	0	0	0	2	19
alter-external-language-transact-sql.md	0	0	0	2	
alter-external-library-transact-sql.md	0	0	0	4	21
alter-fulltext-index-transact-sql.md	0	0	0	11	12
alter-function-transact-sql.md	0	0	0	3	4
alter-index-transact-sql.md	0	0	1	3	74
...					
Anzahl geänderte Dateien	4	19	58	138	351

Tabelle 1: Anzahl Änderungsblöcke bei 359 Dateien über fünf Zeiträume (Ausschnitt)

Die Daten geben bereits einige interessante Einsichten:

- Innerhalb einer Woche ändern im Ausschnitt "T-SQL Statements" wenig Dateien und auch die Anzahl Änderungsblöcke innerhalb einer Datei ist gering.
- Einige Dateien sind neu. Grund könnte die neue Version "SQL Server 2019" sein, die gegen Ende 2019 veröffentlicht wurde.
- Es gibt einige wenige Dateien, die besonders häufig ändern. Es handelt sich um SQL Befehle (T-SQL Statements), die mit jeder neuen Version (und in jedem Angebot) meistens umfangreich erweitert werden.
- Auffallend ist auch, dass bei einigen Dateien (z.B. create-table-transact-sql.md) die Anzahl Änderungsblöcke zuerst zunimmt und dann wieder kleiner wird. Dies könnte damit zusammenhängen, dass sich über einen längeren Zeitraum eine Datei stärker verändert und dadurch die kumulierten Änderungen zu komplex für die Library *difflib* werden. Es werden dann wahrscheinlich weniger, aber grössere Änderungsblöcke ausgewiesen.

3 Toolbox

In diesem Kapitel werden verschiedene Werkzeuge aus dem Bereich NLP beschrieben, mit denen die Texte bzw. die gefundenen Änderungsblöcke/Differenzoperationen ausgewertet worden sind. Die meisten Werkzeuge werden in der Masterarbeit eingesetzt, entweder mit bestehenden Python Libraries oder selbst programmiert.

3.1 Regular Expression

Regular Expression sind ein mächtiges Werkzeug, um ein vorgegebenes Muster (englisch: pattern) in einem Text zu finden oder auch ganze Textpassagen zu ersetzen. (Jurafsky & Martin, 2019, S. 2) bezeichnen es als wichtiges Werkzeug, um Text pattern zu beschreiben. Es lassen sich einzelne/mehrere Zeichen (alphanumerisch, numerisch etc.), ganze Wörter, Wortgruppen oder Sätze finden.

Die Lernkurve ist zu Beginn steil, um in Regular Expression einzusteigen. (Jurafsky & Martin, 2019, S. 3ff.) beschreiben die Funktionsweise in Kapitel 2 auf verständliche Art; dies erleichtert den Einstieg. Die Webseite https://regex101.com unterstützt mit einer graphischen Online-Oberfläche, um eigene Patterns auf einen Text anzuwenden und zu testen.

3.2 Stopwords

Stoppwörter können als Füllwörter bezeichnet werden, die wenig zum eigentlichen Inhalt des Textes beitragen (Beispiele wie "der", "in", "um"). In einem anderen Themenbereich, der Full-Text Search, werden Listen von Stoppwörtern verwendet, um Texte beim Katalogisieren als auch die Suchabfrage zu filtern. Dadurch soll die Relevanz der Suchergebnisse verbessert werden.

Python-Libraries wie *NTLK* und *spaCy* haben eine integrierte Stoppwörter-Liste. Solche Listen sollten aber behutsam eingesetzt werden. Je nach Domäne können vermeintliche Füllwörter wichtige Inhalte darstellen. Bei Programmiersprachen zum Beispiel sind Wörter wie "if", "else" Schlüsselwörter in der Syntax.

3.3 Part-of-speech (POS) tagging

Bei *Part-of-speech* (kurz *POS*) *tagging* geht es darum, Wörter eines Satzes/Textes zu klassifizieren, z.B. als Nomen, Verben, Adjektive. Diese POS-Tags erlauben es nachher zu interpretieren, welche Rolle die Wörter und Worttypen in einem Satz spielen (z.B. wer hatte mit wem ein Meeting) (Kochmar, 2019, S. 137).

Für die Masterarbeit können POS-Tags wertvoll sein, um irrelevante Änderungen zu verwerfen. Zum Beispiel gibt es stilistische Korrekturen von "if Query Store isn't enabled" nach "if **the** Query Store isn't enabled". Das Wort "the" erhält das POS-Tag "DET" (für determiner[11]). Bei der Umsetzung kann geprüft werden, ob solche Tags als irrelevante Differenz ausgeschlossen werden können.

In Python bieten u.a. die Libraries *NTLK* und *spaCy* ein POS-Tagging an.

[11] Ein Determiner zeigt an, ob ein Nomen eine bestimmte (z.B. der, die, das) oder unbestimmte (z.B. ein, eine) Einheit einer Klasse bezeichnet (Universal Dependencies, n.d.).

3.4 Dependency Parsing

Mit *Dependency Parsing* kann die Bedeutung von Wörtern in einem Satz ermittelt werden. Extrahierte Informationen wie "wer trifft wen und wann" helfen, Suchanfragen (z.B. "Wann findet das Meeting statt?") gezielt zu beantworten.

Der Beispielsatz und die folgenden Erläuterungen stammen von (Kochmar, 2019, S. 137):
"On Friday board members meet with senior managers.".

Dependency Parsing versucht die Satzstruktur zu analysieren. Die vorher beschriebenen POS-Tags helfen, die Wörter nach Verben, Nomen etc. zu unterscheiden. Ausgehend von einem root-Wort (oft ein Verb, im Beispiel "meet") können abhängige Wortgruppen ermittelt werden. Das Verb "meet" stellt dabei die Aktion dar. Ein oder mehrere Wörter hängen nun von der Aktion ab. Wortgruppen (wie "senior managers") weisen innerhalb der Gruppe wieder Abhängigkeiten auf. Die wichtigen Wörter in einer Gruppe werden "head" bezeichnet. Die anderen Wörter einer Gruppe tragen zusätzliche Informationen bei (genannt "dependants").

Auseinandergenommen sind "members" und "managers" solche *heads* von Gruppen, die von "meet" abhängen. "board" und "senior" sind zusätzliche Details zu den heads.

Das Zusatzwort "with" lässt Rückschlüsse zu, die Frage nach "wer" mit "wem" zu beantworten.

Die Python Library *spaCy* bietet neben dem POS-Tagging auch ein Dependency Parsing an.

3.5 Edit Distance / Minimum Edit Distance

Die *Edit Distance* hilft festzustellen, wie ähnlich zwei Texte sind. Dabei werden die Anzahl Operationen (Einfügen, Löschen, Ersetzen) ermittelt, um den Ausgangstext in den Zieltext umzuwandeln. Als Algorithmus wird oft die Levenshtein Distance verwendet, um die minimale Distanz zu ermitteln (*Minimum Edit Distance*). Die Bezeichnungen Minimum Edit Distance und Levenshtein Distance werden denn auch austauschbar verwendet (Wikipedia contributors, Edit distance, 2020).

Nachfolgend ein einfaches Beispiel mit den Zeichenfolgen "Levenstein" und "Levenshtein":

Input	L	e	v	e	n	s		t	e	i	n
Output	L	e	v	e	n	s	h	t	e	i	n
Operation							Insert				

Tabelle 2: Beispiel Operationen für Edit Distance

Die Tabelle 2 zeigt, dass sich zwischen "Levenshtein" und "Levenstein" genau ein Buchstabe unterscheidet. Mit der Einfüge-Operation (Insert) des Buchstabens "h" wird aus dem ersten Text der zweite. Je weniger Operationen notwendig sind, desto ähnlicher sind die Texte.

Es gibt auch andere Algorithmen als Levenshtein. Die Python Library *difflib* verwendet eine erweiterte Variante des Algorithmus "*Gestalt pattern matching*" von Ratcliff und Obershelp[12]. Dieser Algorithmus versucht möglichst lange gleichbleibende Zeichenfolgen zu ermitteln. Im Gegensatz zur Levenshtein Distance können dadurch mehr Operationen notwendig sein (keine Minimum

[12] Der Algorithmus wird in der Klasse *SequenceMatcher* eingesetzt (Python Standard Library (difflib), 2019).

Edit Distance). Begründet wird dies damit, dass der Algorithmus Differenzen ausarbeitet, die für Leute richtig ("look right") erscheinen (Python Standard Library (difflib), 2019).

3.6 Word Error Rate und Word Edit Distance

Die *Word Edit Distance* ermittelt analog der Edit Distance die Operationen fürs Einfügen, Löschen und Ersetzen, jedoch pro Wort. Anstatt der absoluten (rohen) Distance-Zahl ist auch eine Liste der betroffenen Wörter und Operationen hilfreich. Andere Tools können dann die Operationen prüfen: z.B. kann mit den POS-Tags (Part-of-Speech) ermittelt werden, ob relevante oder nicht relevante Wörter in den Operationen betroffen sind, oder der Thesaurus kann benutzt werden, um Ersetzungen von inhaltlich gleichwertigen Wörtern zu erkennen.

Die *Word Error Rate* wird als Metrik u.a. bei Speech Recognition Applikationen verwendet (Wikipedia contributors, Word error rate, 2020). Analog der Minimum Edit Distance werden die Anzahl Operationen fürs Einfügen, Löschen und Ersetzen gezählt, jedoch pro Wort, wobei meistens die Levenshtein Distanz verwendet wird (Wikipedia contributors, Word error rate, 2020).

Die Word Error Rate (r) wird berechnet aus den Anzahl Operationen (Op), gemäss Word Edit Distance, im Verhältnis zu den Anzahl Wörtern des Zieltextes (tW = truth Words)[13]:

$$r = \frac{Op}{tW}$$

Die Rate kommt zum Einsatz, um Sätze und Texte herauszufiltern, die sehr hohe Änderungen aufweisen. Bei diesen Texten können weitere Algorithmen meistens nicht sinnvoll angewandt werden, weil die Differenzen nicht aussagekräftig sind, z.B. wenn sie stark umgeschrieben oder Textblöcke verschoben wurden.

Im Internet existiert Python Code[14], um die Word Error Rate zu ermitteln. Dieser Code dient als Basis für die eigene Umsetzung. Jedes Wort eines Textes wird in ein eindeutiges Zeichen umgewandelt. Die Word Edit Distance wird dann anhand der Zeichen berechnet. Nachfolgend ein Beispiel für die Umwandlung:

Wörter	ende	**gut**	alles	**gut**
Zeichen	B	**C**	A	**C**

Tabelle 3: Beispiel Umwandlung Wörter in eindeutige Zeichen

Die Zeichen in der zweiten Zeile können jegliche Werte annehmen (hier wird der ANSI-Zeichensatz verwendet, um lesbare Zeichen zu erzeugen). Das Wort "gut" kommt zweimal vor und weist somit zweimal das gleiche Zeichen "C" auf. Entsprechend wird ein Wort, das in zwei zu vergleichenden Texten vorkommt, ebenfalls mit dem gleichen Zeichen repräsentiert. Die Edit Distance kann nun verwendet werden, um mit der Zeichenfolge "BCAC" die Distanz zu einer anderen Zeichenfolge zu berechnen und die Wörter zurückzugeben.

Die Edit Distance wird in der Masterarbeit nicht mit der Levenshtein Distance, sondern mit dem *SequenceMatcher* von *difflib* ermittelt (siehe 3.4, Edit Distance / Minimum Edit Distance, weiter oben). Die betroffenen Operationen werden mit der Methode "get_opcodes" des Objekts SequenceMatcher zurückgegeben.

[13] Die Bezeichnung "truth" wurde aus dem Code von (present 8x8, 2020) übernommen, weil es den Divisor schön beschreibt.
[14] https://github.com/jitsi/jiwer (present 8x8, 2020)

3.7 Lemma

Wörter können verschiedene Formen aufweisen. Nomen können in Einzahl, Mehrzahl oder konjugiert für Dativ, Genetiv etc. anders geschrieben sein. Verben können unterschiedliche Zeitformen annehmen, wie Präsens, Präteritum etc.

Mit einem Lemmatizer können die verschiedenen Wortformen auf die Basisform reduziert werden. Diese Basisform heisst *Lemma* (Kochmar, 2019, S. 128).

Eine andere Form, Wortformen zu reduzieren, sind Stemmer. Sie kommen z.B. oft bei der Full-Text Search zur Anwendung. Mit einem Algorithmus wird der Stamm ermittelt, z.B. dem Porter-Stemmer.

Der grosse Vorteil von Lemmatizer (oder Lemmas) gegenüber Stemmern ist, dass die Basisform wieder ein Wort bildet, das in einem Wörterbuch gefunden werden kann. Dies ist beim Stemmer nicht der Fall. (Kochmar, 2019, S. 128)

Die Library *spaCy* bietet einen integrierten Lemmatizer an. In dieser Masterarbeit werden an verschiedenen Stellen Lemmas für Text- oder Wortvergleiche verwendet.

spaCy hat dabei eine Besonderheit: Für englische Pronomen (z.B. its, it, you etc.) kann kein Lemma gebildet werden, weil das Pronomen vom Kontext abhängt. spaCy gibt dann als Lemma den Wert "-PRON-" aus[15].

3.8 Spell Checking

Texte weisen oft Rechtschreibfehler auf. Mit einer Rechtschreibprüfung (*Spell Checking*) sollen fehlerhafte Wörter und Texte aufgespürt und entweder manuell oder automatisch korrigiert werden.

Im Kontext von Differenzen spielen Rechtschreibfehler nur eine Rolle, wenn ein Fehler in einem Text korrigiert wurde oder ein neuer Fehler hinzugekommen ist und sich eine Differenz bildet.

Für Python gibt es einige Libraries für Rechtschreibprüfungen. Die Library *pyenchant* (Merejkowsky, 2020) ist auf verschiedenen Systemen wie Windows und Linux einsetzbar und bietet verschiedene Sprachen.

3.9 Thesaurus

Der *Thesaurus* ist ein Wörterbuch für Synonyme und Antonyme. Synonyme sind verschiedene Wörter mit gleicher Bedeutung; Antonyme mit gegensätzlicher Bedeutung (Sarkar, 2019, S. 525).

Ob zwei unterschiedliche Wörter als Synonyme gelten, hängt auch vom Kontext ab. Gerade in der Informatik gibt es viele Synonyme, die in den ursprünglichen Thesauri nicht abgebildet sind.

Als Beispiel gibt es im Englischen die Begriffe "*Select* a button" oder "*Click* a button". Die Wörter "Select" und "Click" sind im englischen Thesaurus nicht zusammen aufgeführt.

[15] Beschreibung von Personal Pronouns in *spaCy*: https://spacy.io/api/annotation#lemmatization

Für die Masterarbeit wird deshalb ein spezifischer Thesaurus erarbeitet, der mit verwandten Begriffen aus der Informatik umgehen kann. Der Thesaurus wird manuell erstellt und erweitert.

Es gibt noch einen zweiten Thesaurus in der Masterarbeit, nämlich für Änderungen von Produktnamen. Microsoft ändert öfters die Namen von Produkten oder Komponenten.

Aktuelles Beispiel ist die Umbenennung von "Azure SQL Data Warehouse" in "Azure Synapse Analytics". Diese Umbenennung ist nur bei der Ankündigung relevant für Entwickler und Datenbankadministratoren. Innerhalb einer Hilfeseite ist die Umbenennung irrelevant, da die zugehörigen Informationen immer noch das gleiche Produkt betreffen.

Der zweite Thesaurus wird ebenfalls manuell erstellt und gepflegt und enthält die Namensänderungen von Produkten.

3.10 Word Similarity

Auf den ersten Blick lässt der Begriff "Word Similarity" auf einen Thesaurus und Synonyme schliessen. Jedoch geht es bei *Word Similarity* darum, für Wörter (oder Texte) anhand der umgebenden Wörter eine Wahrscheinlichkeit zu berechnen, dass Wörter – im gleichen Kontext verwendet – eine hohe semantische Ähnlichkeit haben.

Word Similarity kann bei der Masterarbeit verwendet werden, um Änderungen im Text auf Ähnlichkeiten zu prüfen, z.B. die Änderung von "*After* this option is enabled" nach "*Once* this option is enabled".

Eine Art der Berechnung sind "count-based" Methoden wie die "Latent Semantic Analysis" (LSA). Die andere Art basiert auf voraussagenden (predictive) Methoden mit neuronalen Netzwerken, die anhand umgebender Wörter (Word embeddings) versucht, mittels Wortsequenzen das Wort vorherzusagen. (Sarkar, 2019, S. 232)

Word Similarity Modelle werden mit einem grossen Korpus[16] trainiert. Es gibt verschiedene Modelle, u.a. das *Word2Vec* Modell von Google, das *GloVe* Modell (entstanden an der Stanford University) sowie das *FastText* Model von Facebook. Sie gehören zur zweiten Art.

FastText ist eine Erweiterung von *Word2Vec* und basiert nicht auf Wörtern, sondern jedes Wort wird mit einer Liste von n-grams[17] repräsentiert. Dadurch ergeben sich zwei Vorteile. Morphologische Abweichungen (z.B. Deklinationen) werden besser abgedeckt und seltene Wörter werden besser berücksichtigt, da deren n-grams mit höherer Wahrscheinlichkeit auch durch andere Wörter abgedeckt sind. (Sarkar, 2019, S. 270)

Die Library *spaCy* integriert ein *Word2Vec* Modell, jedoch sind diese Vektoren nur in den grossen Textmodellen (mit Endung "lg" für large) enthalten. Die kleinen Modelle verwenden eine Vereinfachung, die aber auch Similarity-Abfragen zulassen[18]. Es ist zu beachten, dass das Modell[19] nicht mit IT-spezifischen Texten trainiert ist.

---

[16] Korpus ist eine grosse, strukturierte Sammlung von Texten oder Textdaten (Sarkar, 2019, S. 51). Beispiele sind speziell angefertigte Sammlungen wie WordNet und Brown Corpus sowie im weiteren Sinne Wikipedia und die Bibel.

[17] n-grams unterteilen ein Wort in Subwords (Sarkar, 2019, S. 270). Ein bi-gram (zwei Zeichen) unterteilt das Wort "learn" in die Subwords (le)(ea)(ar)(rn)

[18] https://spacy.io/usage/vectors-similarity (AI, kein Datum)

[19] Die Datengrundlage des Modells konnte nicht eruiert werden.

Für *FastText* gibt es eine entsprechende Library in Python, mit der man Wörter vergleichen – und bei Bedarf eigene Modelle trainieren kann. Um einen IT-Bezug bei den Texten herzustellen, wird für diese Masterarbeit ein eigenes *FastText*-Modell anhand aller SQL Server Hilfeseiten berechnet. Aus 22'948 Dateien ergeben sich 13.86 Millionen Wörter und ein 84MB grosses Modell.

Bei Tests mit dem eigenen *FastText*-Modell zeigt sich eine Tendenz zum Overfitting (Wortvergleiche ergeben regelmässig eine sehr hohe Ähnlichkeit, mit Werten über 95%). Deshalb werden die beiden Verfahren von *spaCy* und dem eigenen *FastText*-Modell kombiniert, um ähnliche Wörter zu bestimmen.

3.11 Nicht verwendete Werkzeuge

sclite

sclite wird bei Speech Recognition eingesetzt, um den hypothetischen Text mit einem Referenztext zu vergleichen mit Text oder Word alignment (National Institute of Standards and Technology, 2018). Die Anwendung ist nicht direkt für Python verfügbar und die Installation für Windows muss selbst kompiliert werden. Deshalb wird diese Anwendung nicht in Betracht gezogen. Als Alternative wird die Word Edit Distance mit *difflib* und SequenceMatcher verwendet.

4 Differenzen ermitteln

Der erste wichtige Teil der Masterarbeit ist, die *relevanten* Änderungsblöcke in den Dateien zu ermitteln. Viele Änderungen betreffen Rechtschreibkorrekturen, Metadatenänderungen und Formatierungen (Tabulatoren und Umbrüche), die man als *Noise* bezeichnen kann und die es herauszufiltern gilt.

Die relevanten Änderungsblöcke sind dann die Basis für die folgenden Kapitel, um die Differenzoperationen noch präziser auf Relevanz auszuwerten und in einem weiteren Schritt kompakt für die Benutzer anzuzeigen.

Als erstes wird die Vorgehensweise beschrieben, wie die Änderungsblöcke ermittelt und bereinigt werden. Danach folgt die technische Umsetzung, wie die einzelnen Herausforderungen mit Einsatz der Toolbox und von Python gelöst werden. Am Ende des Kapitels wird das Zwischenresultat erörtert.

4.1 Vorgehen/Methodik

Aus den vorhandenen Dateien (siehe Kapitel 2, Datengrundlage) wird als erstes ein Testobjekt bestimmt. Mit Python und der Library *difflib* können pro Datei und Intervall einerseits die Differenzen ermittelt und diese anderseits in eine separate Differenzdatei gespeichert werden.

Die Datei "alter-database-transact-sql-set-options.md" (nachfolgend *Testdatei* genannt) weist für das Intervall "Halbjahr" die mit Abstand meisten Änderungsblöcke auf. Diese Datei wird als Basis für das weitere Vorgehen verwendet, mit dem Ziel, möglichst viele verschiedene Muster von Differenzen in den Änderungsblöcken zu finden.

Wie in Abschnitt 2.4 Differenzen, Seite 6, dargelegt, können Änderungsblöcke zwischen zwei Dateiversionen nicht nur programmatisch, sondern auch graphisch in Diff-Programmen ausgegeben werden. Als Beispiel dient die Abbildung 10, die einen Ausschnitt der Differenzen in der *Testdatei* angezeigt, aufbereitet mit Visual Studio. Die Ursprungs- und Zieldatei werden dabei vertikal aufgelistet und die betroffenen Zeilen (Änderungsblöcke) und Unterschiede (Differenzoperationen) mit unterschiedlichen Farben markiert.

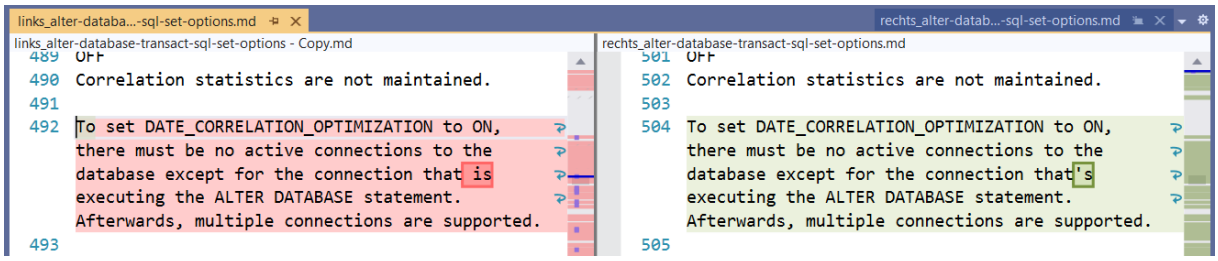


Abbildung 10: Beispiel von Änderungsblöcken und Differenzoperationen (angezeigt in Visual Studio)

Diese Darstellung mit der vertikalen Gegenüberstellung von Differenzen dient in der Folge als Metapher. Der Ursprungstext wird benannt als *links* (oder linke Seite) und der Zieltext als *rechts* (bzw. rechte Seite).

