Projektabschlussbericht

Dokumentation des Moduls "Projektrealisierung"

im Bereich Wirtschaftsinformatik: Data Science
an der Dualen Hochschule Baden-Württemberg Mannheim

Autoren:	Niklas Koch, Niclas Cramer, Jasmina Pascanovic & Antoine Fuchs
Matrikelnummern:	6699912, 7607733, 5711726 & 3008441
Kurs:	WWI20DSA
Studiengangsleiter:	Prof. Dr.-Ing. habil. Dennis Pfisterer
Dozent:	Michael Lang, Enzo Hilzinger
Abgabedatum:	27.07.2023

Inhaltsverzeichnis

1 Datenverarbeitung und -beschaffung **3**
1.1 Geplante Umsetzung . . . 3
1.2 Durchführung . . . 3
1.3 Herausforderungen . . . 3
1.4 Lessons Learned . . . 4

2 Klassifikation **5**
2.1 Geplante Umsetzung . . . 5
2.2 Durchführung . . . 5
2.3 Herausforderungen . . . 5
2.4 Lessons Learned . . . 6

3 Zusammenfassung **7**
3.1 Projektumfang . . . 7
3.2 Projektziel . . . 7
3.3 Herausforderungen und Lösungsanstätze . . . 7
3.4 Lösungsansätze . . . 8
3.5 Erreichte Ziele . . . 9
3.6 Lessons Learned . . . 9

4 Frontend **11**
4.1 Geplante Umsetzung . . . 11
4.2 Durchführung . . . 11
4.3 Herausforderungen . . . 11
4.4 Lessons Learned . . . 11

5 Zusatz-Features **12**
5.1 Projektumfang . . . 12
5.2 Herausforderungen und Lösungsansätze . . . 12
5.3 Erreichte Ziele . . . 12
5.4 Lessons learned . . . 13

6 Gesamtes Fazit zum Projekt **14**

1 Datenverarbeitung und -beschaffung

Dieses Kapitel beschäftigt sich damit, wie wir vorgegangen sind, um unsere geplanten Anforderungen an die Datenbeschaffung und -verarbeitung zu erfüllen, welche Probleme dabei entstanden und was wir bei der Bearbeitung gelernt haben.

1.1 Geplante Umsetzung

Unser Plan bestand zunächst darin, geeignete Texte für die Klassifikation in vier verschiedene Kategorien zu finden. Dies haben wir mittels einer Recherche erledigt. Wir haben uns für folgende Datensätze entschieden:

- Wissenschaftliche Texte: *https://www.tensorflow.org/datasets/catalog/scientific_papers*
- Literarische Texte: *https://www.kaggle.com/datasets/shubchat/1002-short-stories-from-project-guttenberg?select=stories.csv*
- Nachrichtenartikel: *https://www.kaggle.com/datasets/asad1m9a9h6mood/news-articles*
- Reviews: *https://www.kaggle.com/datasets/kritanjalijain/amazon-reviews?select=train.csv*

1.2 Durchführung

Wir haben die Daten heruntergeladen und zu einem großen Datensatz zusammengefügt. Außerdem haben wir zusätzliche Features aus den Texten extrahiert und den Datensatz auf 700 Datenpunkte begrenzt, sowie in einen Trainings- und Testdatensatz gesplittet. Mehr zur genauen Vorgehensweise in der Dokumentation unter Kapitel 1.

1.3 Herausforderungen

Es war nicht ganz so einfach, geeignete Daten für die Klassifikation zu finden, die öffentlich zugänglich und nutzbar sind. Nach einer mehrtägigen Recherche und mehreren Tests konnten wir jedoch einen akzeptablen Trainingsdatensatz erstellen. Die Herausforderungen hielten sich hier in Grenzen.
Eine weitere Herausforderung war die seltsame Punktuation der wissenschaftlichen Texte. Teilweise waren Punkte oder Kommata dort falsch platziert worden. Wir haben dafür ein weiteres Modell genutzt, welche diese Texte korrigiert hat und dieses auch ans Frontend angebunden, sodass Texte mit fehlerhafter Zeichensetzung weitgehend behoben werden.