Mit den folgenden Schritten sollen in der *Testdatei* relevante Änderungsblöcke ermittelt und irrelevante entfernt werden:

- Ersten Überblick über die Differenzen verschaffen

- Markdown-Auszeichnungen entfernen
- Differenzen manuell kategorisieren (Goldstandard)
- Irrelevante Differenzen entfernen

4.1.1 Erster Überblick

Mit der Python Library *difflib* werden die Änderungsblöcke ermittelt. Die *Testdatei* beinhaltet über 400 Änderungsblöcke.

Die erste manuelle Durchsicht dieser Änderungsblöcke zeigt, dass ein Grossteil der Unterschiede auf Formatierungen und Markdown-Auszeichnungen zurückzuführen ist. Bei den Formatierungen handelt es sich um Leerschläge, Tabulatoren und Umbrüche, die *Whitespace* genannt werden. Die Markdown-Auszeichnungen sind Formatierungen, wie fett, unterstrichen etc.

Um die manuelle Kategorisierung zu vereinfachen, werden zuerst die Markdown-Auszeichnungen mit Regular Expression entfernt. Dadurch ist nur noch der Rohtext vorhanden. Mit *difflib* werden nun exakt 407 Differenzen ermittelt.

Danach werden die Whitespace-Differenzoperationen ebenfalls mit Regular Expression entfernt. Die Anzahl Änderungsblöcke verringert sich auf 155 (die 252 Formatierungsänderungen machen also mehr als die Hälfte aus). Diese Anzahl von 155 ist die Ausgangslage für die weiteren Schritte.

4.1.2 Markdown-Auszeichnungen entfernen

Markdown-Auszeichnungen sind spezifische Formatierungsanweisungen, um Text fett, unterstrichen, in einer Tabelle etc. darzustellen.

Es bietet sich an, gleich zu Beginn diese Auszeichnungen zu entfernen, damit am Schluss der Rohtext für die Differenzsuche übrigbleibt.

Der gleiche Schritt kann auch auf andere Formate angewendet werden. Wenn die Informationen z.B. als HTML, PDF oder Word geliefert werden, kann zuerst ein Bereinigungslauf für das entsprechende Format durchgeführt werden.

Die nachfolgende Liste zeigt einen Ausschnitt der Markdown-Auszeichnungen, die bereinigt werden.

Kategorie	Erläuterung / Kurzbeispiel aus Testdatei
Markdown Bold	Hier wurde eine Textpassage mit der Formatierung "Fett" gekennzeichnet oder die Formatierung entfernt. Dies erfolgt in Markdown mit zwei Sternchen vor und nach der Textpassage: `**OFF**`
Markdown (Hyper-)Links	Es kommt in der *Testdatei* häufig vor, dass ein Schlüsselwort auf eine andere Seite innerhalb der Hilfeseiten verlinkt wird. (Hyper-)Links werden in Markdown mit der Konvention [Beschreibung](URL) umgesetzt (siehe Beispiel in Abbildung 10, Seite 16). Inhaltlich ändert sich nichts, solange die [Beschreibung] und der ursprüngliche Text gleichbleiben. `[SELECT](..\..\select.md)`
Markdown Apostroph	Apostrophe, oder genauer Backspaces (`), werden in Markdown für das Kennzeichnen von Code verwendet. `` `SELECT @@VERSION` ``

Kategorie	Erläuterung / Kurzbeispiel aus Testdatei
Markdown Admonition	Admonition[20] ist eine spezielle Markdown-Formatierung. Text kann z.B. in einem Rahmen mit einem Icon "Important", "Note" oder "Warnung" ausgegeben werden. Solche Formatänderungen kommen relativ häufig vor. `> [!IMPORTANT]` `>` SET ANSI_NULLS also must be set to ON.

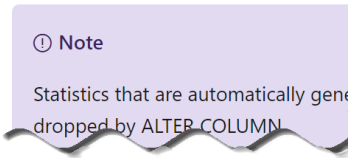


Tabelle 4: Ausschnitt der bereinigten Markdown-Auszeichnungen

Die weiteren Markdown-Auszeichnungen werden kurz aufgeführt:

- Weblink
- Zitat
- Überschrift
- Textzeile fett
- Einfache Fett
- Kursiv
- Tabellenstart
- Tabellenüberschrift
- Tabellenzeilen
- Codeblock
- Infoblock zu Beginn
- Escape-Zeichen

Beim Bereinigen ist darauf zu achten, dass die Anzahl Zeilen in der Datei nicht verändert wird. Grund ist, dass die Struktur der Datei gleichbleiben soll. Die Zeilennummern sind später wichtig, wenn einzelne Sätze anhand ihrer Position z.B. einer Überschrift zugewiesen werden.

4.1.3 Manuelle Kategorisierung

Die manuelle Kategorisierung hat mehrere Zwecke. In erster Linie sollen Muster und deren Quantifizierung ausgearbeitet werden. Anhand der Muster sollen v.a. irrelevante Änderungen aufgedeckt werden (z.B. Hinzufügen von Kommas). Die Muster können auch beurteilt werden, ob sie mit einfacher oder komplexer Programmlogik eliminiert werden können.

In der *Testdatei* gibt es über ein Dutzend Kategorien/Muster, die in der nachfolgenden Tabelle 5 beschrieben werden. Zu beachten ist, dass die Muster meistens in *Kombination* auftreten.

Aus der zweiten, grünen Zeile (Relevante Änderungen) wird sichtbar, dass *71 relevante Änderungsblöcke* in der *Testdatei* vorhanden sind. Diese 71 bilden die *Baseline* für die Bereinigung.

Kategorie	Anzahl	Erläuterung / Kurzbeispiel aus Testdatei
Whitespaces	252	Dieses Muster wurde bereits eliminiert, bevor die weiteren Kategorien ermittelt wurden. Es handelt sich – wie oben beschrieben – um Leerschläge, Tabulatoren und Umbrüche, und zwar wenn sie in keiner Kombination mit den nachfolgenden Kategorien auftreten.
Relevante Änderungen (*Baseline*)	71	Hier handelt es sich um relevante Änderungen oder um Differenzoperationen, die per Code nicht als irrelevant erkannt werden können (z.B. sehr komplexe Änderungen wie Textverschiebungen). Der Umgang mit diesen Änderungen muss in einem späteren Schritt geprüft werden.

[20] Der Begriff "Admonition" wird von der AsciiDoc Markdown-Hilfeseite übernommen (Allen & White).

Kategorie	Anzahl	Erläuterung / Kurzbeispiel aus Testdatei
POS-Tags (Determiner)	31	Bei dieser Kategorie wird der Text stilistisch verbessert, indem fehlende bestimmte oder unbestimmte Artikel hinzugefügt (oder entfernt) werden. Diese Artikel heissen auf Englisch "Determiner" und können über POS-Tags gefunden werden; dies gibt der Kategorie den Namen. Im nachfolgenden Beispiel wurde der Artikel/Determiner "the" entfernt: → Specifies that `the` Query Optimizer… → Specifies that Query Optimizer…
Similarity (16) (VersionInfoLater: 15)	31	Es sind Wörter oder ganze Wortblöcke ausgetauscht worden, die inhaltlich gleich sind. Jedoch handelt es sich nicht direkt um Synonyme. Zwei Beispiele sind nachfolgend aufgeführt: → The options described `below` are values → The options described `in the following sections` are values Ein weiteres Beispiel kommt immer wieder vor, nämlich eine Anpassung der Produktversion (Spezialfall VersionInfoLater) → Version 11 `through Version 15` → Version 11 `and later`
Word Contraction	17	Contractions sind verkürzte Wortformen (Sarkar, 2019, S. 136), die im Englischen häufig verwendet werden, z.B. it's. In der *Testdatei* wurden die langen Wortformen häufig in Contractions umgewandelt (siehe Abbildung 10, Seite 16). Es gibt Contractions in der positiven als auch negativen Form: → that `is` executing → that `'s` executing → that `is not` executing → that `isn't` executing
Semantic Synonyms (Synonyms)	16	Hier wird ein Wort ausgetauscht (z.T. auch mehrere Wörter hintereinander), ohne dass der Inhalt ändert. Für den Leser des Textes ist klar, dass es sich um die gleiche Bedeutung handelt. Jedoch sind die Wörter in einem Thesaurus nicht als Synonyme bekannt, da es sich um technische Wörter handelt oder weil es sich um eine Umformulierung handelt: → `Click` one of the following tabs → `Select` one of the following tabs
Punctuation	11	Änderungen von Satzzeichen führen auch zu Differenzen. Die Schwierigkeit ist, dass Satzzeichen bei Programmcode bzw. Syntax eine (wichtige) Rolle spielen können. → The option is ON`,` by default`,` for the master → The option is ON by default for the master
SentenceRebuilt (7) (davon passive Sätze: 1)	8	Bei dieser Änderung wurde der Satz oder Text umgebaut, aber der Inhalt bleibt der gleiche. Es gibt auch Sätze, die von aktiver in passive Form umformuliert werden oder vice versa. → Be set on `the system databases`: master, model and tempdb. → Be set on `the` master, model and tempdb `system databases`.
CRLF	7	Dieses Muster zeichnet sich dadurch aus, dass die Differenz nur auf der linken oder der rechten Seite vorhanden ist und dass Umbrüche entfernt oder hinzugefügt wurden.

Kategorie	Anzahl	Erläuterung / Kurzbeispiel aus Testdatei
Dash	7	Oft gibt es für Autoren die Herausforderung, ob zusammenhängende Wörter mit oder ohne Bindestrich geschrieben werden. → even `e-mailed` to other users → even `emailed` to other users
Lemma (Mehrzahl, Verbformen)	7	Häufig ändert sich die Ein-/Mehrzahl eines Wortes (Lexeme). Der Wortstamm (Lemma) bleibt dabei gleich. Es gibt auch andere Änderungen, die sich auf den gleichen Wortstamm beziehen, wie Präsens/Vergangenheit. → `Losses` of precision `don't` generate → `Loss` of precision `doesn't` generate
(Personal) Pronouns	7	Wörter oder Wortblöcke werden bei Wiederholungen oft mit Pronomen ersetzt, die sich auf die vorherigen Wörter beziehen. Es gibt einige Änderungen, bei denen Duplikate durch Pronomen ersetzt werden. Oder die persönlichen Pronomen (it, you) werden hinzugefügt oder entfernt. → enables the Query Store and configures `Query Store` parameters → enables the Query Store and configures `its` parameters → compiles a query and `it` runs a cached query plan → compiles a query and runs a cached query plan
Namensänderung (Renaming)	5	Microsoft hat eine sehr aktive Marketingabteilung. Produkte oder Komponenten ändern häufig den Namen, v.a. im Umfeld der Azure-Cloud. Dies widerspiegelt sich auch in der *Testdatei*: → Azure `SQL Data Warehouse` → Azure `Synapse Analytics`
Auxiliary Verbs	5	In den englischen Texten werden Hilfsverben (be, have) häufig hinzugefügt oder entfernt. Dies erfolgt oft auch mit dem Infinitiv Zusatzwort "to". → include characters `not` allowed in → include characters `that aren't` allowed in → `Is` the name of the database to be modified → The name of the database to be modified
Textverschiebung (BlockMove)	3	Diese Kategorie stellt Diff-Tools und -Libraries häufig vor grosse Herausforderungen. Ein Satz oder ganze Textblöcke werden innerhalb der Datei an eine andere Stelle verschoben. In Kombination mit den anderen Kategorien können diese Differenzen meistens nicht aufgelöst werden. Es entstehen wenige aber sehr grosse Differenzen. In der *Testdatei* ist diese Problematik auch vorhanden.
Stopwords	1	Stoppwörter werden v.a. bei der Full-Text Search entfernt, um relevante Datensätze oder Dokumente zu finden. Es besteht aber die Gefahr, dass zu viele Informationen verlorengehen, wenn Stoppwörter konsequent entfernt werden. Vor allem bei Hilfeseiten zur Programmierung könnten auch Programmier-Konstrukte als Stoppwörter gekennzeichnet werden. Stoppwörter als Differenz sollten nur entfernt werden, wenn sich der Inhalt nicht ändert. → To set READ_COMMITTED_SNAPSHOT ON or OFF → To set READ_COMMITTED_SNAPSHOT `to` ON or OFF

Kategorie	Anzahl	Erläuterung / Kurzbeispiel aus Testdatei
Spellchecking	1	Diese Kategorie umfasst Korrekturen von Rechtschreibfehlern oder neue Rechtschreibfehler. Es ist auffallend, dass sehr wenig Rechtschreibfehler (oder Korrekturen) vorhanden sind. Insgesamt lässt sich sagen, dass die Texte eine hohe Qualität bezüglich Rechtschreibfehler aufweisen.

Tabelle 5: Kategorisierung der Differenzen in der Testdatei

Alle Informationen zu den Änderungsblöcken sind in einer separaten Exceldatei hinterlegt, mit den Kategorien verbunden und mit dem Label "IsDiff" (J=relevant; N=irrelevant) markiert. Mit diesem Label ist eine Art *Goldstandard* festgelegt (Kochmar, 2019, S. 73), mit dem die Fortschritte verglichen werden können und auch automatische Auswertungen (z.B. mit einer Confusion Matrix) ermöglicht würden (der Fortschritt wurde in diesem Projekt manuell geprüft).

Die nachfolgende Abbildung 11 zeigt einen Ausschnitt aus der Exceldatei. Die Spalte *Part* gibt an, ob die Änderung auf der linken oder rechten Seite aufgetreten ist; *Lines* = welche Zeilen (von, bis) betroffen sind; *IsDiff* = Label für "Y=relevant; N=irrelevant"; *Reason* = Kategorisierung der Differenzen; *ReasonCount* = wie viele Kategorien betroffen sind[21]; *Assigned* = wie viele Kategorien rechts in den Spalten zugewiesen sind[22]; *Diff* = Kontrollspalte, wie viele Kategorien noch nicht zugewiesen sind (Diff sollte 0 sein); die restlichen (grünen) Spalten[23] bilden die *Kategorien* ab.

Part	Lines	IsDiff	Reason	ReasonCount	Assigned	Diff	DET	Pu
L	2193	N	Dash,Similarity	2	2	0		
L	2299	Y	Compatibility	1	1	0		
L	2321,2323	N	DET	1	1	0	x	
L	2327,2328	N	DET,Contraction	2	2	0	x	
L	2332,2334	Y	New Explanation	1	1	0		
L	2358,2362	N	DET,Pronoun,Synonym	3	3	0	x	
L	2370,2371	N	DET	1	1	0	x	
L	2375,2379	N	DET	1	1	0	x	
L	2383,2386	N	DET	1	1	0	x	
L	2392,2404	N	Pu[illegible]Synonym,DET,Contraction,Lo[illegible]	5	5		x	

Abbildung 11: Exceldatei mit Differenzen, Kategorien und Label "IsDiff"

Die Exceldatei hilft, schnell ein Beispiel für eine Kategorie zu finden und Filter für Auswertungen zu erstellen.

Ziel in der nächsten Phase ist es, möglich viele Änderungsblöcke zu eliminieren, um nahe an diese Anzahl 71 Änderungsblöcke heranzukommen.

4.1.4 Irrelevante Differenzen entfernen

Die erste wichtige Entscheidung ist, eine Triage zwischen einfachen und komplexen Kategorien (und deren Differenzen) vorzunehmen. Ohne viel Vorwissen lässt sich sagen, dass sich Kategorien wie "Dash", "CRLF", "Punctuation" als einfach umsetzbar einstufen lassen.

Kategorien wie "POS-Tags" und "Word Contraction" könnten ein bisschen aufwändiger sein.

[21] *ReasonCount* wird automatisch berechnet anhand der kommaseparierten Kategorien in Spalte "Reason"
[22] *Assigned* wird automatisch berechnet anhand der "x" in den grünen Kategorien-Spalten (z.B. Spalte "DET") ganz rechts.
[23] Der Wert "x" bei den Kategorien wird automatisch berechnet, indem der Titel der Kategorie in der Spalte "Reason" gesucht wird.

Zu den komplexen oder schwierigen Kategorien gehören "Similarity", "Semantic Synonyms" und "Textverschiebungen". Hier muss geprüft werden, welche Werkzeuge aus der Toolbox geeignet sind, die Änderungsblöcke und Differenzoperationen zu bewerten und zu eliminieren.

4.2 Technische Umsetzung

Bei der technischen Umsetzung geht es darum, die Änderungsblöcke und Differenzoperationen anhand der Kategorien aus Abschnitt 4.1.3, Manuelle Kategorisierung, Seite 18, zu eliminieren.

Nochmals als Ausgangslage: Es gibt einfache Kategorien (z.B. "Whitespaces", "Dash", "CRL" und "Word Contraction") und komplexe Kategorien ("Similarity", "Semantic Synonyms" und "Textverschiebungen"). Zusätzlich kann ein Änderungsblock eine beliebige Anzahl von Kategorien kombinieren.

Aus dem Data Warehouse Umfeld gibt es den Begriff "Cleansing", wenn ein ETL-(Extract-Transform-Load)-Prozess Daten aus einem Source-System lädt und dann zuerst bereinigen (cleanse) und evtl. transformieren muss, bevor die Daten ins Data Warehouse geladen werden können.

Deshalb wird nachfolgend der Begriff "Cleansing" verwendet, und es werden verschiedene "Cleansing"-Funktionen definiert. Diese *Cleansing-Funktionen* orientieren sich an den Kategorien, z.B. gibt es die Funktionen "CleanseDash", "CleansePunctuation", "CleanseWordContraction".

4.2.1 Cleansing – Grober Ablauf

Die Cleansing-Funktionen können nun wie mit einem Baukastensystem zu einzelnen Phasen kombiniert werden. In der nachfolgenden Abbildung 12 ist ein grober Ablauf skizziert mit den verschiedenen Phasen. Die auf den Kopf gestellte Pyramide verdeutlicht, dass in jeder Phase möglichst viele Änderungsblöcke bereinigt und möglichst wenige an die nächste Phase übergeben werden.

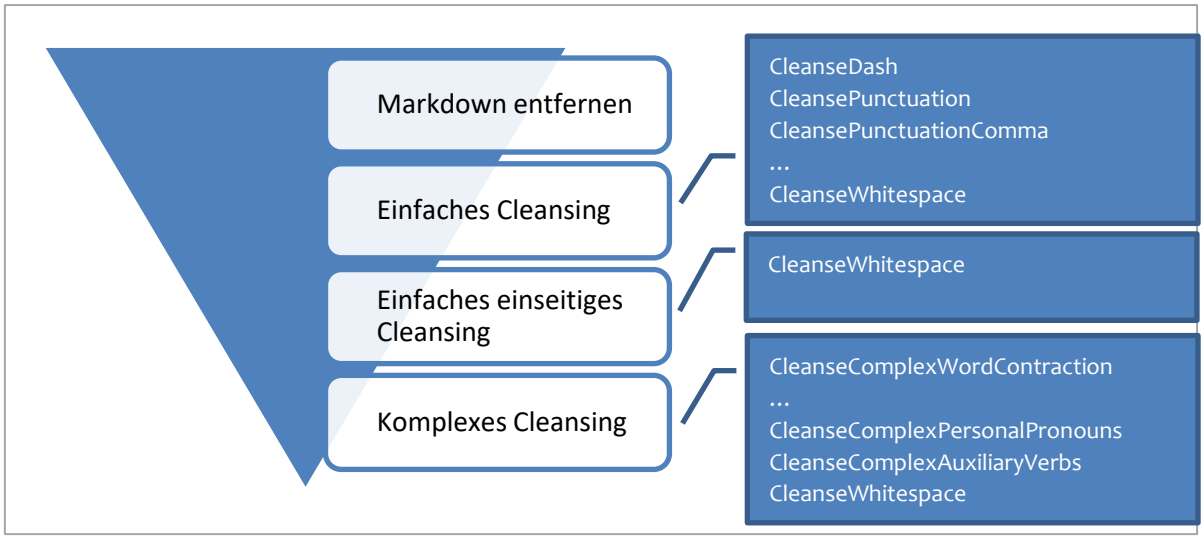


Abbildung 12: Eliminierung Differenzen - Grober Ablauf

Die Phasen werden kurz beschrieben. Die Cleansing-Funktionen werden danach erläutert.

Schritt	Beschreibung	Cleansing-Funktionen
Markdown entfernen	Beim Start werden alle Markdown-Auszeichnungen entfernt, damit der Rohtext für das Ermitteln der Differenzen zur Verfügung steht.	CleanseMarkdown
Einfaches Cleansing	In dieser Phase wird eine Reihe von Cleansing-Funktionen im Baukastensystem zusammengesetzt und in einer Schlaufe auf die Texte angewandt. Wenn die Texte links und rechts nach einem Funktionsaufruf gleich sind, kann der Änderungsblock aus der Liste entfernt werden. Ansonsten wird die nächste Cleansing-Funktion aufgerufen. *Wichtig* zu erwähnen ist, dass die Cleansing-Funktionen *unabhängig* auf den Text der linken und rechten Seite angewendet werden können (die Funktion benötigt den Text der anderen Seite nicht).	CleanseDash CleansePunctuation CleanseProductRenaming CleanseWhitespace
Einfaches einseitiges Cleansing	Dies ist ein Spezialfall des einfachen Cleansing, wenn nämlich nur die linke bzw. rechte Seite einen Änderungsblock aufweist. Dies ist der Fall, wenn bisheriger Text gelöscht wird oder neuer eingefügt wird. Für irrelevante einseitige Änderungsblöcke sind meistens Umbrüche verantwortlich. Im Baukasten ist nur eine Cleansing-Funktion notwendig; die Cleansing-Funktion "CleanseWhitespace" entfernt nämlich neben Leerzeichen und Tabuloren auch Umbrüche.	CleanseWhitespace
Komplexes Cleansing	Diese Phase ist aus zwei Gründen komplex. Einerseits sind die zugehörigen Kategorien wie Semantic Synonyms schwierig(er) umzusetzen und anderseits gibt es Abhängigkeiten zwischen linker und rechter Seite. Als Beispiel für die Abhängigkeit dient die Ein-/Mehrzahl eines Wortes. Die Lösung für Ein-/Mehrzahl ist, das Lemma zu vergleichen. Jedoch müssen dann die Texte links und rechts ausgewertet werden, um das Lemma von beiden Seiten extrahieren zu können. Im "Preprocessing"-Vorgang wird der Text für die komplexen Cleansing-Funktionen mit den minimal notwendigen Schritten bereinigt (z.B. mehrfache Leerzeichen auf eines reduzieren; jedoch möchte man das Satzende-Zeichen nicht entfernen, damit die Library *spaCy* die Satzkonstruktionen möglichst gut nachvollziehen kann).	Preprocessing: → CleansePunctuationComma → CleanseProductRenaming CleanseWordContraction CleanseNegativeWordContraction CleanseWordsWithPOSTagDET CleanseWordsWithSameLemma CleanseVersionInfoChangedToLater CleanseWordThesaurusSynonyms CleansePersonalPronouns CleanseAuxiliaryVerbs

Tabelle 6: Eliminierungsphase - Beschreibung der Schritte

Das *Baukastensystem* bringt einige Vorteile:

- Wenn im Verlaufe der Masterarbeit weitere Phasen im Eliminierungsprozess notwendig werden, können diese einfach eingeklinkt werden. Dies erlaubt eine iterative Vorgehensweise.
- Eine neue Phase kann mit bestehenden Cleansing-Funktionen ausgerüstet werden.
- Neue Cleansing-Funktionen können erstellt werden und bei einzelnen oder allen Phasen hinzugefügt werden.
- Die Reihenfolge, wie Cleansing-Funktionen auf den Text angewandt werden, kann pro Phase explizit festgelegt werden. Zum Beispiel werden die Whitespaces erst am Schluss entfernt, damit Wörter für das POS-Tagging extrahiert werden können.

Jede Phase soll aus Effizienzgründen möglichst viele Änderungsblöcke eliminieren. Es hat sich gezeigt, dass v.a. die komplexe Phase rechen- und speicherintensiv ist. Je weniger Änderungsblöcke dem komplexen Cleansing übergeben werden, desto weniger Extraktionsvorgänge von Wörtern und deren Attributen, wie POS-Tags, Lemma etc., müssen durchgeführt werden.

4.2.2 Markdown-Auszeichnungen

Markdown-Auszeichnungen werden mit Regular Expression entfernt.

Zu beachten ist, dass die Cleansing-Funktionen nur eine Teilmenge aller Markdown-Auszeichnungen abdecken, nämlich jene, die in den Testdateien vorkommen. Bei Bedarf können weitere Funktionen erstellt und hinzugefügt werden.

4.2.3 Einfache Cleansing-Funktionen

Die einfachen Cleansing-Funktionen sind mit Regular Expression umgesetzt. Beim Product Renaming kommt zusätzlich noch der selbsterstellte Thesaurus für Produktnamen zum Einsatz.

4.2.4 Komplexe Cleansing-Funktionen

Komplexe Cleansing-Funktionen verwenden ein oder mehrere Werkzeuge aus der Toolbox, die in Kapitel 3, Seite 10, aufgeführt sind. Nachfolgend werden die Cleansing-Funktionen beschrieben.

Zu beachten ist, dass es sich um Rule-basierte Bereinigungen handelt. Diese Regeln (d.h. Cleansing-Funktionen) sind aufgrund von manuellen Auswertungen der Differenzen zusammengestellt. Der Input für die Cleansing-Funktionen basiert auf kleingeschriebenem Text (lower-case), damit Änderungen zwischen links und rechts einfach(er) geprüft werden können[24].

4.2.4.1 Vereinigung (Union) von Diffs

Die erste Cleansing-Funktion betrifft ungenügende Resultate der Library *difflib* und hat weniger mit den eigentlichen Änderungen in einer Datei zu tun. Es kommt vor, dass Diff-Programme wie *difflib* Schwierigkeiten haben, Änderungsblöcke richtig aufzulösen. Dies trifft v.a. zu, wenn viele Differenzoperationen kombiniert werden mit dem Einfügen/Löschen von Sätzen oder Abschnitten.

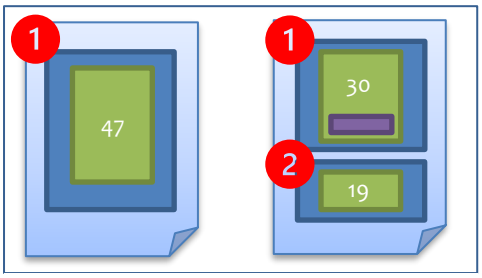


Abbildung 13: Ungleiche Änderungsblöcke

[24] Wenn das Python Projekt später überarbeitet bzw. erweitert wird, dann sollte diese Entscheidung nochmals überprüft werden. Evtl. gibt es bessere Resultate, wenn der Originaltext in den Cleansing-Funktionen verwendet wird. Zu beachten ist, dass die Entscheidung nicht ohne Aufwand rückgängig gemacht werden kann, da jede Cleansing-Funktion und auch das Zusammenspiel aller Funktionen überprüft werden muss.

Das Resultat ist dann ein grosser Änderungsblock auf der einen Seite und zwei oder mehrere kleine Änderungsblöcke auf der anderen (siehe Abbildung 13).

Eigentlich handelt sich auf beiden Seiten um je einen grossen Änderungsblock. Weil rechts ein neuer Satz hinzugefügt wird (kleiner violetter Kasten), teilt die Library *difflib* den grossen Block in zwei kleinere auf (dies passiert nicht immer, aber meistens dann, wenn der grosse Block viele weitere kleinere Differenzoperationen aufweist, v.a. direkt nach dem neuen Satz). Die zwei neuen kleineren Änderungsblöcke weisen nun 30 und 19 Zeilen auf.

Wenn die Änderungsblöcke verglichen werden, dann ergeben sich folgende Abweichungen:

Links	Rechts	Abweichung	Bemerkungen
47	30	17	*difflib* stellt den grossen linken Änderungsblock dem ersten kleinen Änderungsblock gegenüber. Dies ergibt eine Abweichung von 17 Zeilen
	19	-19	Für den zweiten kleinen Änderungsblock gibt es keine Entsprechung auf der linken Seite. Dies ergibt eine Abweichung von -19 (nämlich 0 – 19).

Tabelle 7: Vereinigung von Diffs – Abweichungen von Änderungsblöcken

Der Algorithmus sucht nach grösseren negativen und positiven Abweichungen, die direkt aufeinanderfolgen. Folgende Regeln wurden selbst festgelegt:

- Die aufeinanderfolgenden Abweichungen müssen mindestens 5 Zeilen (absolut) betragen.
- Die (absolute) Summe der beiden aufeinanderfolgenden Abweichungen darf nicht grösser als 2 sein[25] (dadurch ist sichergestellt, dass eine Abweichung positiv ist und die andere negativ).

Im Beispiel aus Tabelle 7 sind beide Regeln eingehalten. Die erste Abweichung von 17 ist grösser als 5, die zweite ist als absolute Zahl ebenfalls grösser als 5, nämlich abs(-19). Die Summe der beiden Abweichungen (17 + -19) ergibt -2; sie ist als absolute Zahl innerhalb des Threshold von 2.

Dieser erste Schritt sagt nur aus, ob die Änderungsblöcke von der Grösse her zusammenpassen könnten. In einem zweiten Schritt muss geprüft werden, ob die Word Error Rate nicht zu gross wird, wenn die Änderungsblöcke vereinigt werden. Der Threshold wird manuell auf 0.3333 festgelegt, d.h. es dürfen sich maximal 1/3 aller Wörter beim Vergleich der neu kombinierten Änderungsblöcke unterscheiden.

Links	Rechts	Word Error Rate	Bemerkungen
47	30+19	0.1123	In der Testdatei ergibt sich nach Vereinigung eine tiefe Word Error Rate; die beiden Änderungsblöcke können auf der rechten Seite für weitere Cleansing-Funktionen vereinigt werden.

Tabelle 8: Vereinigung von Diffs – Word Error Rate nach Vereinigung

Der vereinigte Änderungsblock kann nun wie alle anderen Änderungsblöcke mit den nachfolgenden komplexen Cleansing-Funktionen bereinigt werden.

4.2.4.2 Word contraction / Negative word contraction

Contractions sind verkürzte Wortformen (Sarkar, 2019, S. 136), die im Englischen häufig verwendet werden, z.B. it's. In der *Testdatei* wurden die langen Wortformen häufig in Contractions

[25] Eine Alternative ist, für die absolute Summe einen Threshold prozentual zu der grösseren Abweichung zu definieren, z.B. maximal 25%. Dies ergibt anhand der grösseren Abweichung von abs(-19) einen Threshold von 4 oder 5 Zeilen, je nach Rundung.

umgewandelt (siehe Abbildung 10, Seite 16). Contractions gibt es in der positiven als auch in der negativen Form. Deshalb werden zwei verschiedene Cleansing-Funktionen benötigt.

Eine Variante, Contractions zu vergleichen, ist mit einem Dictionary, der die Abkürzungen und ausgeschriebenen Formen kennt (Sarkar, 2019, S. 136). Das Problem bei diesem Vorgehen ist, dass die abgekürzte Form mehreren ausgeschriebene aufweisen kann, z.B. she's → she is / she has.

Eine zweite Variante ist, mit Regular Expression die Contractions auf der linken Seite zu ermitteln und dann im rechten Text zu überprüfen, ob der Textteil ausgeschrieben oder abgekürzt ist (und nachher umgekehrt von rechts nach links). Hier besteht die Herausforderung, die Textstelle im anderen Text zu finden, falls die beiden Texte links und rechts auch sonst viele Differenzoperationen aufweisen.

Eine dritte Variante besteht darin, den Text in Wörter zu extrahieren, die Word Edit Distance zu ermitteln und dann bei Differenzen das Lemma zu vergleichen. Hier ist die gleiche Herausforderung wie bei Variante zwei, ob die Word Edit Distance die Wörter an der gleichen Stelle einordnet. Zudem kann sich der Algorithmus für die Lemma-Ermittlung irren, wenn es um den Fall "she is" vs. "she has" geht.

Für die Masterarbeit wird zuerst die Variante eins verwendet (ohne die doppeldeutigen Contractions wie she's) und danach noch die Variante zwei. Die Herausforderung mit dem Finden der Textstelle wird so angegangen, dass die Umgebungswörter der Contraction einbezogen wird. Das folgende Beispiel soll dies verdeutlichen:

Beispieltext				
Links	detected by database recovery if `it's` truly an…			
Rechts	detected by database recovery if `it is` truly an…			
Vorgehen				
1. Regular Expression sucht im *linken* Text die Contraction inkl. Wörter vor-/nachher	if	it	's	truly
2. Regular Expression erstellt vier Gruppen (Apostroph ist nicht mehr enthalten)	if	it	s	truly
3. Regular Expression sucht anhand der 3 Gruppen (1, 2, 4) die Stelle im *rechten* Text	if	it		truly
4. Wenn gefunden: Regular Expression extrahiert im *rechten* Text die vier Gruppen	if	it	is	truly
5. Gruppe 3 von links/rechts wird per Code verglichen			"s" vs. "is"	
6. Falls eine Differenz in Schritt 5 gefunden wird, dann wird im *linken* Text die Contraction mit dem rechten Wert ersetzt.	if	it	~~'s~~ → is	truly

Tabelle 9: Word contraction - Vorgehen mit Regular Expression

Wenn sich die Contraction am Textanfang oder -ende befindet, dann entfällt entweder Gruppe 1 (Anfang) oder Gruppe 4 (Ende). Dies kann aber mit Regular Expression mit den Ankern ^ und $ berücksichtigt werden.

Bei der negativen Contraction müssen die Gruppen anders gebildet werden. Im Satz "… recovery it isn't truly …" gibt es fünf Gruppen (recovery it is t truly → t als bereinigte Gruppe 4). Deshalb gibt es der Einfachheit halber zwei Cleansing-Funktionen für positiv und negativ.

Die Vorgehensweise mit den Gruppen funktioniert gut, ist aber auf zwei Umstände anfällig. Erstens, wenn der Kontext im Änderungsblock mehrmals vorkommt und somit die falsche Stelle im Text ermittelt wird. Zweitens, wenn der Kontext im Änderungsblock noch von einer anderen Differenzoperation betroffen ist. Dieser Umstand kommt in der *Testdatei* einmal vor, weil das Wort

in Gruppe 4 ausgetauscht wurde (von "that" nach "which"). Mit den zwei Varianten können aber die meisten Differenzoperationen bereinigt werden.

4.2.4.3 POS-Tag "DET"

Beim Schreiben von Hilfetexten stellt sich die Herausforderung, ob gewisse Konzepte oder Komponenten mit einem bestimmten/unbestimmten Artikel genannt werden sollen. Als Beispiel dient "Query Optimizer" vs. "the Query Optimizer". Es gibt viele Änderungen in der *Testdatei*, die auf das Hinzufügen, Ändern oder Entfernen eines Artikels zurückzuführen sind.

Im Englischen heisst dieser Artikel "Determiner". Mit der Python Library *spaCy* können aus einem Text die Wörter extrahiert werden. Dabei werden pro Wort Attribute wie Lemma und POS-Tags ermittelt.

Die POS-Tags helfen im Fall der Determiner weiter, da die Artikel das POS-Tag "DET" aufweisen. Um Differenzen mit Determiner aufzuspüren, werden verschiedene Werkzeuge verwendet. Wie genannt wird *spaCy* verwendet, um die Wörter zu extrahieren und die POS-Tags zu ermitteln. Anhand der Wörter können mit der *Word Edit Distance* die Unterschiede auf Wortebene ausgegeben werden. Die Differenzen können dann geprüft werden, ob es sich bei den *POS-Tags* um Determiner ("DET") handelt.

Anhand eines fiktiven Beispiels wird die Vorgehensweise aufgezeigt:

Beispieltext						
Links	`The` Query Optimizer in `the` world…					
Rechts	Query Optimizer in `a` world…					
Schritt 1: Mit spaCy die Wörter im Text extrahieren						
Linker Text	The	Query	Optimizer	in	the	world
Rechter Text	Query	Optimizer	in	a	world	
Schritt 2: Mit Word Edit Distance die Operationen auf Wortebene ermitteln						
Linker Text	The	Query	Optimizer	in	the	world
Rechter Text		Query	Optimizer	in	a	world
(Operation)	*Delete*				*Replace*	
Schritt 3: Bei den Operationen das POS-Tag ermitteln (nur Operationen mit POS-Tag "DET" vergleichen)						
Linker Text	The	Query	Optimizer	in	the	world
(POS-Tag)	*DET*	-	-	-	*DET*	-
Rechter Text		Query	Optimizer	in	a	world
(POS-Tag)	-	-	-	-	*DET*	-
Schritt 4a: Wenn das Word von links nach rechts gelöscht wurde, dann links entfernen, um symmetrisch zu bleiben						
Linker Text	→ ~~The~~	Query	Optimizer	in	the	world
Rechter Text	→	Query	Optimizer	in	a	world
Schritt 4b: Wenn das Wort ersetzt (Replace) wurde, dann im alten (linken) Text ersetzen						
Linker Text		Query	Optimizer	in	~~the~~ → a	world
Rechter Text		Query	Optimizer	in	a	world
Schritt 5: Resultat weist keine Differenzen mehr auf						
Linker Text		Query	Optimizer	in	a	world
Rechter Text		Query	Optimizer	in	a	world

Tabelle 10: POS-Tag "DET" – Vorgehen mit Word Edit Distance und POS-Tag

Zu Schritt 4a ist zu bemerken, dass bei einem Insert (von links nach rechts) das rechte Wort herausgelöscht wird, statt beim linken Text hinzuzufügen. Hintergrund ist, dass das Löschen

einfacher umzusetzen ist, als den richtigen Ort im linken Text zu finden, wo das Wort eingesetzt werden muss. Diese Vereinfachung verändert den Satzinhalt des Zielzustandes[26].

Die Word Edit Distance kann auch Operationen mit mehreren Wörtern hervorbringen. Bei Insert/Delete-Änderungen wird Schritt 3 nur ausgeführt, wenn genau ein Wort eingefügt oder gelöscht wurde. Dies ist ebenfalls eine Vereinfachung.

Bei einer Replace-Operation werden Textblöcke mit bis zu vier Wörtern verglichen. Wenn mehr als ein Wort betroffen ist, wird eine weitere Edit Distance auf Basis der POS-Tags durchgeführt. "Determiner" POS-Tags werden mit "y" repräsentiert, alle anderen POS-Tags mit "n". Dadurch kann ermittelt werden, ob in der Differenzoperation ein Determiner hinzugefügt, geändert oder gelöscht wurde:

Beispieltext				
Links	In a future version …			
Rechts	In upcoming versions…			
Schritt 1: Mit spaCy die Wörter im Text extrahieren				
Linker Text	In	a	future	version
Rechter Text	In	upcoming	versions	
Schritt 2: Mit Word Edit Distance die Operationen auf Wortebene ermitteln				
Linker Text		a	future	version
(Operation)		*Replace*	*Replace*	*Replace*
Rechter Text		upcoming	versions	
(Operation)		*Replace*	*Replace*	
Schritt 3: POS-Tag "Determiner" für die Replace-Operation ermitteln und [y]es bzw. [n]o einsetzen				
Linker Text		a	future	version
(POS-Tag "Determiner")		y	n	n
Rechter Text		upcoming	versions	
(POS-Tag "Determiner")		n	n	
Schritt 4: Edit Distance der POS-Tags ermitteln				
Linker Text		y	n	n
Rechter Text			n	n
Schritt 5: Falls POS-Tags unterschiedlich und ein "Determiner" betroffen ist, dann links entfernen oder hinzufügen				
Linker Text		→ ~~a~~	future	version
Rechter Text			upcoming	versions

Tabelle 11: POS-Tag "DET" – Vorgehen bei Replace-Operation mit mehreren Wörtern

Mit der Edit Distance von POS-Tags können in Replace-Operationen Änderungen von Determiner erkannt werden. Die Anzahl Wörter einer Differenzoperation wird auf vier beschränkt, weil bei mehr Wörtern meistens ganze Sätze betroffen sind.

4.2.4.4 Same Lemma

Diese Cleansing-Funktion sucht nach gleichen Lemmas, wenn z.B. der Text bei Ein-/Mehrzahl ändert ("Losses" vs. "Loss"). Es funktioniert aber auch bei verschiedenen Zeiten eines Verbes, z.B. von Präsens nach Vergangenheit.

Das Vorgehen gleicht jener der Funktion POS-Tag "DET" in Abschnitt 4.2.4.3. Wiederum sind zuerst mit *spaCy* die Wörter aus dem Text zu extrahieren. Dabei wird auch gleich das *Lemma* pro

[26] Bei der nächsten Phase mit den Differenzoperationen hat diese Anpassung negative Auswirkungen auf die Suche nach dem Originaltext. Dieser Kompromiss zwischen Vereinfachung und schwieriger Originaltext-Suche wird aktuell in Kauf genommen.

Wort ermittelt. Mit der *Word Edit Distance* werden die Operationen auf Wortebene ausgegeben. Das Lemma ist zu prüfen, wenn in einer Differenzoperation genau ein Wort ersetzt wurde.

Anhand eines fiktiven Beispiels wird die Vorgehensweise aufgezeigt:

Beispieltext						
Links	`Losses` of precision `don't` generate					
Rechts	`Loss` of precision `doesn't` generate					
Schritt 1: Mit spaCy die Wörter im Text extrahieren						
Linker Text	Losses	of	precision	do	n't	generate
Rechter Text	Loss	of	precision	does	n't	generate
Schritt 2: Mit Word Edit Distance die Operationen auf Wortebene ermitteln						
Linker Text	Losses	of	precision	do	n't	generate
(Operation)	*Replace*			*Replace*		
Rechter Text	Loss	of	precision	does	n't	generate
(Operation)	*Replace*			*Replace*		
Schritt 3: Bei den Operationen das Lemma ermitteln						
Linker Text	Losses	of	precision	do	n't	generate
(Lemma)	*Loss*	-	-	*do*	-	-
Rechter Text	Loss	of	precision	does	n't	generate
(Lemma)	*Loss*	-	-	*do*	-	-
Schritt 4: Wenn das Lemma gleich ist, dann im alten (linken) Text ersetzen						
Linker Text	→Losses	of	precision	→do	n't	generate
Rechter Text	Loss	of	precision	does	n't	generate
Schritt 5: Resultat weist keine Differenzen mehr auf						
Linker Text	Loss	of	precision	do	n't	generate
Rechter Text	Loss	of	precision	do	n't	generate

Tabelle 12: Lemma – Vorgehen mit Word Edit Distance und Lemma-Vergleich

Bei Lemma-Vergleichen werden nur Replace-Operationen berücksichtigt. Grund dafür ist, dass sich Wörter geändert haben müssen (neue oder gelöschte Wörter sind nicht vergleichbar). Wenn eine Differenzoperation mehr als ein Wort umfasst, dann wird die Edit Distance mit der Lemma-Form berechnet, wie im nachfolgenden Beispiel dargestellt:

Beispieltext					
Links	In future `version` …				
Rechts	In upcoming `versions`…				
Schritt 1: Mit spaCy die Wörter im Text extrahieren					
Linker Text	In	future	version		
Rechter Text	In	upcoming	versions		
Schritt 2: Mit Word Edit Distance die Differenzen auf Wortebene ermitteln					
Linker Text		future	version		
(Operation)		*Replace*	*Replace*		
Rechter Text		upcoming	versions		
(Operation)		*Replace*	*Replace*		
Schritt 3: Lemma für die Replace-Operation ermitteln					
Linker Text		future	version		
(Lemma)		*future*	*version*		
Rechter Text		upcoming	versions		
(Lemma)		*upcoming*	*version*		

Schritt 4: Edit Distance der Lemma-Form ermitteln				
Linker Text (Lemma)		future	version	
Rechter Text (Lemma)		upcoming	version	
Schritt 5: Falls Lemma gleich (nicht unterschiedlich!) sind, dann links den Wert von rechts einfügen				
Linker Text	In	future	→ versions	
Rechter Text	In	upcoming	versions	

Tabelle 13: Lemma – Vorgehen bei Replace-Operation mit mehreren Wörtern

Der neue Text wird dabei auf der linken Seite eingesetzt, damit der Zieltext erhalten bleibt. Das Ersetzen erfolgt, wenn an der gleichen Position das gleiche Lemma für unterschiedliche Wörter auftritt (im Beispiel version vs. versions, mit dem gleichen Lemma "version").

4.2.4.5 Auxiliary Verbs

Beim Redigieren von Text kommt es häufiger vor, dass Hilfsverben eingefügt oder gelöscht werden. Hilfsverben sind u.a. "sein" und "haben" mit sämtlichen Deklinationen und Zeitformen. Der Inhalt des Textes ändert sich dadurch nicht, aber der Text wird als Änderungsblock ausgewiesen.