1.4 Lessons Learned

Es ist nicht immer leicht, geeignete Daten für ein Projekt oder zum Erreichen seiner Ziele zu finden. Man muss Alternativen erwägen und es kann sein, dass man sehr unterschiedliche Daten findet, die auch in verschiedenen Formaten vorliegen. Man muss seinen Code dementsprechend anpassen. Bei uns lagen die wissenschaftlichen Texte bspw. in einer anderen Form vor, als die restlichen Kategorien. Damit das Modell möglichst unbiased ist, müssen die Daten homogenisiert werden und man muss sicherstellen, dass alle Funktionen, mit den unterschiedlichen Daten, gleich funktionieren.

2 Klassifikation

Dieses Kapitel beschäftigt sich damit, wie wir vorgegangen sind, um unsere geplanten Anforderungen an die Klassifikation von Texten zu erfüllen, welche Probleme dabei entstanden und was wir bei der Bearbeitung gelernt haben.

2.1 Geplante Umsetzung

Wir hatten geplant, verschiedene Arten der Klassifikation zu testen. Sowohl eine Support-Vector-Machine, als auch ein eigenes, simples neuronales Netz, als auch ein Transformer-Modell. Ebenso wollten wir Visualisierungen erstellen, die uns einen genaueren Einblick in die Texte und ihre Unterschiede gewähren.

2.2 Durchführung

Alle drei Klassifikationsansätze wurden durchgeführt. Zuvor jedoch haben wir mit den zusätzlich extrahierten Features diverse Visualisierungen durchgeführt, um einen Überblick über die Texte zu erhalten. Wir konnten bspw. feststellen, dass manche Texte, wie Reviews oder Nachrichtenartikel im Schnitt deutlich kürzer sind, als die anderen beiden Kategorien. Außerdem wurden Unterschiede in der Anzahl an Füllwörtern gefunden, sowie Unterschiede bei der Wortwahl (in literarischen Werken wurde bspw. deutlich häufiger Pronomen wie „I“, „you“ oder „he“ verwendet). Diese Features wurden auch als statistisch signifikant in Bezug auf die jeweilige Klasse des Textes identifiziert. Man kann also davon ausgehen, dass sich die Modelle daran inhärent orientieren. Sowohl die SVM als auch das simple neuronale Netz haben sehr gut performt. Das neuronale Netz wurde letztendlich in unser Frontend integriert. Es gibt die vorhergesagte Klasse mit einer dazugehörigen Wahrscheinlichkeit, d.h. wie sicher sich das Modell mit seiner Aussage ist, aus. Das Transformer-Modell konnte leider nicht zum Laufen gebracht werden, mehr dazu im folgenden Kapitel.

2.3 Herausforderungen

Die quasi einzige und größte Herausforderung lag darin, das Transformer-Modell zum Laufen zu bringen. Wir konnten es lediglich trainieren, nicht aber testen oder nutzen. Haben wir die Funktionen zum Testen/Nutzen des Modells angepasst, wie es Experten oder die Dokumentation des Moduls „tensorflow“ vorgeschlagen haben, so gab es daraufhin Probleme mit dem Training des Modells. Teilweise kam es zu Fehlermeldung, die man nicht mal mittels Google genauer ermitteln konnte. Da die Zeit irgendwann knapp wurde und das neuronale Netz gut genug performt hat, haben wir den Ansatz mit dem Transformer vorerst fallen ge-

lassen. Der Code existiert aber noch und kann in Zukunft mit mehr Zeit eventuell korrigiert bzw. angepasst werden.

2.4 Lessons Learned

Trotz guter Dokumentation und einer großen Community mit Leuten, die sich in dem Bereich gut auskennen, war es uns nicht möglich unser Transformer-Modell für die Klassifikation zum Laufen zu bringen. Warum es bei der Zusammenfassung ohne Probleme funktioniert hat und bei der Klassifikation nicht, ist uns ein Rätsel. TensorFlow hat mehrmals unsere Python-Umgebungen zerschossen und wir mussten sie neu aufsetzen. Das zeigt uns, dass man manchmal in der Entwicklung auf einen vermeintlich schwächeren Lösungsansatz ausweichen muss, um einen MVP zum Laufen zu bekommen. In diesem Fall ist das neuronale Netz nicht schlecht, ein Transformer könnte aber wahrscheinlich noch genauer arbeiten. Wir haben noch Glück im Unglück.