Mithilfe der Word Edit Distance, POS-Tagging und Dependency Parsing können solche Operationen mit Hilfsverben erkannt und bereinigt werden. Dabei zeigt sich, dass die Hilfsverben häufig mit anderen Füllwörtern benutzt werden. Diese Unterschiede können über die POS-Tags ebenso eliminiert werden.

In der Library *spaCy* können die Hilfsverben ermittelt werden mit dem POS-Tag "VERB" und den Dependency Labels "aux" (für auxiliary [verb]) und "auxpass" (für auxiliary [verb] passive).

Beispieltext						
Links	… values that can `be` set…					
Rechts	… values that can set…					
Schritt 1: Mit spaCy die Wörter im Text extrahieren						
Linker Text	values	that	can	be	set	
Rechter Text	values	that	can	set		
Schritt 2: Mit Word Edit Distance die Operationen auf Wortebene ermitteln						
Linker Text	values	that	can	be	set	
(Operation)				*Delete*		
Rechter Text	values	that	can		set	
(Operation)						
Schritt 3: POS-Tag und Dependency-Label für die Delete-Operation ermitteln						
Linker Text	values	that	can	be	set	
(POS-Tag)	*NOUN*	*ADJ*	*VERB*	*VERB*	*VERB*	
(Dependency Label)	*attr*	*nsubjpass*	*aux*	*auxpass*	*relcl*	
Rechter Text	values	that	can		set	
(POS-Tag)	*NOUN*	*ADJ*	*VERB*		*VERB*	
(Dependency Label)	*attr*	*nsubjpass*	*aux*		*relcl*	
Schritt 2: Text auf der linken Seite ersetzen (bzw. hier löschen)						
Linker Text	values	that	can	→	set	
Rechter Text	values	that	can		set	

Tabelle 14: Auxiliary Verbs – Vorgehen mit POS-Tag und Dependency-Label

Die Änderung kann noch weitere Wörter betreffen (z.B. "not allowed" nach "`that are` not allowed"). Wörter wie "that" werden auch entfernt, wenn es sich entweder um Stoppwörter oder

um folgende Dependency Labels handelt: (PART, NSUBJPASS, DET). Die Liste der Labels wurde manuell anhand von Beispielen in der *Testdatei* ermittelt.

4.2.4.6 Infinitive Verbs

Umformulierungen betreffen auch die Verbformen. Eine häufige Änderung erfolgt mit infinitiven Verben, z.B. von "to capture" nach "capturing" und vice versa.

Diese Änderung kann in folgender Form für das Beispiel bereinigt werden:

Beispieltext						
Links	… miss `to capture` important queries…					
Rechts	… miss `capturing` important queries…					
Schritt 1: Mit spaCy die Wörter im Text extrahieren						
Linker Text		miss	to	capture	important	queries
Rechter Text		miss	capturing	important	queries	
Schritt 2: Mit Word Edit Distance die Operationen auf Wortebene ermitteln (einfachheitshalber wird geprüft, ob die Operation auf der einen Seite zwei Wörter betrifft und auf der anderen Seite ein Wort → ist im Beispiel der Fall)						
Linker Text		miss	to	capture	important	queries
(Operation)			*Replace*	*Replace*		
Rechter Text		miss		capturing	important	queries
(Operation)				*Replace*		
Schritt 3: Prüfen, ob das POS-Tag "VERB" vorkommt inkl. dem Wort "to"						
Linker Text		miss	to	capture	important	queries
(POS-Tag)				*"VERB"*		
(Lemma)				*"capture"*		
Rechter Text		miss		capturing	important	queries
(POS-Tag)				*"VERB"*		
(Lemma)				*"capture"*		
Schritt 4: Beide Verben haben gleiches Lemma (Text auf der linken Seite ersetzen)						
Linker Text		miss	→	capturing	important	queries
Rechter Text		miss		capturing	important	queries

Tabelle 15: Infinitive Verbs – Vorgehen mit POS-Tags und Lemma

Der Algorithmus wird vereinfacht, indem nur Differenzoperationen berücksichtigt werden, bei denen auf der einen Seite zwei Wörter betroffen sind und auf der anderen Seite ein Wort. In den Testdaten zeigt sich, dass diese Operation häufig vorkommt. Dies ist eine Vereinfachung, jedoch können damit die meisten Vorkommnisse abgedeckt werden.

4.2.4.7 Personal Pronouns

Wiederholungen im Text können mit Personalpronomen vermieden werden. In den Texten sind oft stilistische Änderungen ersichtlich, wie nachfolgend aufgeführt:

Links	The following example enables the query store and configures `query store` parameters.
Rechts	The following example enables the query store and configures `its` parameters.

Personalpronomen werden in der Library *spaCy* speziell ausgewiesen. Das Lemma kann nicht bestimmt werden und weist deshalb den Wert "-PRON-" auf (siehe Abschnitt 3.7 Lemma, Seite 13).

Dieser Wert kann benutzt werden, um nach Personalpronomen zu suchen. Das Vorgehen ist mit den anderen Algorithmen vergleichbar:

- Mit *spaCy* die Wörter ermitteln und mit Word Edit Distance die Unterschiede im Änderungsblock finden
- Suchen, ob es Replace-Operationen gibt mit einem Wort, das das Lemma "-PRON-" aufweist (im Beispiel das Wort "its").
- Wenn ja, kann der rechte Text auf der linken Seite eingesetzt werden (im Beispiel wird der linke Text "query store" mit "its" ersetzt).

4.2.4.8 Sentence Rebuilt

Der Satzumbau gehört zu den komplexen Differenzen. Wörter werden innerhalb des Satzes verschoben. Das folgende Beispiel zeigt einen Satzumbau:

Links	The option cannot be set on `these system databases: master, model, and tempdb`.
Rechts	The option cannot be set on `the master, model, and tempdb system databases`.

Der Algorithmus beschränkt sich auf Änderungen innerhalb eines Satzes. Satzübergreifende Änderungen werden nicht berücksichtigt. Die Library *spaCy* kann neben den POS-Tags und Lemmas auch die Satzgrenzen ermitteln.

Ungenauigkeiten ergeben sich dabei mit Semikolon und Doppelpunkt. Nebensätze nach diesen Zeichen werden oft als eigenständige Sätze ausgegeben, so auch im linken Beispielsatz. Deshalb werden beim nachfolgenden Algorithmus diese beiden Satzzeichen durch Kommas ersetzt.

Eine weitere Vereinfachung ist, dass die Anzahl Sätze in den Änderungsblöcken gleich sein müssen. Grund ist, dass dadurch der erste Satz auf der linken Seite mit dem ersten Satz auf der rechten Seite verglichen werden kann (usw. für Satz zwei, drei etc.)

Folgende Schritte werden abgearbeitet (im Beispiel hat der Änderungsblock nur je einen Satz):

Beispieltext								
Links	The option cannot be set on `these system databases: master, model, and tempdb`.							
Rechts	The option cannot be set on `the master, model, and tempdb system databases`.							
Schritt 1: Semikolon und Doppelpunkte ersetzen mit Komma, damit der Nebensatz nicht separiert wird								
Links	The option cannot be set on these system databases, master, model, and tempdb.							
Rechts	The option cannot be set on the master, model, and tempdb system databases.							
Schritt 2: Mit spaCy und difflib die Wörter im Text extrahieren (Satzzeichen werden im Beispiel nicht angezeigt, werden aber ebenfalls als Differenzoperationen ausgegeben)								
Links	...	these	system	databases	master	model	and	tempdb
Rechts	...	the	master	model	and	tempdb	system	databases
Schritt 3: Prüfen, ob der linke und rechte Satz Differenzoperationen aufweisen. Zu beachten ist, dass die Library *difflib* die ersten drei Wörter als eine Replace-Operation ausgibt; man hätte für "system" und "databases" auf der linken Seite auch zwei Delete-Operationen erwarten können.								
Links	...	these	system	databases	master model tempdb			
(Operation)		*Replace*	*Replace*	*Replace*				
Rechts	...	the			master model tempdb		system	databases
(Operation)		*Replace*	*Replace*	*Replace*			*Insert*	*Insert*
Schritt 4: Irrelevante Wörter anhand der POS-Tags aus den Differenzoperationen entfernen (nämlich Tag "PUNCT" für Satzzeichen und "DET" für Determiner, u.a. the, these).								
Links	...	→ ~~these~~	system	databases				
Rechts	...	→ ~~the~~					system	databases
Schritt 5: Lemma der restlichen Differenzen ermitteln								
Links	...		system	databases				
(Lemma)			system	database				
Rechts	...						system	databases
(Lemma)							system	database

Schritt 6: Pro Seite eine Liste von eindeutigen Lemmas erstellen und vergleichen.								
Links		{ database, system }						
Rechts		{ database, system }						
Schritt 7: Wenn die Listen übereinstimmen, dann handelt es sich um inhaltlich gleiche Sätze.								
Links	...	the	master	model	and	tempdb	system	databases
Rechts	...	the	master	model	and	tempdb	system	databases

Tabelle 16: Sentence Rebuilt – Vorgehen mit POS-Tags und Lemma

Bei Schritt 4 ist zu beachten, dass Stoppwörter nicht herangezogen werden, um möglichst wenig Wörter auszuschliessen.

4.2.4.9 Spell Checking

Es gibt wenig Rechtschreibfehler in den Hilfeseiten. Deshalb sind Rechtschreibkorrekturen entsprechend selten.

Bei diesem Algorithmus geht es nicht darum, das Dokument auf Rechtschreibfehler zu prüfen (und zu verbessern), sondern bei Differenzoperationen zu eruieren, ob ein Fehler korrigiert oder ein neuer Fehler eingefügt wurde.

Gegenüber einer herkömmlichen Rechtschreibprüfung bieten die Differenzoperationen den Vorteil, dass das richtige Wort auf der linken oder rechten Seite vorhanden ist. Für das nachfolgende, fehlerhafte Beispielwort "Defalt" liefert die Rechtschreibprüfung mehrere Korrekturvorschläge, u.a. "dealt", "default", "desalt". Anhand des rechten Textes kann der richtige Vorschlag ausgewählt werden: "Default".

Links	examining the `is_auto_create_stats_on` column. `Defalt`
Rechts	examining the `is_auto_create_stats_on` column. `Default`

Wie in Abschnitt 3.8 Spell Checking, Seite 13, beschrieben, gibt es verschiedene Libraries, die eine Rechtschreibkorrektur anbieten. *pyenchant* ist einfach installier- und bedienbar und liefert für verschiedene Sprachen eine Rechtschreibprüfung. Für die Online-Hilfetexte wird die Sprache "en_US" (also das amerikanische Englisch) verwendet.

Die Umsetzung des Algorithmus ist konservativ gewählt. Das Wort darf sich nicht zu stark ändern, und es werden nur fehlerhafte (nichtexistierende) Wörter berücksichtigt (eine Rechtschreibkorrektur könnte nämlich auch zwei existierende Wörter prüfen, z.B. rosten vs. rasten). Im Beispiel oben existiert das Wort "Default", hingegen "Defalt" gibt es nicht.

Die Library *pyenchant* bietet die Möglichkeit, mit der "check"-Methode zu prüfen, ob das Wort fehlerhaft ist. Der Algorithmus ist wie folgt aufgebaut:

- Mit *spaCy* die Wörter und mit *difflib* die Word Edit Distance ermitteln.
- Es werden Differenzoperationen gesucht, bei denen je ein Wort links und rechts verändert wird (Korrekturen von auseinandergeschriebenen Wörtern werden damit nicht gefunden; dies ist eine Vereinfachung und kann in einer späteren Phase verbessert werden).
- Für das linke und rechte Wort einer solchen Replace-Differenzoperation wird die Edit Distance der zugrundeliegenden Buchstaben ermittelt. Wenn die Edit Distance Rate über dem (manuell festgelegten) Threshold von 0.3 liegt, dann hat sich das Wort zu umfangreich verändert. Dann wird keine Rechtschreibprüfung vorgenommen und der Algorithmus wird für die Wortüberprüfung abgebrochen. Im Beispiel beträgt die Rate 0.1429 (1/7), weil eine Operation existiert und das Zielwort "Default" sieben Zeichen aufweist.

Links	d	e	f	a		l	t
Rechts	d	e	f	a	u	l	t

- Wenn alle Vorprüfungen in Ordnung sind, dann erfolgt die Verifikation mit der Rechtschreib-Library: die Library muss eines der beiden Wörter als fehlerhaft erkennen ("Defalt") und das andere Wort als korrekt bzw. existierend ("Default").
- Wenn diese Prüfung erfolgreich ist, dann wird das nichtexistierende ("Defalt") mit dem existierenden Wort ("Default") ersetzt.

Die Rechtschreibprüfung wird zurückhaltend eingesetzt, um falsche Treffer zu vermeiden.

4.2.4.10 Word Duplication

Einige Texte weisen doppelte Wörter auf, z.B. wegen eines Verschreibers. Diese Doppelungen werden relevant, wenn sie korrigiert werden oder hinzukommen und somit als Differenz ausgewiesen werden (analog den Rechtschreibfehlern).

Nachfolgend ein Beispiel:

Links	restricted_user `allows allows` the members
Rechts	`restricted_user restricted_user` allows these members

Doppelte Wörter können mit der Word Edit Distance und Regular Expression bereinigt werden. Eine Vereinfachung ist, dass Doppelungen von zusammengesetzten Begriffen nicht unterstützt werden (z.B. die Doppelung: Word Error Rate *Word Error Rate*).

Folgende Schritte werden im Algorithmus ausgeführt:

Beispieltext						
Links	restricted_user `allows allows` the members					
Rechts	`restricted_user restricted_user` allows these members					
Schritt 1: Mit Regular Expression doppelte Wörter finden und als Liste aufbereiten (wenn die Liste leer ist, also keine Doppelungen gefunden wurden, dann kann der Algorithmus beendet werden)						
(Doppelungen als Liste)	{allows, restricted_user}					
Schritt 2: Mit Word Edit Distance die Operationen auf Wortebene ermitteln						
Linker Text		restricted _user	allows	allows	the	member
(Operation)				*Replace*	*Replace*	
Rechter Text	restricted _user	restricted _user	allows		these	member
(Operation)	*Insert*				*Replace*	
Schritt 3: Prüfen, ob Wörter aus den Differenzoperationen in der Liste der Doppelungen vorkommen ("allows" und "restricted_user" werden erkannt, dass sie in der Liste der Doppelungen vorkommen)						
Linker Text		restricted _user	allows	allows	the	member
(Ist in Liste der Doppelungen)				*Ja*		
Rechter Text	restricted _user	restricted _user	allows		these	member
(Ist in Liste der Doppelungen)	*Ja*					
Schritt 4a: Prüfen, ob das erste gefundene Wort "allows" eine Doppelung angrenzend aufweist, jedoch ausserhalb der Differenzoperation (die Differenzoperation "Replace" besteht aus zwei Wörtern "allows" und "the"; deshalb kann nur links von der Differenzoperation gesucht werden; siehe Erläuterungen weiter unten).						
Linker Text		restricted _user	allows	allows	the	member
(Ist in Liste der Doppelungen)				*Ja*		
Schritt 4b: Wenn das links anliegende Wort die Doppelung bestätigt, dann die Doppelung entfernen.						
Linker Text		restricted _user	allows	→	the	member
(Ist in Liste der Doppelungen)						

Schritt 5a: Prüfen, ob das zweite gefundene Wort "restricted_user" eine Doppelung angrenzend aufweist, jedoch ausserhalb der Differenzoperation (die Differenzoperation "Insert" besteht nur aus einem Wort; deshalb kann links und rechts von der Differenzoperation gesucht werden; siehe Erläuterungen weiter unten).					
Rechter Text	restricted _user	restricted _user	allows	these	member
(Ist in Liste der Doppelungen)	*Ja*				
Schritt 5b: Wenn das Wort links oder rechts die Doppelung bestätigt, dann die Doppelung entfernen.					
Rechter Text	→	restricted _user	allows	these	member
(Ist in Liste der Doppelungen)					
Schritt 6: Die Doppelungen sind bereinigt (die verbleibende Differenz mit "the" und "these" wird mit dem Algorithmus "Determiner" bereinigt, siehe Abschnitt 4.2.4.3 POS-Tag "DET, Seite 27).					
Linker Text		restricted _user	allows	the	member
Rechter Text		restricted _user	allows	these	member

Tabelle 17: Word Duplication – Vorgehen mit Regular Expression und Word Edit Distance

In den Schritten 4a und 5a werden Doppelungen nur ausserhalb der Differenzoperation gesucht. Der Grund ist, dass nur neu entstandene Doppelungen als Differenz relevant sind. Nicht geprüft werden bestehende Doppelungen auf beiden Seiten, z.B. wenn sich der Text wie folgt ändert:

Links	*restricted_user restricted_user* allows `the` members
Rechts	*restricted_user restricted_user* allows `these` members

Der kursive Text ist nicht Teil einer Differenzoperation(grau markiert). Der Text hat sich also in der Zeitachse nicht geändert und deshalb wird die Doppelung nicht bereinigt (auch wenn der Text inhaltlich eventuell falsch ist). Die gleiche Logik wird angewandt, wenn eine neue Doppelung innerhalb der Differenzoperation eingefügt wird:

Links	restricted_user allows `the` members
Rechts	restricted_user allows *only only* `these` members

Die Doppelung *only* wird neu eingefügt. Inhaltlich ist dies wahrscheinlich ein Fehler. Aber aus Sicht der Differenzen spielt es keine Rolle, ob die Doppelung bereinigt wird oder nicht, die Differenz mit *only* (als einem oder zwei Wörtern) wird bestehen bleiben. Deshalb wird dies nicht korrigiert.

4.2.4.11 Thesaurus Synonyms

Wenn Differenzoperationen durch den Austausch von Synonymen verursacht werden, dann können diese mit einem Thesaurus überprüft werden. Gerade bei IT-spezifischen Wörtern liefern die bestehenden Thesauri aber keine Treffer.

Deshalb wird ein Thesaurus manuell angelegt mit Wörtern, die anhand von Stichproben häufig in Differenzoperationen vorkommen. Ein solcher Thesaurus ist auf die Domäne "IT" ausgelegt. Vorteilhaft ist, dass die Liste jederzeit erweitert werden kann, jedoch entsteht daraus auch gleich der grösste Nachteil: die Wartung des Thesaurus ist mit Aufwand verbunden, weil neue Synonyme mit weiteren Stichproben gesucht werden müssen.

Folgende Synonyme sind in der Liste enthalten, und zwar basierend auf dem Lemma, damit möglichst viele Konjugationen abgedeckt sind:

Wort (Lemma)	Synonym (Lemma)
run	execute
SQL	query
click	select

Wort (Lemma)	Synonym (Lemma)
later	higher
later	future
later	upcoming
upcoming	future

Tabelle 18: Thesaurus Synonyms – Manuell erstellte Liste von Synonymen als Lemma

Bei der Tabelle 18 sind drei Punkte zu beachten:

- Die Spalten "Wort" und "Synonym" können in beide Richtungen verwendet werden, z.B. kann "run" mit "execute" ausgetauscht werden und vice versa.
- Einzelne Wörter können mehrmals vorkommen, da sie mit verschiedenen Synonymen ausgetauscht werden können (z.B. later, upcoming, future). In der Liste sind nur bekannte Muster gemäss Stichproben enthalten. Es wäre denkbar, dass auch "higher" und "future" Synonyme sind durch die Ableitung aus den Kombinationen mit "later". Jedoch kam diese Paarung "higher"/"future" nie vor.
- Das Lemma von "higher" bleibt in der Library *spaCy* "higher" (es ist nicht das Wort "high", wie man erwarten könnte).

Der Thesaurus kann nun wie folgt eingesetzt werden:

Beispieltext				
Links	… the connection `executing` the …			
Rechts	… the connection `running` the …			
Schritt 1: Mit Word Edit Distance die Operationen auf Wortebene ermitteln (und zwar "Replace"-Operationen, die genau ein Word links und rechts betreffen, also den Austausch von einem Wort)				
Linker Text	the	connection	executing	the
(Operation)			*Replace*	
Rechter Text	the	connection	running	the
(Operation)			*Replace*	
Schritt 2: Lemma der Wörter ermitteln				
Linker Text	the	connection	executing	the
(Lemma)			*execute*	
Rechter Text	the	connection	running	the
(Lemma)			*run*	
Schritt 3: Prüfen, ob beide Lemma als Kombination im manuellen Thesaurus aufgelistet sind.				
Linker Text	the	connection	executing	the
(Lemma im Thesaurus vorhanden?)			*execute* ☑	
Rechter Text	the	connection	running	the
(Lemma im Thesaurus vorhanden?)			*run* ☑	
Schritt 4: Bereinigen des linken Texts, wenn die Kombination der Lemma im manuellen Thesaurus aufgelistet ist				
Linker Text	the	connection	→ running	the
Rechter Text	the	connection	running	the

Tabelle 19: Thesaurus Synonyms – Vorgehen mit Word Edit Distance und Thesaurus

Die Einschränkung auf ein Wort in der Replace-Operation ist zurzeit eine Vereinfachung, die jedoch den grössten Teil solcher Synonym-Änderungen abdeckt. In Zukunft könnten auch Mehrwörter-Operationen in Betracht gezogen werden.

4.2.4.12 Word Similarity

Der manuelle Thesaurus aus dem vorherigen Abschnitt 4.2.4.11 hat seine Grenzen und eignet sich für Synonyme im engeren Sinne.

Stilistische Textänderungen, bei denen inhaltlich (semantisch) nichts ändert, können nicht immer mit einem Thesaurus abgedeckt werden.

Hier kommen Prüfungen auf Textähnlichkeit (Similarity) zum Einsatz. Im Abschnitt Toolbox (siehe 3.10 Word Similarity, Seite 14) wird beschrieben, wie trainierte Modelle dazu verwendet werden können.

Der Algorithmus, um diese Ähnlichkeiten zu finden, benutzt die zwei Modelle aus Abschnitt 3.10. Einerseits wird das selbsttrainierte *FastText*-Modell eingesetzt, das auf dem gesamten Umfang der Microsoft SQL Server Online-Hilfeseiten basiert. Andererseits kommt das *Word2Vec*-Modell der Library *spaCy* zum Einsatz.

Das selbsttrainierte Modell tendiert zu Overfitting. Deshalb ist das *spaCy Word2Vec*-Modell das primäre Kriterium für die Ähnlichkeit. Wenn der Vergleichswert von *spaCy* stufenweise unter einen gewissen Threshold fällt, dann kommt das selbsttrainierte FastText-Modell als sekundäres Kriterium zur Anwendung.

Die nachfolgende Tabelle zeigt die stufenweise Auswertung der beiden Kriterien. Die Threshold-Werte sind manuell festgelegt aufgrund von Stichproben.

Abstufungskategorie	spaCy Word2Vec-Modell	FastText-Modell	Bemerkungen
1	>= 0.75	>= 0.00	Wenn das *Word2Vec*-Modell einen höheren Wert als 0.75 für die Similarity eines Wortes (oder Textes) zurückliefert, dann wird es als semantisch gleich eingestuft (ohne auf das *FastText*-Modell zurückzugreifen; deshalb wird dort 0.00 eingesetzt[27])
2	>= 0.74	>= 0.9725	
3	>= 0.73	>= 0.9775	
4	>= 0.72	>= 0.9825	
5	>= 0.71	>= 0.9875	
6	>= 0.70	>= 0.9925	

Tabelle 20: Word Similarity –Abstufungskategorien anhand von Threshold-Werten

Es ist zu bemerken,

- dass im *spaCy Word2Vec*-Modell ein relativ tiefer Wert von 0.75 reicht, dass eine semantische Ähnlichkeit angenommen wird. Besser wäre natürlich, wenn dieser Threshold mit einem höheren Wert im Bereich von 0.85-0.90 konfiguriert werden könnte. Das Modell liefert aber für ähnliche Wörter relativ tiefe Werte, entsprechend sind selten Werte über 0.75 zu finden.
- dass die Bandbreite bei den Abstufungen von *spaCy Word2Vec* und *FastText*-Modell sehr klein ist (0.70 bis 0.75 bei *spaCy*, 0.9725 bis 0.9925 bei *FastText*). Tiefere Werte können im Moment[28] nicht benutzt werden, weil sonst viele False Positive Ähnlichkeiten gefunden werden.

Auch dieser Algorithmus ist konservativ ausgelegt, um falsche Ergebnisse möglichst zu vermeiden auf Kosten von Ähnlichkeiten, die dadurch nicht gefunden werden.

[27] Das FastText-Modell kann auch Minuswerte ausgeben, v.a. wenn Zahlen oder Satzzeichen betroffen sind. Diese Vergleiche werden dann in jedem Fall als fehlgeschlagen klassifiziert, auch wenn Word2Vec einen Wert höher als 0.75 ausgeben würde.

[28] Im Nachfolgeprojekt kann geprüft werden, ob das grosse, aber langsamere spaCy Word2Vec-Modell bessere Resultate liefert.

Beispieltext						
Links	… controls `whether` the query store …					
Rechts	… controls `if` the query store …					
Schritt 1: Mit Word Edit Distance die Operationen auf Wortebene ermitteln (und zwar "Replace"-Operationen mit bis zu drei Wörtern. Bei grösseren Änderungen ist es unwahrscheinlich(er), semantische Ähnlichkeiten zu finden)						
Linker Text	controls	whether	the	query	store	
(Operation)		*Replace*				
Rechter Text	controls	if	the	query	store	
(Operation)		*Replace*				
Schritt 2: Similarity-Werte der beiden Modelle ermitteln						
Linker Text	controls	whether	the	query	store	
Rechter Text	controls	if	the	query	store	
(spaCy Word2Vec-Modell)		*0.8009*				
(FastText-Modell)		*0.7489*				
Schritt 3: Similarity-Werte mit Abstufungsliste vergleichen (der *spaCy*-Wert liegt über 0.75 und der *FastText*-Wert von 0.7489 ist höher als der Threshold 0.00)						
Linker Text	controls	whether	the	query	store	
Rechter Text	controls	if	the	query	store	
(spaCy Word2Vec-Modell)		*0.8009* ☑				
(FastText-Modell)		*0.7489* ☑				
Schritt 4: Bereinigen des linken Texts, wenn die Similarity-Werte den Threshold übersteigen						
Linker Text	controls	→ if	the	query	store	
Rechter Text	controls	if	the	query	store	

Tabelle 21: Word Similarity – Vorgehen mit dem Word2Vec- und FastText-Modell

Bei jedem Aufruf loggt der Algorithmus die berechneten Werte und Abstufungskategorien. Die *Testdatei* hat aus 110 Abfragen 10 positive Ähnlichkeiten zurückgegeben:

Kategorie	Anzahl	Werte spaCy Word2Vec	Wert FastText-Modell	Wörter
1	4	0.8009	0.7489	if whether
1	1	0.8415	0.9622	causes returns
1	2	0.8351	0.9842	beginning starting
2	2	0.7454	0.9779	once after
4	1	0.7273	0.9833	produce trigger

Tabelle 22: Word Similarity – Ähnlichkeiten anhand Testdatei

Die manuelle Nachkontrolle zeigt, dass alle Resultate als True Positive eingestuft werden können. Einige Kombinationen kommen mehrmals vor, weil der gleiche Text in mehreren Angeboten (monikers) verwendet wird.

4.2.4.13 VersionInfo Later

Bei dieser Textänderung handelt es sich um eine Eigenart der Microsoft SQL Server Hilfeseiten. Gewisse Passagen werden mit der Information gekennzeichnet, für welche Version des SQL Servers sie gilt, z.B. "Applies to SQL Server (SQL Server 2012 through SQL Server 2019).

Dieser Text wird nun in vielen Fällen zukunftsfähig angepasst mit "Applies to SQL Server (SQL Server 2012 and later)". Bei der nächsten Version, z.B. 2021, muss der Text dadurch nicht mehr über-

arbeitet werden. Inhaltlich ergibt sich dadurch keine Änderung, so dass Differenzoperationen mit diesem Muster bereinigt werden können.

Zu beachten ist, dass in den Hilfeseiten die Versionsinformationen in separaten Dateien ausgelagert sind, die über Include-Anweisungen integriert werden. Der Text sieht in einer Hilfedatei vereinfacht so aus: "Applies to SQL Server (include[ssSQL11] through include[sscurrent]).

Es ist anzunehmen, dass der nachfolgende Algorithmus nur im Umfeld von Microsoft SQL Server Hilfeseiten nützlich ist:

Beispieltext				
Links	… include[ssSQL11] `through include[sscurrent]` …			
Rechts	… include[ssSQL11] `and later` …			
Schritt 1: Mit Word Edit Distance die Operationen auf Wortebene ermitteln (und zwar "Replace"-Operationen, die entweder das Wort "through" oder "later" aufweisen)				
Linker Text	through	include	[sscurrent]	
(Operation)	*Replace*	*Replace*	*Replace*	
Rechter Text	and	later		
(Operation)	*Replace*	*Replace*		
Schritt 2a: Prüfen, ob ein der beiden Texte den Block "and later" aufweist				
Linker Text	through	include	[sscurrent]	
(Wortblock "and later")				
Rechter Text	and	later		
(Wortblock "and later")	☑	☑		
Schritt 2b: Prüfen mit Regular Expression, ob einer der beiden Text das Pattern[29] "through include[xxxx]" aufweist				
Linker Text	through	include	[sscurrent]	
(Pattern "through include[xxx]")	☑	☑	☑	
Rechter Text	and	later		
(Pattern "through include[xxx]")				
Schritt 3: Prüfen, ob Schritt 2a und 2b auf je einem Text erfolgreich gefunden werden (wenn ja, den linken Text ersetzen)				
Linker Text	→ and	→ later		
Rechter Text	and	later		

Tabelle 23: VersionInfo Later – Vorgehen mit dem Regular Expression

4.3 Zwischenresultat

Nach Anwendung aller Cleansing-Funktionen kann nun geprüft werden, wie viele Änderungsblöcke in der *Testdatei* eliminiert sind.

In der nachfolgenden Tabelle 24 werden die Ausgangslage (Total), die eliminierten und die restlichen Änderungsblöcke aufgelistet. Die *Baseline* ist am Schluss als Vergleich aufgeführt.

Änderungsblöcke	Anzahl
Total	407
Eliminiert	-322
Rest	**85**
Baseline	71

Tabelle 24: Differenzen ermitteln – Zwischenresultat mit der Testdatei

[29] Das Pattern entspricht nicht dem effektiven Regular Expression Ausdruck, sondern ist eine Vereinfachung, wobei xxxx jeglichem Text entspricht. In den Hilfedateien sind auch die include-Anweisungen komplizierter aufgebaut und im Beispiel vereinfacht.

Die Tabelle 24 zeigt, dass mit den Cleansing-Funktionen der grösste Teil der irrelevanten Änderungsblöcke bereinigt ist.

Die Abweichung gegenüber der Baseline von 14 Änderungsblöcken (85 – 17) hat verschiedene Gründe:

- Viele Cleansing-Funktionen können nicht alle Differenzoperationen bereinigen. Gerade weil mehrere Algorithmen konservativ umgesetzt sind, werden einige Differenzen nicht gefunden.
- Bei Differenzoperationen mit mehreren Kategorien kann es sein, dass die einzelnen Algorithmen ihre Arbeit nicht vollständig umsetzen, weil die Änderungen zu komplex oder verschachtelt sind. Als Beispiel sei der Satzumbau genannt, der Wörter an eine andere Stelle setzt und somit die Patternsuche verhindern kann.
- Nicht alle Kategorien sind eins-zu-eins umgesetzt (z.B. Stoppwörter).

Das Zwischenresultat wird im Sinne des Paretoprinzips als gut genug bewertet, um die restlichen Änderungsblöcke an die nächste Phase zu übergeben. Dort werden die Differenzoperationen aufbereitet und auf Relevanz geprüft, damit die Daten für die Benutzer aufbereitet werden können.

Zu vermerken ist, dass das Zwischenresultat in einer Datei gespeichert werden kann (mit der Serialisierung der Objektdaten). In der nächsten Phase kann die Datei eingelesen und der Zwischenstand rekonstruiert werden (Deserialisierung). Die Bereinigung der Änderungsblöcke ist je nach Datei zeitaufwändig (bei der *Testdatei* dauert es ca. 1 - 2 Minuten). Die weiteren Programmieraufgaben können viel schneller umgesetzt werden, wenn das Zwischenergebnis eingelesen werden kann, statt jedes Mal berechnet wird.

5 Relevante Differenzoperationen

Ausgangspunkt sind die Änderungsblöcke aus dem vorherigen Kapitel. Jedoch sind die Änderungsblöcke keine geeignete Einheit, dem Benutzer als Differenz anzuzeigen:

- Ein Änderungsblock kann sehr gross sein und komplexe Differenzoperationen aufweisen, die für den Benutzer nicht durchschaubar sind.
- Ein grosser Änderungsblock kann aber auch nur ein/zwei kleine Differenzoperationen aufweisen, die dann im Text untergehen könnten.
- Ein grosser Änderungsblock kann mehrere Überschriften (in der Dokumentstruktur) und/oder Moniker (Angebote) betreffen.
- Die Relevanz ist bei Änderungsblöcken, v.a. grossen, schwierig zu ermitteln.

Deshalb werden die Änderungsblöcke auf die Differenzoperationen reduziert. Als Einheit wird der einzelne Satz gewählt. Die Differenzoperation allein gibt den Benutzern zu wenig Kontext. Mit einem Satz kann der Kontext der Änderung besser aufgezeigt werden.

Für die Relevanzbestimmung werden die Sätze mit verschiedenen Metadaten angereichert.

In diesem Kapitel wird die Vorgehensweise beschrieben. Einfache technische Umsetzungen werden direkt bei der Methodik erläutert.

5.1 Vorgehen/Methodik

Die Aufteilung in einzelne Sätze und das Ermitteln der Relevanz erfolgt in drei weiteren Phasen:

- Die Sätze ermitteln und mit Zusatzinformationen (Metadaten) anreichern
- Die Sätze kategorisieren, ob sie für den Benutzer relevant sind oder nicht
- Die Sätze in einer Datei für die weiteren Schritte zwischenspeichern

Die drei Phasen werden in den nachfolgenden Abschnitten beschrieben.

5.1.1 Sätze ermitteln und anreichern

5.1.1.1 Satzextrahierung

Um die auszuwertenden Informationen zu reduzieren, werden als Erstes die Sätze aus den Änderungsblöcken extrahiert. Es werden nur Sätze berücksichtigt, die von einer (oder mehreren) Differenzoperation betroffen sind. Tabelle 25 zeigt, wie aus einem Änderungsblock mit einer Differenzoperation der betroffene Satz extrahiert wird.

Einheit	Links	Rechts
Änderungsblock mit einer Differenzoperation	many database set options can be configured for the current session by using set statements and are often configured by applications when they connect. session level set options override the alter database set values. the database options described `below` are values that can set for sessions that do not explicitly provide other set option values.	many database set options can be configured for the current session by using set statements and are often configured by applications when they connect. session level set options override the alter database set values. the database options described `in the following sections` are values that can set for sessions that do not explicitly provide other set option values.

Einheit	Links	Rechts
Extrahierter Satz	the database options described `below` are values that can set for sessions that do not explicitly provide other set option values.	the database options described `in the following sections` are values that can set for sessions that do not explicitly provide other set option values.

Tabelle 25: Extrahieren von Sätzen aus Änderungsblöcken

Die Library *spaCy* ermittelt die Satzgrenzen innerhalb eines Textes. Zu beachten ist, dass der Text keine grossgeschriebenen Wörter enthält. Grund ist, dass der *bereinigte* Text als Basis dient. Bereinigt heisst, dass der Text in Kleinschreibung umgewandelt wurde (siehe Abschnitt 4.2.3 Einfache Cleansing-Funktionen, Seite 24), und dass die Cleansing-Funktionen den Text verändert haben, um Differenzoperationen zu eliminieren. Diese Kleinschreibung hat nun für das Ermitteln der Sätze einen Nachteil:

- Die Library *spaCy* kann Sätze weniger gut erkennen, wenn die normale Gross-/Kleinschreibung entfällt. Gerade bei Satzzeichen wie einem Doppelpunkt wird die Satztrennung oft falsch ermittelt.

Wie in Abschnitt 4.2.3 beschrieben kann eine Alternativlösung mit Gross-/Kleinschreibung im Nachgang dieser Masterarbeit geprüft werden, da der Einfluss auf die Cleansing-Funktionen ebenfalls in Betracht gezogen werden muss. Die Library *spaCy* hat auch Probleme mit Syntaxbeschreibungen oder Codebeispielen. Diese weisen keine Sätze im klassischen Sinne auf, entsprechend zufällig ist die Satzaufteilung. Im Anschluss an die Masterarbeit ist zu prüfen, ob bei spezifischen Blöcken wie Syntax auf Sätze zu verzichten ist und dafür ganze Abschnitte oder Änderungsblöcke zu verwenden und auszuwerten sind[30].

5.1.1.2 Originalsätze finden

Ziel ist, dass die Benutzer in der aufbereiteten Ansicht nicht die bereinigten, sondern die Originaltexte sehen. Deshalb wird als nächstes für jeden bereinigten Satz der Originalsatz ermittelt.

Satz	Originalsatz	Extrahierter bereinigter Satz
Links	The database options described `below` are values that can be set for sessions that **don't** explicitly provide other set option values.	the database options described `below` are values that can set for sessions that **do not** explicitly provide other set option values.

Tabelle 26: Unterscheide Originalsatz / bereinigter Satz

Dies ist technisch eine Herausforderung, weil der Satz von Cleansing-Funktionen ab und zu umfangreich verändert wird. Tabelle 26 zeigt einen Originalsatz, der bereinigt wurde. Die fettmarkierten Stellen sind bereinigt (*don't* vs. *do not*). Die Suche funktioniert somit nicht mehr mit einem reinen Textvergleich. Ein mehrstufiges Verfahren ermittelt deshalb den Originalsatz[31]:

- Als erstes wird der reine Textvergleich eingesetzt, um alle einfachen Fälle abzudecken.
- Im erfolglosen Fall wird danach der Schnittmengen-Operator (intersect) verwendet. Die bereinigten Wörter des Satzes werden als Set aufbereitet (d.h. jedes Wort ist genau einmal enthalten). Es wird nach einem Satz gesucht, der jedes Wort des Sets aufweist (alle Wörter des Sets sind in der Schnittmenge mit diesem Satz). Diese Vorgehensweise ist nützlich, wenn sich die Satzstruktur geändert hat, oder wenn Wörter bei der Bereinigung herausgelöscht wurden.

[30] Dies könnte bedeuten, dass die Programmierlogik komplexer wird, weil die nachfolgende Logik nun für Sätze und Syntaxblöcke umzusetzen ist.

[31] Zu beachten ist, dass die Suche nicht in der ganzen Datei erfolgt, sondern im Original-Änderungsblock, zu dem der Satz gehört.

- Als letztes Mittel wird die Edit Distance verwendet. Es wird nicht eine Differenz gesucht, sondern übereinstimmende Zeichen (Equals-Operationen). Im Kasten rechts ist ein Beispiel für die Edit Distance Operationen dargestellt: die obere Reihe bildet den Original-Änderungsblock ab, die untere Reihe den gesuchten bereinigten Satz. "E" entspricht einer Equals-Operation, wenn ein Zeichen übereinstimmt (Operation D = Delete, Operation I = Insert). Der bereinigte Satz ist normalerweise nur eine Teilmenge des Änderungsblocks, deshalb sollten viele "D"-Operationen vor und nach dem Satz vorhanden sein (um den oberen in den unteren Text zu konvertieren). Ziel ist, dass eine möglichst hohe Anzahl "E"-Operationen in der unteren Reihe ausgewiesen wird. Wenn mehr als 90% der Zeichen in der unteren Reihe überstimmen, dann kann angenommen werden, dass der Originalsatz vom ersten bis zum letzten E herauskopiert werden kann (im Programmcode werden die *spaCy* Satzbegrenzungen verwendet werden für ein genaueres Resultat).

```
DDDDDDDDEEEEEEEEE EDDDDDDDDD
EEEEEEEEEIE
```

Mit diesen Methoden können praktisch alle Originaltexte gefunden werden, jedoch ist die aktuelle Umsetzung noch nicht perfekt (z.T. werden veränderte Sätze nicht oder nicht ganz erkannt).

5.1.1.3 Zusatzinformation anreichern

Der letzte Vorgang in dieser Phase reichert die Sätze mit Zusatzinformationen (Metadaten) an, die für die Beurteilung der Relevanz wichtig sein können. Die Metadaten ergeben sich aus der Dokumentstruktur, aus Markdown-Auszeichnungen in der Originaldatei und Informationen des Änderungsblocks.

Ziel ist es, mit den Zusatzinformationen in Tabelle 27 eine Aussage zu treffen, ob ein Satz bzw. eine Differenzoperation relevant ist für die Benutzer (der Zusatztext "Technisch" zeigt auf, woher die Informationen stammen bzw. ermittelt werden):

Zusatzinformation	Herkunft	Beschreibung
Operation	Änderungsblock	Gibt an, ob der zugehörige Änderungsblock eine Insert-Operation (neuer Block), eine Delete-Operation (Block gelöscht) oder eine Replace-Operation (Block geändert mit einer oder mehreren Differenzoperationen) ist. *Technisch:* Der Wert wird aus der vorherigen Phase "Differenzen ermitteln" übergeben.
Differenzoperation	Satz	Wenn ein Änderungsblock eingefügt wird (Operation Insert), dann ist auch die Differenzoption ein Insert. Dasselbe gilt für die Delete-Operation. Bei einer Replace-Operation kann es aber vorkommen, dass ein Wort oder ein ganzer Satz innerhalb des Änderungsblocks eingefügt (Differenzoperation Insert), gelöscht (Differenzoperation Delete) oder geändert wurde (Differenzoperation Replace). Die Operation des Änderungsblocks und die Differenzoperation können dann unterschiedlich sein. *Technisch:* Die Differenzoperation hängt vom folgenden Wert "Origin" ab. Die Vorgehensweise wird im Anschluss an diese Tabelle erläutert.
Origin	Änderungsblock	Dieses Attribut bezieht sich darauf, wie eine Differenzoperation zustande kommt. Die beiden definierten Werte "Native" und "Split" werden im Anschluss an diese Tabelle erläutert.

Zusatzinformation	Herkunft	Beschreibung
DiffIndex	Änderungsblock	Eindeutiger Index des Änderungsblocks, damit nachverfolgt werden kann, welche Sätze zu einem Änderungsblock zusammengehören. *Technisch:* Der Wert wird aus der vorherigen Phase "Differenzen ermitteln" übergeben.
SentenceNumber (intern DiffNumber)	Änderungsblock	Eindeutige Nummer für den Satz innerhalb eines Änderungsblocks, damit die Reihenfolge der Sätze im Änderungsblock bekannt ist. *Technisch:* Der Wert wird innerhalb der verarbeitenden Schlaufe über alle Sätze berechnet.
LineNumber Von/Bis links	Änderungsblock	Gibt an, bei welcher Zeilennummer der linke Satz beginnt und bei welcher Zeilennummer er aufhört. Diese Information kann für die nachfolgenden Metadaten benutzt werden, z.B. um zu ermitteln, zu welchen Überschriften ein Satz gehört. *Technisch:* Es werden die Anfangs- und Endzeilennummern des Änderungsblocks als Basis verwendet, die aus der vorherigen Phase übergeben werden. Zuerst werden Start- und Endposition des Satzes innerhalb des Änderungsblockes gesucht. Anhand dieser Positionen können die Zeilenumbrüche vor und innerhalb des Satzes und somit die Zeilennummern berechnet werden.
LineNumber Von/Bis rechts	Änderungsblock	dito. für den Satz rechts
IsSentence	Änderungsblock	Ist True, wenn ein kompletter Satz geändert hat. *Technisch:* Die Library *spaCy* liefert die Bestandteile (Wörter und Satzzeichen) des Satzes und die Library *difflib* die Differenzoperationen. Wenn alle Wörter (und Satzzeichen) eine Differenzoperation aufweisen, dann ist der Wert True. Diese Vorgehensweise gilt auch für die nachfolgenden Werte "IsMultiSentence", "IsSingleWord" und "IsFragment".
IsMultipleSentence	Änderungsblock	Ist True, wenn die Differenzoperation mehrere Sätze umfasst. Dies kann ausnahmsweise vorkommen, wenn die Library *spaCy* die Satzbegrenzungen nicht richtig erkennt oder bei Satz-übergreifenden Differenzoperationen.
IsSingleWord	Änderungsblock	Ist True, wenn nur ein Wort im Satz geändert hat.
IsFragment	Änderungsblock	Ist True, wenn mehrere Wörter im Satz geändert haben, aber nicht der komplette Satz.
IsInfoblock	Dokumentstruktur	Jede Markdown-Hilfedatei hat einen einleitenden Infoblock, der Metadaten zur Datei enthält, z.B. Datum, Autor, Reviewer, Keywords für die Suche, DefaultMoniker (für welche Angebote der Text geschrieben ist). Der Infoblock ist nicht relevant für die Benutzer. *Technisch:* In der Dokumentstruktur werden für den Infoblock die Zeilennummern für Anfang und Ende ermittelt. Anhand der Zeilennummern des Satzes kann geprüft werden, ob er in den Bereich des Infoblocks fällt. Dieses Vorgehen gilt auch für die anderen Zusatzinformationen, die mit der Dokumentstruktur ausgewertet werden (IsCodeblock, CodeblockLanguage, Heading, Moniker).
IsCodeblock	Dokumentstruktur	Codebeispiele werden in der Datei mit einer Markdown-Anweisung gekennzeichnet, damit der Code mit einem speziellen Font/Hintergrund formatiert wird. Wenn der Satz innerhalb eines Codeblocks ist, gibt diese Information True aus.