3 Zusammenfassung

Im Rahmen eines universitären Projekts besteht die Notwendigkeit, eine große Menge an Textdaten effizient zu verwalten und zu analysieren. Die manuelle Durchführung dieser Aufgaben ist zeitaufwendig und unterliegt menschlichen Fehlern und Einschränkungen. Daher wird ein Textzusammenfassungstool benötigt, um den Prozess der Datenverarbeitung zu automatisieren und zu optimieren. Durch die Implementierung dieses Tools können Texte effizient analysiert und gemäß festgelegter Kompressionsraten abhängig von der Anzahl der Wörter zusammengefasst werden, wodurch der Aufwand für manuelle Datenanalysen erheblich reduziert wird.

3.1 Projektumfang

Der Projektumfang umfasste die folgenden Aktivitäten:

- Forschung und Literaturrecherche zu bestehenden Ansätzen für die Textkompression und den Einsatz von Transformer-Modellen.
- Auswahl und Beschaffung der benötigten Datensätze und Ressourcen für das Training des Modells.
- Implementierung und Training mehrerer Transformer-Modelle unter Verwendung verschiedener Kompressionsraten und Hyperparameter.
- Durchführung von Evaluierungs- und Leistungstests, um die Effektivität und Qualität der komprimierten Texte zu bewerten.
- Vergleich der Ergebnisse der verschiedenen Modelle und Identifizierung des besten Ansatzes zur Textkompression.

3.2 Projektziel

Das ursprüngliche Projektziel umfasste die Entwicklung und Implementierung von mehreren Transformer-Modellen für die Textkompression. Die Umsetzung sollten mit verschiedenen Modellen wie Pegasus, BART oder TF-5 geschehen. Die Vorgehensweise wurde jedoch geändert, um adäquate Ergebnisse zu erzielen. Weitere Informationen sind im Unterkapitel „Herausforderungen“ und „Lösungsansätze“ zu finden (3.3).

3.3 Herausforderungen und Lösungsanstätze

Während der Durchführung des Projektes traten einige Probleme und Herausforderungen auf, die die Entwicklung eines effektiven Textkompressionsmodells beeinflussten. Die wichtigsten Schwierigkeiten waren:

Maximale Verarbeitungslänge

- Begrenzte Token-Länge der Modelle: Die meisten der untersuchten Transformer-Modelle hatten eine maximale Token-Länge von 256 bis 512, was die Kompression langer Texte erschwerte und zu Informationsverlust führte.
- Alternativer Ansatz: das Pegasus-Modell kann 4096 Tokens aufnehmen
- Problem: die Ausgaben des Modells sind immer auf eine feste Länge komprimiert (30 %)

Idee: Aufteilen in Textblöcke

- Alternativer Ansatz: Die Aufteilung des Textes in Blöcke von 180 Tokens
- Problem: Die Ergebnisse der Zusammenfassungen sind nicht zufriedenstellend. Sätze werden einfach unterbrochen, ohne am Satzende zu sein.

Die untersuchten Ansätze wurden für unzureichend befunden und verworfen. Im folgenden Unterkapitel (3.4) wird der finale Lösungsansatz beschrieben.

3.4 Lösungsansätze

Um die oben genannten Probleme zu bewältigen und ein effektives Textkompressionsmodell zu entwickeln, wurden verschiedene Lösungsansätze erforscht und implementiert:

- Eingrenzung der Kompressionsrate: Die Kompressionsrate wurde auf einen variablen Bereich von 20 % bis 80 % begrenzt, die für diesen Bereich konsistent gute Ergebnisse erzielt werden können. Die weggefallenen Randbereiche wurden inhaltlich zudem als weniger relevant bewertet, anders als in den Anforderungen zunächst definiert.
- Kompressionsspielraum: Bei Auswählen einer Kompressionsrate wird dem Modell ein Bereich von +-5% ausgehend von diesem Wert gewährt, welcher als korrekt aus Sicht der Länge komprimiert gewertet wird.
- Hybrider Ansatz: Die finalen Modelle bestehen aus einem extraktiven Ansatz (TextRank) und einem abstraktiven Ansatz (Paraphrase), um eine optimale Kompressionsrate sowie Textqualität zu erzielen. Es wurde von dem alternativen abstraktiven Ansatz abgesehen, anders als zunächst in den Anforderungen beschrieben aufgrund der in Kapitel 3.3 beschriebenen Probleme.
- TextRank: Durch untersuchen von Key Words werden die relevantesten Sätze eines Textes identifziert. Die Kompressionsrate bestimmt die Anzahl der extrahierten Sätze. Über diesen Ansatz wird vor allem die Output-Textlänge gesteuert und der Inhalt für die Paraphraser vorbereitet.

- Paraphraser: Jede Textkategorie hat einen eigens trainierten Paraphraser. Die Paraphraser-Modelle basieren auf dem Bart-Large-CNN Modell und wurden auf Trainigsdaten der einzelnen Kategorien trainiert, evaluiert und getestet. Das Bart-Modell hat sich nach Testen der möglichen Modelle als bestes Modell erwiesen. Zeitgleich bildet ein bereits fertig-trainiertes Bart-Modell auch die Grundlage für das Erstellen der Trainingsdaten. Dadurch entstand ein Synergieeffekt, da bereits eine selbsterstellte und -angepasste Pipeline wiederverwendet werden konnte.

3.5 Erreichte Ziele

- Entwicklung von Zusammenfassungsmodellen für alle Kategorien: Wir haben erfolgreich Zusammenfassungsmodelle basierend auf der hybriden Ansatz für alle vorgegebenen Kategorien entwickelt. Dies beinhaltet das Trainieren und Evaluieren der Modelle. Diese Modelle sind in der Lage, Texte unterschiedlicher Domänen wie wissenschaftliche Texte, Nachrichten, Geschichten und Bewertungen zu komprimieren und inhaltlich stimmige Zusammenfassungen zu erstellen.
- Berücksichtigung der Kompressionsrate: Ein zentrales Ziel war es, sicherzustellen, dass die gewünschte Kompressionsrate in den Zusammenfassungen angemessen berücksichtigt wird, ohne die Textqualität zu beschränken. Die Kombination von Textrank und Paraphraser ist gut geeignet, um die Texte auf eine optimale Länge zu reduzieren, während wichtige Informationen und Kontextinformationen erhalten bleiben.
- Inhaltlich stimmige Zusammenfassungen: Ein weiteres wichtiges Ziel war es, sicherzustellen, dass die erstellten Zusammenfassungen inhaltlich stimmig sind und den Kern der Originaltexte korrekt wiedergeben. Dieses Ziel wurde durch unseren Lösungsansatz erreicht. So wurden Fehler wie unvollständige Sätze behoben.
- Zeitmanagement: Die Erstellung und Evaluierung der Modelle erfolgte im vorher gesetzten Zeitrahmen. Durch das Aufnehmen von geeigneten Zeiträumen und Puffern wurden zeitliche Probleme umgangen.
- Ausrichtung auf Wiederverwendbarkeit: Durch das Erstellen von Funktionen und Pipelines war es uns möglich, verschiedene Ansätze und Modelle zu erstellen und zu trainieren, ohne dass viele Zeitressourcen verschwendet wurden. Dies hat es uns zudem ermöglicht, einheitliche Modelle für alle Kategorien zu erstellen.

3.6 Lessons Learned

Während der Entwicklung des Zusammenfassungsmoduls haben wir wertvolle Lessons Learned gewonnen, die uns dabei geholfen haben, ein effektiveres und erfolgreicheres System zu schaffen:

Flexibilität Unserer erster Ansatz, der Transformer-Modelle zur Zusammenfassungserstellung nutzen wollte, zeigte sich als nicht ausreichend für unsere Ansprüche. Aus diesem Grund mussten wir uns spontan für eine Alternativlösung entscheiden, welche die gesteckten Anforderungen erfüllt.