Zusatzinformation	Herkunft	Beschreibung
CodeblockLanguage	Dokumentstruktur	Wenn der Satz Teil eines Codeblocks ist, dann wird mit diesem Attribut die Programmiersprache ausgegeben. Markdown-Anweisungen können nicht nur den Codeblock markieren, sondern auch eine Sprache definieren. Wenn keine Sprache definiert ist, dann handelt es sich (normalerweise) um allgemeine Syntaxbeschreibungen. In der Testdatei wird "sql" benutzt. Es sind aber auch andere Programmiersprachen möglich, z.B. "python".
Heading1	Dokumentstruktur	Bezeichnung der 1. Überschrift des Abschnitts, in dem sich der Satz befindet (dies ist normalerweise der Titel des Dokuments) Beispiel: "ALTER DATABASE SET options (Transact-SQL)"
Heading2	Dokumentstruktur	dito. für 2. Überschrift, z.B. der Abschnitt "Syntax"
Heading3	Dokumentstruktur	dito. für 3. Überschrift
Heading4	Dokumentstruktur	dito. für 4. Überschrift
Heading5	Dokumentstruktur	dito. für 5. Überschrift
IsHeading	Dokumentstruktur	Ist True, wenn der Satz/Text eine Überschrift betrifft.
MonikerText	Dokumentstruktur	Alle Monikers, die auf den Satz zutreffen. Ein Satz (bzw. dessen Abschnitt) kann keinem, einem oder mehreren Moniker (Angeboten) zugewiesen sein. Die Dokumentstruktur verwaltet alle Monikers. Beispiel ">=sql-server-2016||>=sql-server-linux-2017||=sqlallproducts-allversions"
MonikerListe	Dokumentstruktur	In der Dokumentstruktur werden auch detaillierte Informationen zu den Monikers gespeichert. Unter anderem wird der MonikerText auf die einzelnen Angebote aufgeteilt (Trennzeichen ist das doppelte Pipe-Zeichen |). Beispiel: >=sql-server-2016 >=sql-server-linux-2017 =sqlallproducts-allversions
MonikerDetails	Dokumentstruktur	Um die Monikerliste noch besser auswerten zu können, werden auch die drei Moniker-Bestandteile "Angebot", "Vergleichsoperator" und "Version" geführt: sql-server >= 2016 sql-server-linux >= 2017 sqlallproducts = allversions
IsMoniker	Dokumentstruktur	Ist True, wenn der Text eine Moniker-Kennzeichnung ist
IsDefaultMoniker	Dokumentstruktur	Ist True, wenn der Text eine Defaultmoniker-Kennzeichnung ist. Dateien können am Anfang im Infoblock einen Defaultmoniker aufweisen. Wenn sich dieser ändert oder hinzugefügt wird, dann betrifft der Text den Defaultmoniker.

Tabelle 27: Zusatzinformationen (Metadaten) zu einem Satz

5.1.1.4 Zusatzinformation "Origin" und "Differenzoperation"

Der Wert "Origin" wird hier genauer erläutert. Differenzoperationen können auf zwei verschiedene Arten in Sätze aufgeteilt werden.

Der Normalfall wird mit dem Wert "Native" beschrieben. Wenn ein Änderungsblock hinzugefügt (Insert-Operation) wird, dann weisen auch die zugehörigen Sätze "Insert"-Differenzoperationen auf (das gleiche gilt für Delete). Alle Sätze haben *nativ* den gleichen Status wie der Änderungsblock. Nachfolgend ein Beispiel mit einer Insert-Operation (Texte sind nur rechts vorhanden).

Insert-Operation	Origin	Links	Rechts
Änderungsblock *(Insert)*	-		The MAX_DOP limit is set per task. It is not a per request or per query limit. This means that during a parallel query execution, a single request can spawn multiple tasks which are assigned to a scheduler.
1. Satz extrahiert *(Differenzoperation Insert)*	Native		The MAX_DOP limit is set per task.
2. Satz extrahiert *(Differenzoperation Insert)*	Native		It is not a per request or per query limit.
3. Satz extrahiert *(Differenzoperation Insert)*	Native		This means that during a parallel query execution, a single request can spawn multiple tasks which are assigned to a scheduler.

Tabelle 28: Attribut Origin mit Wert "Native" für Insert-Operationen

Wenn ein Änderungsblock geändert wird (Replace), dann hängt der Wert "origin" von den linken und rechten Textblöcken ab. Wenn der Satz auf beiden Seiten vorkommt, dann handelt es sich weiterhin um eine native Replace-Differenzoperation. Hat ein Satz jedoch kein Gegenstück auf der anderen Seite, dann wird dies mit dem Wert "Split" gekennzeichnet.

Das nachfolgende Beispiel in Tabelle 29 zeigt, dass der Änderungsblock ungleich verteilt ist. Auf der linken Seite ist nur ein Satz vorhanden, auf der rechten sind es drei. Beim Extrahieren entspricht der erste Satz einer nativen Replace-Differenzoperation. Für die zwei weiteren Sätze gibt es kein Gegenstück auf der linken Seite, sie werden als Insert-Differenzoperation mit dem Origin "Split" gekennzeichnet.

Replace-Operation	Origin	Links	Rechts
Änderungsblock *(Replace)*	-	Create a workload group named newReports.	Create a workload group named newReports which uses the Resource Governor default settings, and is in the Resource Governor default pool. Workload groups are containers for a set of requests and are the basis for how workload management is configured on a system. Workload groups provide the ability to reserve resources for workload isolation, contain resources, define resources per request, and adhere to execution rules.
1. Satz extrahiert *(Differenzoperation Replace)*	Native	Create a workload group named newReports.	Create a workload group named newReports which uses the Resource Governor default settings, and is in the Resource Governor default pool.
2. Satz extrahiert *(Differenzoperation Insert)*	Split		Workload groups are containers for a set of requests and are the basis for how workload management is configured on a system.

Replace-Operation	Origin	Links	Rechts
3. Satz extrahiert *(Differenzoperation Insert)*	Split		Workload groups provide the ability to reserve resources for workload isolation, contain resources, define resources per request, and adhere to execution rules.

Tabelle 29: Attribut Origin mit Wert "Native" und "Split" für Replace-Operationen

In der technischen Umsetzung werden die Werte "Differenzoperation" und "Origin" gemeinsam ermittelt, wenn die Sätze aus dem Änderungsblock extrahiert werden:

Änderungsblock Operation	Differenzoperation	Origin	Bemerkungen
Insert	Insert	Native	
Delete	Delete	Native	
Replace	Replace	Native	Wenn der Satz ein Gegenstück hat (links und rechts).
Replace	Insert	Split	Wenn der rechte Satz kein Gegenstück auf der linken Seite hat (er wurde hinzugefügt).
Replace	Delete	Split	Wenn der linke Satz kein Gegenstück auf der rechten Seite hat (er wurde gelöscht).

Tabelle 30: Attribute Differenzoperation / Origin – Technische Umsetzung

Die Liste aller Sätze mit den Zusatzinformationen ist Basis für die nächste Phase "Kategorisierung".

5.1.2 Sätze kategorisieren

In dieser Phase wird festgelegt, ob ein Satz dem Benutzer als Differenz angezeigt wird. Die Sätze werden dabei kategorisiert, aus welchem Grund sie angezeigt oder nicht angezeigt werden.

Abbildung 17 in Anhang B, Seite 63, zeigt einen Ausschnitt der tabellarischen Auflistung aller Sätze für eine Datei. Die Sätze sind über verschiedene Dateien manuell anhand des Textes, der Operationen und der Metadaten ausgewertet worden.

Folgende Kategorien sind daraus entstanden (die Priorität wird nach der Tabelle erläutert):

Kategorie	Anzeigen	Priorität	Beschreibung
Infoblock	Nein	100	Änderungen innerhalb eines Infoblocks sind Interna von Microsoft und haben keine Relevanz. *Technisch:* Die Zusatzinformation "IsInfoblock" wird geprüft.
HeadingLine	Nein	100	Änderungen an Überschriften haben keine direkte Relevanz. *Technisch:* Die Zusatzinformation "IsHeading" wird geprüft.
MonikerLine	Nein	100	dito. für Moniker *Technisch:* Die Zusatzinformation "IsMoniker" wird geprüft.

Kategorie	Anzeigen	Priorität	Beschreibung
Beispielcode (Example)	Nein	100	Änderungen innerhalb von Beispielen werden nicht ausgegeben, da sie gemäss Stichproben selten notwendige Informationen aufweisen. *Technisch:* In den Überschriften 2 und 3 (Zusatzinformation Heading2 bzw. Heading3) wird geprüft, ob das Wort "Example" verkommt.
Siehe auch (See also)	Nein	100	Der Schlusspunkt einer Hilfeseite sind oft Verweise zu anderen Seiten. Änderungen in diesem Bereich werden als nicht relevant betrachtet. *Technisch:* In den Überschriften 2 und 3 (Zusatzinformation Heading2 bzw. Heading3) wird geprüft, ob das Wort "See also" verkommt.
Click a product	Nein	100	Dies ist eine Spezialität in den Hilfeseiten von Microsoft. Eine Hilfeseite kann für mehrere Angebote (monikers) geschrieben werden. Deshalb fügt Microsoft neu einen Abschnitt ein, damit die Benutzer zwischen den Angeboten wechseln können (die Abbildung 5, Seite 5, zeigt einen solchen Abschnitt). Diese Benutzerführung weist keine relevanten Informationen auf. *Technisch:* In der Überschrift 2 (Zusatzinformation Heading2) wird geprüft, ob das Wort "Click a product" verkommt.
Syntax	Ja	200	Syntax-Änderungen sind wichtige Informationen für Entwickler und Administratoren. *Technisch:* In den Überschriften 2 und 3 (Zusatzinformation Heading2 bzw. Heading3) wird geprüft, ob das Wort "Syntax" verkommt. Wenn ja, wird der Satz als Syntax-relevant gekennzeichnet. Ausnahme ist, wenn die Syntax nur einen (Code) Kommentar aufweist.
Differenzoperation Replace	Evtl.	200	Die Stichproben zeigen, dass Änderungen innerhalb eines Satzes in der überwiegenden Zahl keine relevanten Informationen vermitteln. Meistens sind es stilistische Verbesserungen, Neuformulierungen, Vereinheitlichungen aufgrund von (internen) Richtlinien oder Rechtschreibkorrekturen. Deshalb werden Replace Differenzoperationen ausgeschlossen. Es gibt zwei Ausnahmen; wenn die höher priorisierte Kategorie "Syntax" zutrifft, oder wenn der vorhergehende Satz angezeigt wird, z.B. eine Insert Differenzoperation. Die zweite Ausnahme wird im Anschluss an diese Tabelle erläutert.
Gleicher Text	Nein	300	Bei einer Replace-Differenzoperation gibt es je einen Satz links und rechts. Diese beiden Sätze werden zuerst bereinigt. Wenn sie exakt gleich sind, werden sie ausgeschlossen. *Technisch:* Die Bereinigung umfasst die gleichen komplexen Cleansing-Funktionen, die bereits bei den Änderungsblöcken im Abschnitt "Differenzen ermitteln" angewendet wurden (z.B. Same Lemma, Auxiliary Verbs).

Kategorie	Anzeigen	Priorität	Beschreibung
Gleicher Satz an anderer Stelle	Nein	250	Sämtliche Sätze werden überprüft, ob es zu einem eingefügten Satz einen entsprechenden gelöschten Satz gibt im gleichen oder in einem anderen Änderungsblock (und vice versa, ob ein gelöschter Satz einen eingefügten aufweist). Dadurch kann das Verschieben von Textblöcken angegangen werden. *Technisch:* Die Sätze werden zuerst bereinigt, indem alle Nicht-Wortzeichen[32] entfernt werden. Voraussetzung für eine Übereinstimmung ist, ▪ dass die Sätze den gleichen Moniker (also das gleiche Angebot) aufweisen ▪ dass es nur einzelne vollständige Sätze sind (via Zusatzinformation "IsSentence" = True) Wenn es eine Übereinstimmung gibt, werden beide Sätze mit Anzeigen = Nein gekennzeichnet.
Include Link	Nein	300	In den Hilfeseiten gibt es Textstellen, die Microsoft nun in externe Dateien auslagert (um den gleichen Text in verschiedenen Hilfeseiten wiederverwenden zu können), z.B. Produktnamen. Wenn der bisherige ausgeschriebene Text und der Inhalt der inkludierten Datei übereinstimmen, dann ist es keine relevante Differenz. *Technisch:* Das Programm lädt zu Beginn alle Include-Dateien[33] mit ihrem Text in einen Dictionary. Die Url bildet den Key. Im Satz wird nach Include-Links gesucht. Wird ein Link gefunden, dann wird seine Url im Dictionary gesucht. Wenn gefunden, wird der Text anstelle des Links eingefügt, damit beide Sätze verglichen werden können.
Differenzoperation Insert	Ja	400	Eingefügte Sätze werden angezeigt, sofern keine andere Kategorie dies negiert.
Differenzoperation Delete	Nein	400	Gelöschte Sätze werden nicht angezeigt.

Tabelle 31: Kategorisieren von Sätzen

Mehrere Kategorien können auf eine Differenzoperation (und deren Satz) zutreffen. Deshalb wird eine Priorisierung benötigt, welche Kategorie Vorrang hat.

Die höchste Priorität (Priorität ist aufsteigend sortiert) haben sechs Kategorien, u.a. Infoblock, IsHeading, IsMoniker. Dies sind irrelevante Änderungen, die nie einen Mehrwert liefern. Die Kategorie Syntax hat die zweithöchste Priorität, da Syntaxänderungen sehr hohe Relevanz haben.

Die Anwendung der Priorität lässt sich gut am Beispiel "Differenzoperation Insert" zeigen. Wenn ein Satz eingefügt wird, dann ist er normalerweise für den Benutzer relevant. Wird er aber innerhalb eines Infoblocks hinzugefügt, dann handelt es sich um Microsoft-interne Informationen. "Infoblock" hat höhere Priorität als "Differenzoperation Insert", der Satz wird nicht angezeigt.

Die Kategorie "Differenzoperation Replace" ist ein Spezialfall. Bei diesen Sätzen wird sichergestellt, dass der Textfluss bestehen bleibt, wenn der vorherige und nachfolgende Satz angezeigt werden. Das Beispiel in Tabelle 32 zeigt, wie innerhalb eines Änderungsblocks mehrere verschiedene Differenzoperationen vorkommen können:

[32] Regular Expression Pattern: \W

[33] Es werden nur Include-Dateien berücksichtigt, die genau eine Zeile aufweisen. Es gibt auch Include-Dateien, die kleinere oder grössere Textabschnitte aufweisen. Diese umfassen mehrere Sätze und ein Vergleich würde damit erschwert.

Operation	Differenzoperation	Links	Rechts
Replace	Insert		Please note that in order for the inheritance to work, the three individual tuning options FORCE_LAST_GOOD_PLAN, CREATE_INDEX and DROP_INDEX need to be set to DEFAULT on databases.
Replace	Insert		CUSTOM Using the CUSTOM value, you'll need to custom-configure each of the Automatic Tuning options available on databases.
Replace	Replace	Enables or disables FORCE_LAST_GOOD_PLAN automatic tuning option.	Enables or disables automatic index management CREATE_INDEX option of Automatic tuning.
Replace	Insert		CREATE_INDEX = { DEFAULT | ON | OFF } DEFAULT Inherits default settings from the server.

Tabelle 32: Kategorisieren von Sätzen – Beispiel für Differenzoperation Replace

Der dritte Satz würde als Replace-Differenzoperation eliminiert. Da er jedoch innerhalb von mehreren Insert-Differenzoperationen rangiert, würde der Textfluss der vier Sätze gestört. Deshalb ist die Kategorie "Replace" so umgesetzt, dass eine Replace-Differenzoperation ausgegeben wird, wenn die vorhergehende Differenzoperation angezeigt wird (Attribut "Anzeigen" = Ja).

Die Daten sind nun vorbereitet und werden in eine Datei gespeichert, damit sie für die Benutzer aufbereitet werden können.

5.1.3 Sätze zwischenspeichern

Pro Hilfeseite werden alle Sätze mit den Zusatzinformationen und dem Attribut "Anzeigen" in einer Datei zwischengespeichert. Die gespeicherten Dateien können im nächsten Schritt verarbeitet werden, um den Benutzern die Differenzen anzuzeigen.

Dateien sind eine einfache Möglichkeit, dass einerseits die letzte Phase mit dem Anzeigen der Daten unabhängig entwickelt werden kann und dass andererseits mehrere Ausgangsdateien zu einer Anzeigeseite kombiniert werden können.

Als Speicherformat wird JSON verwendet. JSON ist ein leichtgewichtiges Format, das von vielen Programmiersprachen verarbeitet werden kann. Auch Python hat eine Library namens *json*, mit der JSON genutzt werden kann.

5.2 Zwischenresultat

Am Ende dieses Kapitels stehen die relevanten Änderungen – in Form von Sätzen – bereit. Zusatzinformationen helfen, das Resultat zu evaluieren und zu verstehen.

Es gibt auch hier noch Verbesserungspotenzial im Sinne der 80/20-Regel. Das Extrahieren der Sätze ist noch nicht vollkommen, z.B. wenn die Library *spaCy* die Satzgrenzen nicht erkennen kann. Auch die Suche nach den Originalsätzen weist noch Ungenauigkeiten auf, entweder weil der Satz gar nicht gefunden wird oder nicht mit den richtigen Satzgrenzen.

Das Resultat kann aber benutzt werden, um die Anzeige für die Benutzer in der letzten Phase umzusetzen.

6 Verdichtete Anzeige

Einer der wichtigsten Punkte eines solchen Projektes ist es, die ermittelten relevanten Differenzen in einer benutzerfreundlichen Form aufzubereiten.

Heutige Benutzeroberflächen müssen attraktiv sein, einfach bedienbar und die Benutzer schnell zum Ziel führen.

6.1 Vorgehen

Die generierten Daten aus den vorherigen Schritten sind statisch: sie ändern sich nicht mehr. Deshalb kann eine statische Webseite mit einigen dynamischen Elementen verwendet werden. Solche Webseiten können einfach umgesetzt werden und lassen sich in einem Browser auf Computern, Tablets und sogar Smartphones anzeigen.

Die Webseite für die Datenausgabe wird über ein Template generiert. Ein solches Template enthält Informationen über die Bestandteile der Oberfläche (Header, Footer, Menu, Hauptbereich), es inkludiert Anweisungen für die Formatierung und Farbgebung und definiert spezifische Platzhalter, wo die Daten eingefügt werden.

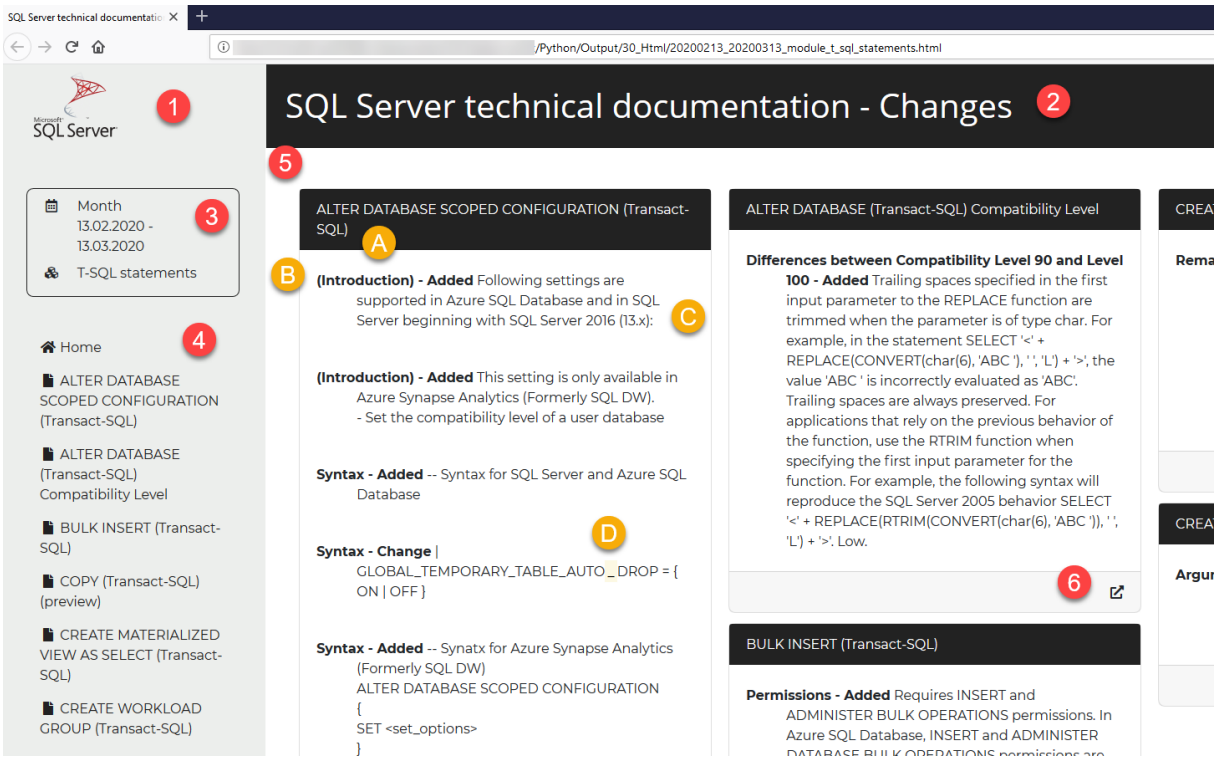


Abbildung 14: Beispiel - Statische Webseite

Abbildung 14 zeigt zuoberst den Header mit einem Logo (1) und dem Titel (2). Auf der linken Seite ist der Kontext (3): es werden die Differenzen für den Zeitraum "Month" und das Modul "T-SQL statements" aus der Online-Hilfe angezeigt. Darunter ist die Navigation (4), die alle betroffenen Dateien als Links auflistet. Im Hauptbereich (5) ist pro Datei eine Karte aufbereitet mit den Differenzen. Am Ende einer Karte ist ein Symbol (6) aufgeführt, mit der die Benutzer die zugehörige Original-Hilfeseite in einem neuen Fenster anzeigen können.

Jede Karte hat einen Titel [A] (entspricht Überschrift 1 bzw. Heading1). Für die Anzeige der Differenzen gibt es mehrere Möglichkeiten. In der ersten Version werden die relevanten, anzuzeigenden Sätze eines Änderungsblocks zusammengefasst. Einleitend zu einem Änderungsblock wird die zugehörige Überschrift 2 [B] fettgedruckt dargestellt, im Beispiel "Syntax" weiter unten oder "(Introduction)", wenn keine Überschrift vorhanden ist. Danach folgen die geänderten bzw. hinzugefügten Sätze [C]. Änderungen werden als "Change" aufgelistet. Wenn neuer Text hinzugefügt[34] wurde, ist dies farblich markiert [D].

Wenn eine Datei sehr viele Änderungen aufweist, kann die Karte sehr lang werden. Dies wird in dieser ersten Umsetzung in Kauf genommen. Die beste Ausgabeform muss mit Benutzerfeedback in Form von Prototyping im Anschluss an die Masterarbeit erarbeitet werden.

Die Webseite passt sich automatisch an verschiedene Auflösungen an. Wenn die Ausgabefläche schmaler wird (z.B. auf einem Smartphone), dann werden Titel, Navigation und Karten untereinander angezeigt. Dieses Vorgehen wird Responsive Webdesign genannt (Wikipedia contributors, Responsive web design, 2020).

6.2 Technische Umsetzung

Für das Erstellen der Webseite sind drei Bestandteile wichtig:

- Daten
- Webtechnologie(n)
- Template

6.2.1 Daten

Die Daten werden aus den JSON-Dateien eingelesen, die im vorherigen Kapitel erstellt wurden. Aus den JSON-Strukturen werden in Python Listen generiert, weil die nachfolgende Template-Engine v.a. mit solchen Listen die Daten in der Webseite darstellt.

Die Daten enthalten Informationen zur Datei (Titel, Dateiname, Modul) und zum Zeitraum. Die Sätze sind zusammengefasst nach Änderungsblock und Überschriften. Diese Gruppierung (v.a. nach Überschriften) ist wichtig, weil bei einem langen Änderungsblock mehrere Überschriften vorkommen können. Die Überschrift leitet den Differenztext ein.

6.2.2 Webtechnologie(n)

Die Webentwicklung hat sich in den letzten Jahren stark geändert und dieser Trend setzt sich fort.

Die zwei Hauptpfeiler HTML und CSS bilden seit Jahren eine stabile Basis. HTML (Hypertext Markup Language) ist die Auszeichnungssprache, die beschreibt, *was* auf der Webseite angezeigt wird. CSS (Cascading Style Sheets) ist die Formatierungssprache, die beschreibt, *wie* und *wo* etwas auf der Webseite dargestellt wird.

Daneben gibt es eine Frameworklandschaft, die sich kontinuierlich verändert. Ein Framework kann innerhalb kurzer Zeit veraltet sein und durch ein neues abgelöst werden. Aktuell ist *Bootstrap*[35] ein bekanntes und häufig eingesetztes Framework, um konsistente, responsive

[34] Aktuell wird nur eingefügter Text markiert. Änderungen sind noch nicht umgesetzt. Eine Möglichkeit, Änderungen anzuzeigen, ist mit einem Vorher/Nachher-Vergleich, z.B. <termination> ::= { ROLLBACK AFTER ~~integer~~ number [SECONDS]

[35] Bootstrap wurde von Twitter entwickelt (Wikipedia contributors, Bootstrap (front-end framework), 2020): https://getbootstrap.com

Webseiten zu bauen. Mit wenig Aufwand lassen sich Seitenaufbau, Header, Footer, die Navigationsbar und die Karten festlegen.

Bootstrap ist aber auch erweiterbar, denn das Framework liefert nur ein Farbschema mit. Mit dem Einsatz eines "Themes" können Farben und Schriften anders gestaltet werden. Themes können selbst erstellt werden, oder im Internet gibt es sowohl kostenlose als auch kostenpflichtige Angebote. Für diese Masterarbeit wird das kostenlose Theme *Monotone* von TopHat[36] verwendet, um ein stilvolles, konsistentes Design zu erreichen.

Eine weitere Library kommt für die Anzeige von Icons und Elementen zum Zuge. Auf der linken Seite werden u.a. ein Kalendersymbol und ein Dokumentsymbol in der Navigation angezeigt. Diese Symbole werden nicht als Grafikdateien, sondern in Form einer Schriftart aufbereitet. Die Library *Font Awesome*[37] liefert unzählige Symbole; bei der kostenlose Variante mit eingeschränktem Umfang. Der grosse Vorteil als Font ist, dass die Grösse der Symbole mit wenig Aufwand an die Schriftgrösse des Textes angepasst werden kann.

Die Anzeige kann man mit entsprechendem Wissen in HTML und CSS selbst anpassen und erweitern. Z.B. sind die leicht abgerundeten Kartenecken mit einem CSS-Style hinzugefügt worden.

6.2.3 Template

Die Webseiten selbst mit HTML zu programmieren ist (zu) aufwändig. Mit Templates können Webseiten datengetrieben schnell und erfolgreich erstellt werden.

Die Python Library *Flask* bietet eine Template-Engine. Sie benötigt zwei Bestandteile: ein Template und Daten.

Das Template umfasst ein Skelett mit allen Anweisungen für die Webseite mit HTML, CSS, *Bootstrap*, *Font Awesome* und dem Theme *Monotone*. In dieses Skelett werden Platzhalter integriert, die später mit den Daten ersetzt werden.

Neben Platzhaltern sind auch (verschachtelte) Schlaufen erlaubt, um eine Liste auszugeben oder eine Navigationsbar zu erstellen. Im Template für die Webseite gibt es eine Schlaufe für die Navigationsbar auf der linken Seite (Auflistung der Dateien). Im Hauptbereich werden die Karten mit einer verschachtelten Schlaufe erstellt (die innere Schlaufe erzeugt die Differenzblöcke).

Mit wenigen Zeilen Code wird aus Template und Daten eine statische HTML-Datei generiert, die gespeichert werden kann. Man kann sie lokal im Browser anschauen oder auf einem Webserver veröffentlichen.

```
1  {% for document in items.Documents %}
2    <li class="nav-item">
3      <a class="nav-link" href="#{{ document.Link }}"><i class="fas fa-file fa-fw"></i>
4        {{ document.Title|safe }}
5      </a>
6    </li>
7  {% endfor %}
8  </ul>
```

Abbildung 15: Ausschnitt Flask Template

Abbildung 15 zeigt einen Ausschnitt aus dem Template. Die geschweiften Klammern definieren entweder die Schlaufe (Zeile 1 und 7) oder einen Daten-Platzhalter `{{ document.Title }}`. Die anderen Elemente sind HTML-Tags sowie CSS-Anweisungen von *Bootstrap* (z.B. "nav-item") und *Font Awesome* (z.B. "fas fa-file fa-fw").

[36] https://themesguide.github.io/top-hat/dist/monotone
[37] https://fontawesome.com

6.3 Abgrenzung

Die Webseite ist als Proof-of-Concept zu betrachten. Für eine Veröffentlichung im Internet fehlen einige Punkte, die im Anschluss an die Masterarbeit umgesetzt werden können:

- Es gibt weder eine Einstiegsseite, noch eine hierarchische Struktur mit allen generierten Zeiträumen und den Modulen der Online-Hilfe.
- Die Datumsausgabe ist fix auf Deutsch und passt sich nicht an die Browsersprache an.
- Die Navigation ist noch rudimentär umgesetzt.

6.4 Vergleich mehrerer Zeiträume

Die Ausgabe kann nun über verschiedene Zeitintervalle erstellt und verglichen werden.

Zeitintervall	Anzahl Tage (gerundet)	Angezeigte Dateien	Angezeigte Änderungen[38]
Woche	7	0	0
Monat	30	6	11
Quartal	90	16	36
Halbjahr	180	40	210
Jahr	360	93	545

Tabelle 33: Vergleich von Zeitintervallen

Die Resultate für die fünf vorbereiteten Zeitintervalle in Tabelle 33 und Abbildung 16 zeigen ein nahezu lineares Wachstum. Innerhalb eines Jahres ändert über ein Viertel aller Dateien (93 von 359), und es ergibt sich eine hohe Anzahl an Änderungen (545).

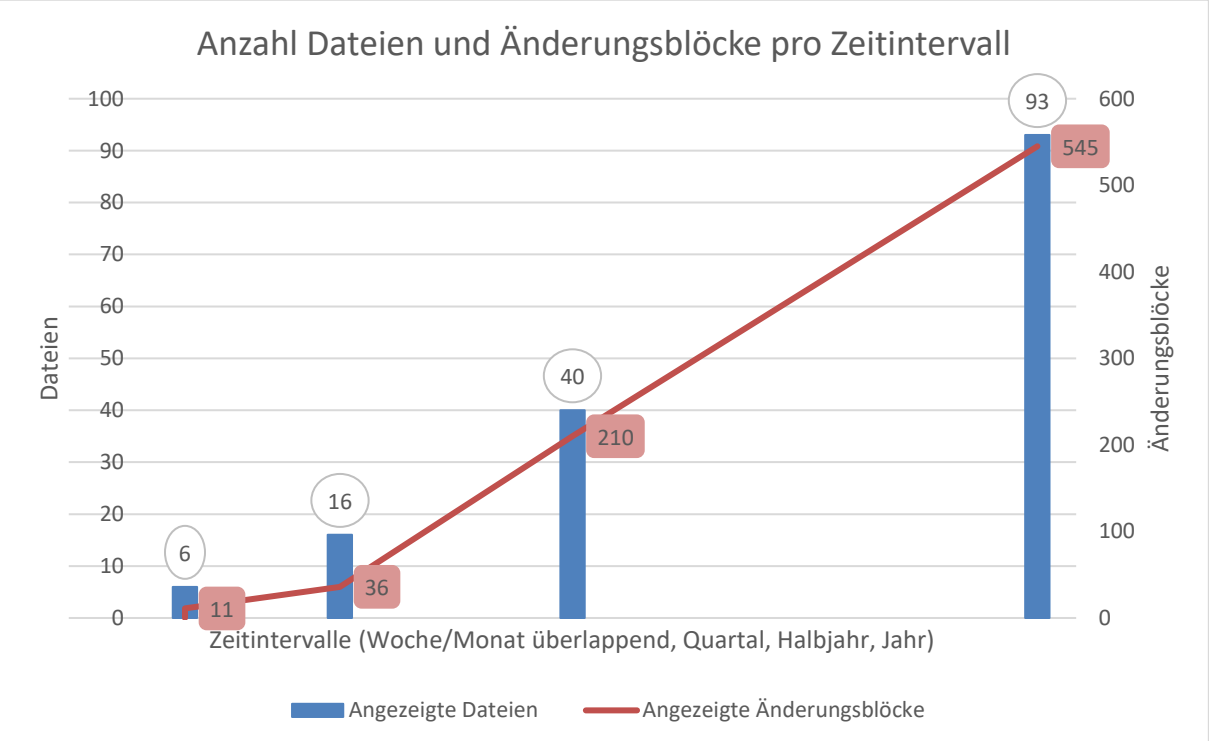


Abbildung 16: Diagramm mit Anzahl Dateien/Änderungsblöcke pro Zeitintervall (an Zeit angeglichen)

[38] Die Änderungen betreffen Änderungsblöcken, zusätzlich gruppiert nach Überschrift.

Wenn man in Betracht zieht, dass die gesamte Online-Hilfe über 23'000 Dateien aufweist, dann kommen für die grossen Intervalle Quartal, Halbjahr und Jahr zu viele Änderungen zusammen, um sie den Benutzern übersichtlich und in geeigneter Form darstellen zu können.

Sinnvoller ist ein wöchentlicher oder monatlicher Rhythmus, um die Webseite nicht zu überfrachten. Beide Intervalle haben den zusätzlichen Vorteil, dass die Änderungen weniger komplex sind. Denn viele kleine Änderungen über einen längeren Zeitraum sind in der Summe schwieriger aufzubereiten und zu bereinigen.

Grössere Intervalle könnten in Frage kommen, um historische Änderungen über ein längeren Zeitraum aufzuzeigen, z.B. bei Hilfeseiten, die sich selten ändern. Wie oben beschrieben nimmt aber die Ungenauigkeit bei den Differenzen stark zu. Eine Alternative ist, pro Datei die wöchentlichen oder monatlichen Änderungen als Karten aufzulisten.

7 Diskussion und Ausblick

7.1 Diskussion

Natural Language Processing (NLP) ist ein spannendes Gebiet und liefert interessante Konzepte, auch für diese Masterarbeit. POS-Tagging und Lemma fanden erfolgreich Anwendung beim Bereinigen von Differenzen. Andere Werkzeuge wie Thesaurus und Word Similarity konnten erst mit den Hilfetexten umgehen, als sie manuell für die Aufgabe erweitert wurden.

Alle umgesetzten Algorithmen sind regelbasiert. Es werden nur Differenzen bereinigt, für die eine Regel existiert, alle anderen Differenzen fallen durch das Raster. In der heutigen Zeit von Machine Learning und Deep Learning scheint dies nicht mehr zeitgemäss. Trotzdem liefert die regelbasierte Vorgehensweise brauchbare Resultate. Unklar ist, ob sich die Regeln auf andere Themen, z.B. auf andere Microsoft-Produkte oder IT-fremde Gebiete übertragen lassen.

Kritisch zu hinterfragen ist, ob es richtig war, dass die Änderungsblöcke auf Kleinschreibung umgestellt wurden und dass die Cleansing-Funktionen den Text verändert haben. Die erste Phase mit der Bereinigung von Änderungsblöcken wurde dadurch vereinfacht, aber die Suche nach Originalsätzen wurde sehr komplex und fehleranfällig. Das Projekt war zum Zeitpunkt dieser Fragestellung zu weit fortgeschritten. Eine Umstellung der Vorgehensweise hätte das Projekt gefährdet, weil alle Funktionen und das Zusammenspiel der Funktionen hätten überarbeitet werden müssen. Im Ausblick wird eine alternative Vorgehensweise aufgezeigt.

Auch die Library *difflib* sollte nochmals auf den Prüfstand kommen. Bei einigen Dateien und Zeiträumen hatte die Library grosse Probleme, die Änderungsblöcke bestmöglich zu ermitteln, bzw. sie gab Differenzen aus, die eigentlich nur Whitespace-Anpassungen sind. Grafische Diff-Tools mit anderen Algorithmen lieferten bei einem Vergleich bessere Resultate. Im Python-Umfeld gibt es wenig Alternativen, deshalb ist die Suche auf andere Programmiersprachen, z.B. .NET oder Java, auszuweiten. Allenfalls kommen auch grafische Diff-Tools in Frage, wenn sie Kommandozeile und die Diff-Ausgabe in eine Datei unterstützen.

Beim Ausarbeiten, welche Differenzen für die Benutzer relevant sind, war es überraschend, dass v.a. eingefügte Sätze in Betracht zu ziehen sind. Kleine Änderungen (Zeichen, Wörter) innerhalb eines Satzes sind fast ausnahmslos unbedeutend. Ausnahmen gibt es, wenn die Syntax betroffen ist. Verblüffend war aber auch die schiere Menge an Änderungen, die sich in einem halben oder ganzen Jahr ergeben haben. Ein Grund ist, dass Microsoft ein neues Angebot in der Cloud (SQL Server managed instance) aufgeschaltet hat. Ein weiterer ist, dass ein Grossteil der Seiten ständig verbessert und mit weiteren Erläuterungen angereichet wird. Diese Menge an Informationen stellt aber eine grosse Herausforderung dar, wie die relevanten Informationen in geeigneter Form dargestellt werden. Der erste Prototyp ist v.a. für kleine Zeitintervalle geeignet, weil die Änderungen noch überschaubar sind. Der Ausblick geht darauf nochmals ein.

Bei der Erstellung der Webseite zeigt sich, dass man mit den aktuellen Frameworks und Themes, wie *Bootstrap*, mit überschaubarem Aufwand sehr gute Ergebnisse erzielen kann, ohne Experte in Webtechnologien sein zu müssen. Die erste Beurteilung des Webseiteninhalt ergab eine gewisse Ernüchterung, weil einzelne Sätze den Gesamtkontext nicht klar aufzeigten. Im Ausblick wird die Integration des Kontexts besprochen.

In Bezug auf das Python Projekt haben die komplexen Cleansing-Funktionen nicht nur den grössten Aufwand verursacht und sondern haben auch den grössten Anteil am Programmcode. Im Verlaufe des Projekts sind immer wieder neue Herausforderungen hinzugekommen: sei es, dass

innerhalb von Dateien mehrere Angebote (monikers) beschrieben sind; sei es, dass die Library *difflib* bei sehr komplexen Dateiänderungen nur noch undifferenziert grosse Blöcke ausgibt. Die Vorgehensweise musste dadurch mehrmals angepasst werden. Auch die Projektstruktur musste mit dem Funktionswachstum schritthalten; Unit Tests wurden notwendig, um sicherzustellen, dass Codeänderungen funktionsfähig bleiben.

Insgesamt ist ein Projekt entstanden, das die Basis für eine erfolgreiche Ausgabe von Differenzen für interessierte Benutzer bildet. Anpassungen und Verbesserungen sind noch notwendig, damit eine grosse Benutzerakzeptanz erreicht werden kann.

7.2 Ausblick

Mehrere Punkte sind im Anschluss an die Masterarbeit notwendig, um mit der Webseite produktiv gehen zu können.

Die *Qualität* der Ausgabetexte kann verbessert werden. Die Originalsätze werden noch nicht richtig erkannt. Es ist zu prüfen, ob sich das Resultat bessert, wenn nicht die ganzen Änderungsblöcke bereinigt werden, sondern zuerst die Sätze extrahiert und auf Differenzen geprüft werden. Jedoch müssen Abschnitte mit Syntax und Codebeispielen sorgfältig geprüft werden, weil sie keine eigentliche Sätze aufweisen. Allgemein sollten die Satzgrenzen aber besser gefunden werden, wenn die Gross-/Kleinschreibung belassen wird. Weiter sind die Libraries *difflib* und *spaCy* nicht immer in der Lage, Überschriften und Abschnittstexte auseinanderzuhalten. Hier können gezielte Bereinigungen den Inhalt aufwerten.

Die *Darstellung* der Differenzen ist zu optimieren. Es gibt viele Dateien mit wenig Änderungen und einige Wenige, die sehr viele Anpassungen aufweisen. Die grossen Dateien nehmen viel Platz in Anspruch und dadurch wird die Anzeige mit den Karten unausgewogen. Das gesamte UI ist mit Design Thinking Methoden zu überarbeiten, indem Prototypen mit Personen aus der Zielgruppe erprobt werden. Dabei sind auch die anzuzeigenden Texte nochmals zu überdenken. Gerade bei Differenzen mit genau einem Satz ist der Gesamtkontext nicht oder nicht sofort klar. Hier ist bei den Prototypen zu prüfen, ob die Benutzer bei Bedarf vor- und nachgehende Sätze einblenden können, um den Kontext zu verstehen.

Eine *hierarchische Seitenstruktur* wird notwendig, sobald das Projekt auf alle 23'000 Hilfeseiten ausgeweitet wird und regelmässig Differenzen für ein gewähltes Intervall aufbereitet werden. Die Struktur sollte die Informationen nach Modulen innerhalb des SQL Server aufteilen und eine Benutzerführung durch die Intervalle erlauben. Alternative Ansichten könnten den Benutzern zusätzliche Wege geben, z.B. wenn pro Datei alle monatlichen, relevanten Differenzen aufgelistet werden.

Wichtig ist auch, dass *neue Dateien* erkannt und als Änderungsinformation aufgenommen werden. Neue Dateien können für die Benutzer sehr relevant sein, weil damit ein neuer Befehl oder ein neues Konzept im SQL Server beschrieben wird. Falls in einem Ordner neue und gelöschte Dateien existieren, dann muss sorgfältig geprüft werden, ob Dateien umbenennt wurden (statt sie als neu anzupreisen).

Um die Seite betreiben zu können, ist der komplette Ablauf zu *automatisieren*. Das Programm muss die Seiten von Github automatisiert herunterladen und die Intervalle vorzubereiten. Dann können die Differenzen ermittelt und für die Benutzer aufbereitet werden.

Literaturverzeichnis

AI, E. (kein Datum). *Word Vectors and Semantic Similarity*. Abgerufen am 7. April 2020 von spaCy: https://spacy.io/usage/vectors-similarity

Allen, D., & White, S. (kein Datum). *Asciidoctor.* Von Asciidoctor Writer's Guide: https://asciidoctor.org/docs/asciidoc-writers-guide/#admonitions abgerufen

Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing (Third Edition Draft).

Kochmar, E. (2019). *Essential Natural Language Processing (MEAP Version 2).* Manning Publications.

Merejkowsky, D. (11. März 2020). *pyenchant/pyenchant: spellchecking library for python*. Von Github: https://github.com/pyenchant/pyenchant abgerufen

National Institute of Standards and Technology. (4. März 2018). *usnistgov / SCTK*. Von NIST SCLITE Scoring Package Version 1.5: https://github.com/usnistgov/SCTK/blob/master/doc/sclite.htm abgerufen

present 8x8, I. (22. März 2020). *jitsi/jiwer: Evaluate your speech-to-text system with simualarity measures such as word error rate (WER)*. (present 8x8, Inc.) Von Github: https://github.com/jitsi/jiwer/ abgerufen

Python Standard Library (difflib). (12. November 2019). *difflib - Helpers for contributing deltas*. Von The Python Standard Library: https://docs.python.org/3/library/difflib.html abgerufen

Sarkar, D. (2019). *Text Analytics with Python (Second Edition).* Apress Media LLC.

Universal Dependencies. (n.d.). *Universal Dependencies*. Retrieved from Universal POS tags: https://universaldependencies.org/u/pos/all.html#det-determiner

Wikipedia contributors. (30. Juni 2020). *Bootstrap (front-end framework)*. (Wikipedia, The Free Encyclopedia) Von https://en.wikipedia.org/w/index.php?title=Bootstrap_(front-end_framework)&oldid=963875235 abgerufen

Wikipedia contributors. (5. März 2020). *Edit distance*. (Wikipedia, The Free Encyclopedia) Von Wikipedia: https://en.wikipedia.org/wiki/Edit_distance abgerufen

Wikipedia contributors. (2020, 06 26). *Responsive web design*, 964531463. (Wikipedia, The Free Encyclopedia.) Retrieved from https://en.wikipedia.org/w/index.php?title=Responsive_web_design&oldid=964531463

Wikipedia contributors. (7. Februar 2020). *Word error rate*. (Wikipedia, The Free Encyclopedia) Von Wikipedia: https://en.wikipedia.org/wiki/Word_error_rate abgerufen

Anhang A – Änderungsblöcke pro Intervall

Die nachfolgenden Tabelle 34 zeigt die gesamte Entwicklung der Änderungen bei jeder betroffenen Datei. Die Änderungsblöcke wurden mit der Python Library *difflib* ermittelt.