Wichtigkeit von Testen Zusammenhängend mit dem vorhergehenden Punkt, ist es wichtig zu erkennen, ob die aktuelle Richtung der Arbeit funktioniert und zu guten Zielen führt. Nur durch das Testen und Ausprobieren von Alternativen konnten wir feststellen, dass andere Ansätze bessere Ergebnisse liefern. Aus diesem Grund wurden vorher in den Zeitplänen Zeiträume für das Testen von Modellen eingeräumt. Die gelungene Umsetzung bestärkt uns in der Wichtigkeit des Testens.

Qualität und Kompressionsrate Es gibt eine Trade-off zwischen Qualität der Zusammenfassungen und Kompressionsrate. Wenn der Fokus auf einem der beiden Punkte liegt, sinkt der andere erheblich. Da beide jedoch gleich wichtig sind, ist eine Verschlechterung der endgültigen Kompressionsrate nicht abwendbar. Andernfalls entstehen Probleme wie in Kapitel 3.3.

4 Frontend

Dieses Kapitel beschäftigt sich damit, wie wir vorgegangen sind, um unsere geplanten Anforderungen an das Frontend zu erfüllen, welche Probleme dabei entstanden und was wir bei der Bearbeitung gelernt haben.

4.1 Geplante Umsetzung

Für das Tool zur Textklassifikation und -zusammenfassung sollte keine ansprechende grafische Oberfläche, sondern ein einfaches, prototypisches Frontend erstellt werden. Zudem konnten die Anforderungen der Einsprechfunktion von Texten, sowie sollten die Anforderung des Hochladens von Dateien in .pdf, .docx und .txt Formaten mit eingebaut werden. Die Modelle zur Textklassifikation und -zusammenfassung sollten an das Frontend angebunden werden. Die Anforderungen an die Softwareergonomie und einer barrierefreien Gestaltung für Menschen mit Sehbehinderung wurden beachtet.

4.2 Durchführung

Die Oberfläche wurde prototypisch mittels streamlit erstellt, und beinhaltet nur die definierten Aspekte der Eingabe rohen Texts, des Einsprechens von Text und des Hochladens von Textdateien in den angegebenen Formaten. Die Softwareergonomie wurde durch Feedback, internen Usability-Tests sowie einem konsistenten Interaktionsdesign sichergestellt. Zudem ist eine Vorlesefunktion entwickelt worden, und die Schriftgröße veränderlich per „Strg/Cmd“ & „+ “.

4.3 Herausforderungen

Die größte Herausforderung bestand darin, passende Funktionen für die notwendigen Funktionalitäten zu entwickeln. Insbesondere die Speech-to-Text Funktion, die freie Gestaltung der Schriftgröße sowie die Anbindungen der Modelle stellten sich als zeitintensive Hürden dar.

4.4 Lessons Learned

Die Lessons Learned im Kapitel des Frontends beinhalten auf jeden Fall den Aspekt der mangelnden Zeit, entweder sollte hier eine zusätzliche Ressource herangeschafft werden, die sich mit der Thematik auskennt, oder bei einem weiteren Projekt in dieser Zusammenstellung sollten ein paar zusätzliche Tage für die Erstellung des Frontends eingeplant werden.

5 Zusatz-Features

Im Rahmen dieses Projekts hatten wir das Ziel, zusätzliche Module für den Dokumenten-Upload und die Spracheingabe (Text-to-Speech) zu entwickeln. Die Erweiterung des bestehenden Systems sollte es den Benutzern ermöglichen, Dokumente hochzuladen und den Inhalt von Texten durch Spracheingabe zu generieren.

5.1 Projektumfang

Der Projektumfang umfasste die Implementierung und Integration der beiden Module in das bestehende System. Das Dokumenten-Upload-Modul sollte es den Benutzern ermöglichen, Dokumente in verschiedenen Formaten hochzuladen, die Formate PDF, DOXC und TXT. Das Text-to-Speech-Modul sollte die Spracheingabe durch den Benutzer erkennen und in Text umwandeln, der dann weiterverarbeitet werden kann.