Zuerst wird das kleinste Intervall "Woche" aufgelistet, die Daten sind eine Woche alt; danach kommen die weiteren grösseren Intervalle (die Zeit läuft rückwärts von links nach rechts, die Datenbasis wird immer älter). Grüne Farben bedeuten wenig Differenzen, rötliche und rote Differenzen viele Differenzen. Blau weist auf eine aussergewöhnliche hohe Anzahl hin und weisse Zellen zeigen an, dass die Datei erst nach diesem Intervallzeitpunkt erstellt wurde.

Dateiname	2020			2019	
Originalzustand per 13.03.2020	06.03.	13.02.	13.12.	13.09.	13.03.
Intervall rückwärts zur Originalzustand	Woche	Monat	Quartal	Halbjahr	Jahr
add-sensitivity-classification-transact-sql.md	0	0	0	8	9
alter-application-role-transact-sql.md	0	0	0	3	4
alter-authorization-transact-sql.md	0	0	0	3	4
alter-certificate-transact-sql.md	0	0	1	2	17
alter-column-encryption-key-transact-sql.md	0	0	0	9	9
alter-database-scoped-configuration-transact-sql.md	0	8	10	51	52
alter-database-transact-sql-compatibility-level.md	0	1	3	38	36
alter-database-transact-sql-file-and-filegroup-options.md	0	0	1	7	10
alter-database-transact-sql-set-options.md	0	1	406	405	275
alter-database-transact-sql.md	0	0	22	36	82
alter-external-data-source-transact-sql.md	0	0	0	2	19
alter-external-language-transact-sql.md	0	0	0	2	
alter-external-library-transact-sql.md	0	0	0	4	21
alter-fulltext-index-transact-sql.md	0	0	0	11	12
alter-function-transact-sql.md	0	0	0	3	4
alter-index-transact-sql.md	0	0	1	3	74
alter-login-transact-sql.md	0	0	9	18	19
alter-materialized-view-transact-sql.md	0	0	0	2	
alter-procedure-transact-sql.md	0	0	0	2	3
alter-queue-transact-sql.md	0	0	0	7	10
alter-resource-governor-transact-sql.md	0	0	0	1	2
alter-resource-pool-transact-sql.md	0	0	0	5	6
alter-role-transact-sql.md	0	0	1	1	2
alter-schema-transact-sql.md	2	2	2	2	3
alter-security-policy-transact-sql.md	0	0	0	1	2
alter-server-audit-specification-transact-sql.md	0	0	1	1	3
alter-server-audit-transact-sql.md	0	0	0	7	8
alter-server-configuration-transact-sql.md	0	0	0	30	30
alter-table-column-constraint-transact-sql.md	0	0	0	2	3
alter-table-column-definition-transact-sql.md	0	0	0	5	6
alter-table-computed-column-definition-transact-sql.md	0	0	0	2	3
alter-table-index-option-transact-sql.md	0	0	0	12	17
alter-table-table-constraint-transact-sql.md	0	0	0	2	3
alter-table-transact-sql.md	0	0	0	48	102
alter-trigger-transact-sql.md	0	0	0	4	6

Dateiname	2020			2019	
Originalzustand per 13.03.2020	06.03.	13.02.	13.12.	13.09.	13.03.
Intervall rückwärts zur Originalzustand	Woche	Monat	Quartal	Halbjahr	Jahr
alter-user-transact-sql.md	0	0	96	4	5
alter-view-transact-sql.md	0	0	0	1	2
backup-transact-sql.md	0	0	0	9	20
bulk-insert-transact-sql.md	0	2	5	27	27
collations.md	0	0	0	2	5
create-aggregate-transact-sql.md	0	0	0	1	2
create-application-role-transact-sql.md	0	0	0	1	2
copy-into-transact-sql.md	0	1	4		
create-assembly-transact-sql.md	0	0	0	4	5
create-asymmetric-key-transact-sql.md	0	0	0	3	20
create-availability-group-transact-sql.md	0	0	0	1	6
create-broker-priority-transact-sql.md	0	0	0	2	3
create-certificate-transact-sql.md	0	0	19	25	27
create-column-encryption-key-transact-sql.md	0	0	0	14	14
create-column-master-key-transact-sql.md	0	0	0	14	14
create-columnstore-index-transact-sql.md	0	0	0	5	24
create-contract-transact-sql.md	0	0	0	1	2
create-credential-transact-sql.md	0	0	3	7	8
create-database-audit-specification-transact-sql.md	0	0	2	2	3
create-database-scoped-credential-transact-sql.md	0	1	1	13	14
create-database-transact-sql.md	0	0	125	145	141
create-event-session-transact-sql.md	0	0	1	1	7
create-external-data-source-transact-sql.md	0	5	91	91	7
create-external-library-transact-sql.md	0	0	0	2	31
create-external-language-transact-sql.md	0	4	4	7	
create-external-table-transact-sql.md	2	2	41	42	23
create-fulltext-catalog-transact-sql.md	0	0	0	1	2
create-fulltext-index-transact-sql.md	0	0	0	2	12
create-fulltext-stoplist-transact-sql.md	0	0	0	3	4
create-function-sql-data-warehouse.md	0	0	0	2	3
create-function-transact-sql.md	0	2	2	7	9
create-index-transact-sql.md	0	0	2	63	157
create-login-transact-sql.md	0	1	17	25	35
create-master-key-transact-sql.md	0	0	0	1	10
create-procedure-transact-sql.md	0	0	1	15	16
create-materialized-view-as-select-transact-sql.md	0	8	8	10	
create-queue-transact-sql.md	0	0	0	4	3
create-remote-table-as-select-parallel-data-warehouse.md	0	0	0	1	3
create-resource-pool-transact-sql.md	0	0	0	6	38
create-role-transact-sql.md	0	0	0	3	5
create-security-policy-transact-sql.md	0	2	2	2	3
create-server-audit-specification-transact-sql.md	0	0	2	2	4
create-server-audit-transact-sql.md	0	0	0	6	7
create-spatial-index-transact-sql.md	0	0	0	12	90
create-statistics-transact-sql.md	0	0	0	3	5
create-synonym-transact-sql.md	0	0	0	2	5

Dateiname	2020			2019	
Originalzustand per 13.03.2020	06.03.	13.02.	13.12.	13.09.	13.03.
Intervall rückwärts zur Originalzustand	Woche	Monat	Quartal	Halbjahr	Jahr
create-table-as-select-azure-sql-data-warehouse.md	0	0	0	8	10
create-table-azure-sql-data-warehouse.md	0	0	0	11	63
create-table-transact-sql.md	0	2	2	187	122
create-trigger-transact-sql.md	3	3	3	11	13
create-type-transact-sql.md	0	0	0	11	14
create-user-transact-sql.md	0	0	0	14	19
create-view-transact-sql.md	0	0	0	10	11
create-workload-classifier-transact-sql.md	3	6	10	12	13
create-workload-group-transact-sql.md	0	23	30	15	5
delete-transact-sql.md	0	0	5	10	12
deny-availability-group-permissions-transact-sql.md	0	0	1	1	2
deny-database-permissions-transact-sql.md	0	0	0	4	5
deny-database-principal-permissions-transact-sql.md	0	0	1	4	5
deny-search-property-list-permissions-transact-sql.md	0	0	1	1	2
deny-xml-schema-collection-permissions-transact-sql.md	0	0	1	1	2
disable-trigger-transact-sql.md	0	0	1	4	5
drop-column-encryption-key-transact-sql.md	0	0	0	4	4
drop-column-master-key-transact-sql.md	0	0	0	4	4
drop-database-transact-sql.md	0	0	0	3	4
drop-external-library-transact-sql.md	0	0	0	2	9
drop-function-transact-sql.md	0	0	3	3	4
drop-index-transact-sql.md	0	0	0	10	15
drop-service-transact-sql.md	0	0	0	1	2
drop-trigger-transact-sql.md	0	0	0	1	2
drop-workload-classifier-transact-sql.md	0	0	0	5	9
drop-workload-group-transact-sql.md	0	0	11	7	7
enable-trigger-transact-sql.md	0	0	0	5	6
execute-as-transact-sql.md	0	0	1	5	12
get-conversation-group-transact-sql.md	0	0	0	2	5
grant-availability-group-permissions-transact-sql.md	0	0	1	1	2
grant-database-permissions-transact-sql.md	0	0	0	7	11
grant-database-principal-permissions-transact-sql.md	0	0	1	3	4
grant-schema-permissions-transact-sql.md	0	0	0	1	2
grant-search-property-list-permissions-transact-sql.md	0	0	1	1	2
grant-type-permissions-transact-sql.md	0	0	0	5	6
grant-xml-schema-collection-permissions-transact-sql.md	0	0	1	1	2
insert-transact-sql.md	0	0	0	13	17
merge-transact-sql.md	0	0	0	1	59
permissions-grant-deny-revoke-azure-sql-… house.md	0	0	1	2	3
restore-statements-arguments-transact-sql.md	0	0	0	2	3
restore-statements-for-restoring-… -sql.md	0	0	1	1	2
pick-a-product-template.md	0	0	5		
restore-statements-headeronly-transact-sql.md	0	0	0	1	4
restore-statements-transact-sql.md	0	0	0	33	34
revert-transact-sql.md	0	0	0	3	4
revoke-availability-group-permissions-transact-sql.md	0	0	1	1	2

Dateiname	2020		2019		
Originalzustand per 13.03.2020	06.03.	13.02.	13.12.	13.09.	13.03.
Intervall rückwärts zur Originalzustand	Woche	Monat	Quartal	Halbjahr	Jahr
revoke-database-permissions-transact-sql.md	0	0	0	8	9
revoke-database-principal-permissions-transact-sql.md	0	0	1	8	9
revoke-search-property-list-permissions-transact-sql.md	0	0	1	1	2
revoke-server-principal-permissions-transact-sql.md	0	0	1	1	2
revoke-transact-sql.md	0	0	0	2	3
revoke-type-permissions-transact-sql.md	0	0	0	5	6
revoke-xml-schema-collection-permissions-transact-sql.md	0	0	1	1	2
set-arithabort-transact-sql.md	0	0	1	1	2
set-concat-null-yields-null-transact-sql.md	0	0	0	4	5
set-context-info-transact-sql.md	0	1	1	1	2
set-result-set-caching-transact-sql.md	0	0	0	3	
set-xact-abort-transact-sql.md	0	0	0	5	6
update-statistics-transact-sql.md	0	0	2	6	7
Anzahl geänderte Dateien	**4**	**19**	**58**	**138**	**351**

Tabelle 34: Anzahl Änderungsblöcke bei 359 Dateien über fünf Zeiträume (Gesamtansicht)

Anhang B – Übersicht für Satzkategorisierung

Die Abbildung 17 zeigt einen Ausschnitt der tabellarischen Auflistung aller Sätze in der Datei "alter-database-scoped-configuration-transact-sql.md".

Die Übersicht ist die Grundlage für die Kategorisierung der Sätze, ob sie relevant oder irrelevant für die Anzeige sind. Die Sätze sind über verschiedene Dateien manuell anhand des Textes, der Operationen und der Zuatzinformationen (Metadaten) ausgewertet worden.

alter-database-scoped-configuration-transact-sql.md

Index	No	Sentence Left	Sentence Right	Text Left	Text Right	Diff Operation	Operation	Origin	Category	Omit	Omit Always	Show	Show Syntax	Heading1	Heading2
0	0		monikerRange: "= azuresqldb-current || = azuresqldb-mi-current || >= sql-server-2016 || >= sql-server-linux-2017 ||=azure-sqldw-latest|| = sqlallproducts-allversions"		monikerRange: "= azuresqldb-current || = azuresqldb-mi-current || >= sql-server-2016 || >= sql-server-linux-2017 ||=azure-sqldw-latest|| = sqlallproducts-allversions"	0	0	Native	InfoBlock	True	True	True	False		
1	1	INCLUDE[tsql-applicsto-ss2016-asdb-xxxx-xxx-md.md]	INCLUDE[tsql-applicsto-ss2016-asdb-asdw-xxx-md.md]	include[tsql-applicsto-ss2016-asdb-xxxx-xxx-md.md]	include[tsql-applicsto-ss2016-asdb-asdw-xxx-md.md]	R	R	Native	Unknown	False	False	False	False	ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL)	
1	2	This statement enables several database configuration settings at the individual database level.	This command enables several database configuration settings at the individual database level.	this statement enables several database configuration settings at the individual database level.	this command enables several database configuration settings at the individual database level.	R	R	Native	NoDifference	True	False	False	False	ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL)	
1	3	This statement is available in INCLUDE[sssdsfull-md.md] and in INCLUDE[ssnoversion-md.md] beginning with INCLUDE[sssql15-md.md].		this statement is available in include[sssdsfull-md.md] and in include[ssnoversion-md.md] beginning with include[sssql15-md.md].		R	D	Split	Unknown	True	False	False	False	ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL)	
1	4		Following settings are supported in INCLUDE[sssdsfull-md.md] and in INCLUDE[ssnoversion-md.md] beginning with INCLUDE[sssql15-md.md]:		following settings are supported in include[sssdsfull-md.md] and in include[ssnoversion-md.md] beginning with include[sssql15-md.md]:	R	I	Split	Unknown	False	False	True	False	ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL)	
1	5	Those settings are:		those settings are:		R	D	Native	Unknown	True	False	False	False	ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL)	
2	1		This setting is only available in Azure Synapse Analytics (Formerly SQL DW).		This setting is only available in Azure Synapse Analytics (Formerly SQL DW).	I	I	Native	Unknown	False	False	True	False	ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL)	
2	2		- Set the compatibility level of a user database		- Set the compatibility level of a user database	I	I	Native	Unknown	False	False	True	False	ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL)	
3	1		-- Syntax for SQL Server and Azure SQL Database		-- Syntax for SQL Server and Azure SQL Database	I	I	Native	Unknown	False	False	True	False	ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL)	Syntax
4	1	| GLOBAL_TEMPORARY_TABLE_AUTODROP = { ON | OFF }	| GLOBAL_TEMPORARY_TABLE_AUTO_DROP = { ON | OFF }	| global_temporary_table_autodrop = { on | off }	| global_temporary_table_auto_drop = { on | off }	R	R	Native	Unknown	False	False	False	True	ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL)	Syntax

Abbildung 17: Tabellarische Übersicht für Satzkategorisierung

Anhang C – Python Projekt

Die Masterarbeit basiert auf Python. Das Projekt ist mit der Zeit stark gewachsen. Es war immer wieder notwendig, die Struktur anzupassen und Code umzuschreiben (Refactoring genannt).

Für die Cleansing-Funktionen gibt es Unit Tests (rechts in Abbildung 18). Diese Tests können per Menüpunkt alle auf einmal ausgeführt werden. Sie helfen, Fehler zu erkennen, wenn man in einem Algorithmus eine falsche Änderung macht.

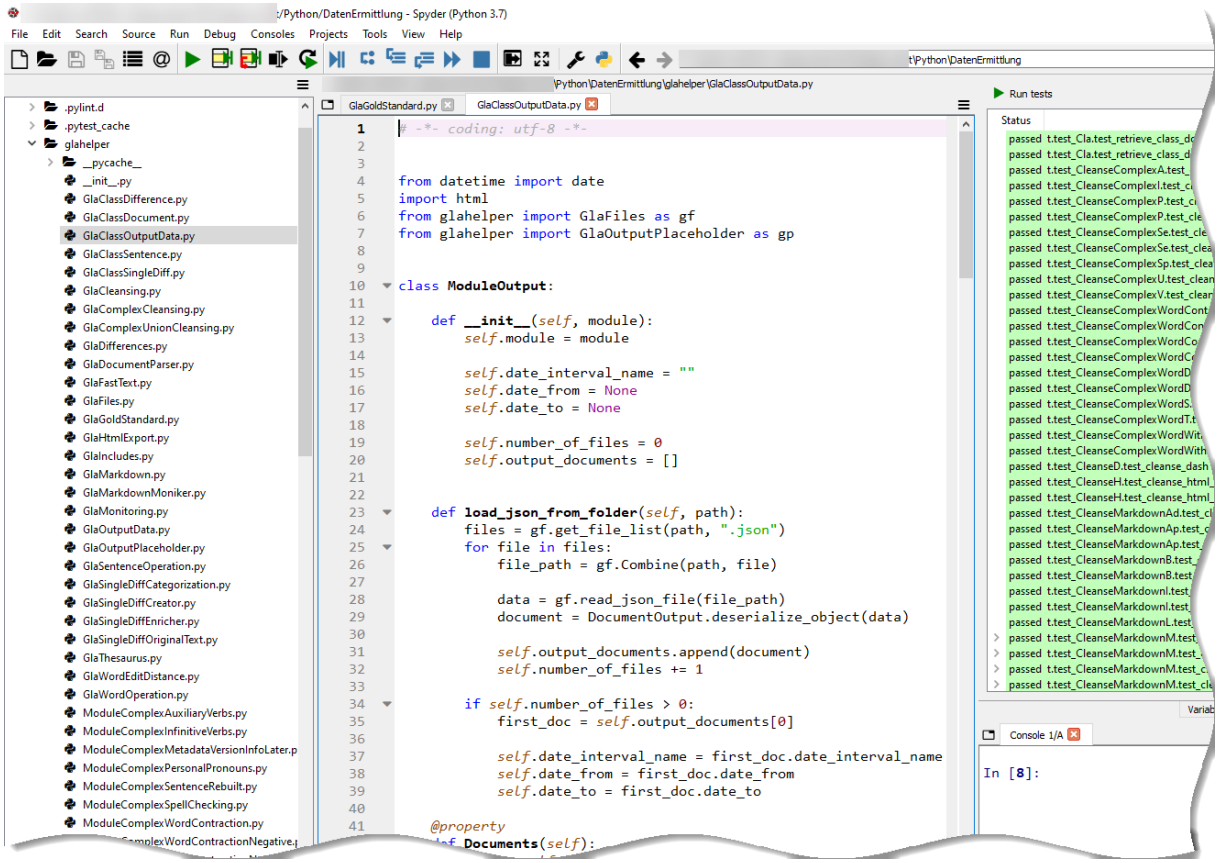


Abbildung 18: Python Projekt - Ausschnitt aus Entwicklungsumgebung Spyder

Einige Zahlen zum Python Projekt:

Einheit	Anzahl	Bemerkung
Dateien mit Programmcode	52	
Dateien mit Unit Tests	43	
Codezeilen	ca. 10'400	davon ca. 9'100 Programmcode davon ca. 1'300 Unit Test Code

Tabelle 35: Zahlen zum Python Projekt

Dieses Python Projekt wird wie die Microsoft Onlinehilfe auf Github verwaltet. Bei Abgabe der Masterarbeit wird der Stand des Projekts mit einem Tag namens "Abgabe" versehen, damit der Zustand zu einem späteren Zeitpunkt wieder hergestellt werden kann.

Anhang D – Hinweise zur Automatisierung

Dieser Abschnitt zeigt kurz auf, wie der Zugriff auf die Originaldaten automatisiert werden kann und wie zwei Zeitabschnitte aufbereitet werden können.

Die Daten sind in einem Repository namens Github gespeichert, und zwar unter der Adresse https://github.com/MicrosoftDocs/sql-docs.git. Die zugrundeliegende SourceCode-Verwaltung heisst Git.

Mit drei Befehlen von Git kann die Datengrundlage automatisiert werden. Die Befehle werden in der Konsole von Git (git-bash.exe)[39] ausgeführt:

Vorgang	Befehle	Automatisierung mit git-bash
Kopie des Repositories lokal erstellen	git clone	`"C:\Program Files\Git\git-bash.exe" -c "cd /c/Github/sql-docs2; **git clone** https://github.com/MicrosoftDocs/sql-docs.git"`
Letzten Stand herunterladen	git pull	`"C:\Program Files\Git\git-bash.exe" -c "cd /c/Github/sql-docs; **git pull origin live**"`
Stand zu einem bestimmten Zeitpunkt[40]	git checkout git rev-list -1	`"C:\Program Files\Git\git-bash.exe" -c "cd /c/Github/sql-docs; **git checkout `git rev-list -1 --before='2020-03-13' live`**"`

Tabelle 36: Git Befehle inkl. Automatisierung

Der erste Befehl kopiert den Stand von Git auf die lokale Festplatte. Die zweite Anweisung ist notwendig, um von Zeit zu Zeit die Änderungen herunterzuladen, damit die Kopie den neuesten Stand aufweist. Der dritte Befehl ist nützlich, um in der lokalen Kopie den Zustand zu einem gewissen Datum anzuzeigen.

Es empfiehlt sich, zwei lokale Kopien des Repositories anzulegen. Um ein Intervall von einer Woche vorzubereiten (z.B. 13.03.2020 – 19.03.2020), kann bei der Kopie 1 der Zustand zum Zeitpunkt 13.03.2020 festgelegt werden (dritter Befehl in Tabelle 36), bei der Kopie 2 der Zustand für 19.03.2020.

Mit dem Python Programm können dann die zwei Kopien für das festgelegte Intervall verglichen werden.

[39] Die Automatisierung mit git-bash ist im folgenden Stackoverflow Thread beschrieben: https://stackoverflow.com/questions/45524126/how-to-automate-git-bash-commands-on-windows-machine

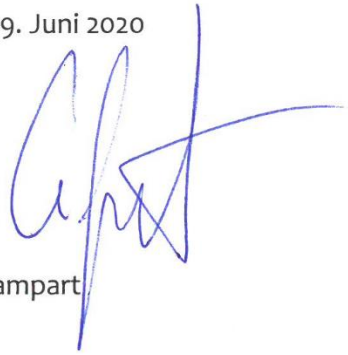
[40] Git erlaubt, den Zustand des Repositories zu einem bestimmen Zeitpunkt anzuzeigen. Der Trick mit git checkout und git rev-list ist auf der Webseite https://www.endpoint.com/blog/2014/05/19/git-checkout-at-specific-date beschrieben.

Selbständigkeitserklärung

Mit der Abgabe dieser Abschlussarbeit versichert der Studierende, dass er die Arbeit selbständig und ohne fremde Hilfe verfasst hat (bei Teamarbeiten gelten die Leistungen der übrigen Teammitglieder nicht als fremde Hilfe).

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

Zürich, 29. Juni 2020



Georg Lampart