5.2 Herausforderungen und Lösungsansätze

Während der Entwicklung des Projekts traten einige Herausforderungen auf. Eine der Hauptprobleme war die Genauigkeit der Spracheingabeerkennung. Die verwendete Spracherkennungssoftware erkannte manchmal Wörter falsch, was zu fehlerhaften Ausgaben führte. Als Lösung haben wir die Möglichkeit geschaffen, dass Benutzer manuell nachbessern können, indem sie die erkannten Wörter korrigieren und erneut bestätigen können.
Zudem fehlte bei der Nutzung der Speech-To-Text-Funktion jegliche Form von Interpunktion. Dafür wurde ein zusätzliches Model in der Datenverarbeitung angebunden, welches dieses Problem behebt.
Für die Implementierung der Hochladefunktion wurden die Bibliotheken „tkinter“, „docx“ und „pyPDF2“ genutzt. Die Speech-to-Text-Funktion nutzt die Module „speech_recognition“ und „gtts“.

5.3 Erreichte Ziele

Wir freuen uns, bekannt zu geben, dass wir alle gesteckten Ziele erfolgreich erreicht haben. Die Integration der Module in das bestehende System wurde erfolgreich durchgeführt, und die Benutzer können nun Dokumente hochladen und die Spracheingabe verwenden, um Texte zu generieren.
Das Dokumenten-Upload-Modul unterstützt verschiedene Dateiformate und ermöglicht es Benutzern, ihre Dokumente auf einfache Weise hochzuladen und in das System zu integrieren.

Das Text-to-Speech-Modul erkennt die Spracheingabe der Benutzer und wandelt sie in Text um. Die Genauigkeit der Spracherkennung kann durch die Möglichkeit für den Anwender eine manuelle Nachbesserung durchzuführen, verbessert werden.

5.4 Lessons learned

Die Umsetzung der extra Module verlief relativ reibungslos. Alleine das Ausführen auf MacOS-Systemen erforderte Anpassungen. Deshalb ist einer der größten Punkte, den wir aus diesem Arbeitsteil mitnehmen können, dass schon mit verhältnismäßig wenig Aufwand die User-Experience durch Vereinfachungen wie Dokumenten-Upload und Spracheingabe maßgeblich verbessert werden kann.

6 Gesamtes Fazit zum Projekt

Insgesamt sind wir mit der Umsetzung des Projekts sehr zufrieden. Trotz des anfänglich teils überfordernden, sehr großen Umfangs des Projekts und unserer Skepsis hinsichtlich der Machbarkeit der gesamten Anforderungen, in der doch kurzen Zeit, konnten wir alle Aufgaben zeitig erfüllen. Auch unsere Ergebnisse, die der Klassifikation und der Zusammenfassung haben für die gegebenen Umstände und Anforderungen eine gute Performance erreicht. Das Frontend mit Zusatzfeatures wurde funktional umgesetzt und mit Funktionalitäten für die Barrierefreiheit ergänzt.
Die wichtigsten Lessons Learned sind für uns:

- Der große Projektumfang konnte mit regelmäßigen Absprachen und einem durchgeplanten Projektmanagement bewältigt werden
- Die Teamarbeit und der regelmäßige Austausch sind wichtig
- Manchmal können Arbeitssitzungen auch länger als geplant dauern
- Die Aspekte, die nicht im Scope festgehalten wurden, müssen wir nicht beachten, somit hatten wir indirekt auch einen Einfluss auf die Gestaltung des Projekts
- Die Datenbeschaffung und -bereinigung benötigen immer etwas an Zeit, diese haben wir ganz gut eingeschätzt
- Eigene Modelle aufzubauen mit mangelnden Rechenressourcen ist sehr zeitintensiv

Einen Ausblick für zukünftige Projekte möchten wir noch in der Hinsicht geben, dass es wichtig ist, Personen, mit verschiedenen Kenntnissen und Fähigkeiten in ein Projektteam mit aufzunehmen, damit man sich gegenseitig gut ergänzen kann. Zudem sollte man mit regelmäßigen Absprachen für regen Austausch sorgen, auch sollte nach sehr langen Arbeitseinheiten noch Zeit für ein paar Scherze sein, um das Teambuilding zu fördern. Dass das Projektmanagement, insbesondere das Management von Personal und Zeit notwendig und wichtig sind, hat uns dieses Projekt für die Zukunft nochmals aufgezeigt. Wir werden darauf und auf die Teamarbeit und den Zusammenhalt, auch in anstrengenden Phasen, in Zukunft weiter bauen.