

Jessica Hassibi
6377520
Bachelor (PO 2011)
Informatik / Anwendungsfach Musik
8. Semester
s6653550@stud.uni-frankfurt.de

Bachelorarbeit

Multi- und monolinguale Textklassifikation von Wikipedia-Artikeln

Jessica Hassibi

Abgabedatum: 28.11.2022

Text Technology Lab
Betreuer: Prof. Dr. Alexander Mehler

Erklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen Quellen oder Hilfsmittel als die in dieser Arbeit angegebenen verwendet habe.

Ort, Datum

Unterschrift

Danksagung

Zunächst danke ich Prof. Dr. Alexander Mehler für die Vergabe dieses spannenden Themas. Durch die Motivation eines musikwissenschaftlichen Anwendungsfalls wurde ich auf einen Bereich der Musikgeschichte aufmerksam gemacht, der mir zuvor eher unbekannt war. Außerdem danke ich dem Team des Text Technology Labs, das mich technisch beriet und mir vielfältige Impulse für die Arbeit gab. Insbesondere war Mevlüt Bagci stets erreichbar und hilfsbereit. Ein ganz besonderer Dank geht an meinen Ehemann und meine Mutter, die mich zu jeder Zeit unterstützt haben. Die Bachelorarbeit widme ich meinem wunderbaren Sohn.

Zusammenfassung

Diese Bachelorarbeit befasst sich mit dem Problem der automatischen Textklassifikation auf einem kleinen, mehrsprachigen Datensatz und experimentiert mit verschiedenen Ansätzen eine binäre Textklassifikation durchzuführen.

Unter der Ausnutzung der Möglichkeit, Wikipedia-Artikel zu einem bestimmten Konzept in verschiedenen Sprachen zu erhalten, wurden zunächst zu einem Thema der Musikgeschichte mittels Web Scraping Texte in den Sprachen Deutsch und Englisch extrahiert.

Bei den Texten handelt es sich um den biografischen Teil der Wikipedia-Artikel von Komponisten des 20. Jahrhunderts. Während die eine Gruppe ermordet, oder ins Exil getrieben wurde, profitierten andere Komponisten vom Nationalsozialismus.

Es wurde angenommen, dass es bestimmte Merkmale innerhalb der Texte einer Gruppe gibt, welche eine Maschine, die auf den Texten trainiert wurde, befähigen könnten, die „Klassen-zugehörigkeit“ eines Komponisten zu bestimmen. Mit transformer-basiertem Topic Modeling konnten Themen aus den Texten extrahiert werden, welche in einer Variante der in dieser Arbeit verglichenen Textklassifikationen verwendet wurden. Als Features für den Machine Learning Algorithmus für die Textklassifikation können Word Embeddings dienen. Hier wurden die Word Embedding Techniken Word2Vec und fastText verwendet. Ein umfangreiches Pre-Processing und Tokenisierung in der jeweiligen Sprache der Texte wurde vor dem Training durchgeführt. Es wurde experimentiert, ob sprachübergreifende und vortrainierte fastText-Embeddings die Textklassifikation verbessern.

Eine Herausforderung für Textklassifikation wurde in der Unausgeglichenheit des Datensatzes vermutet. Um dem entgegenzuwirken, wurden Resamplingstrategien getestet.

Mit den genannten Techniken wurden verschiedene Textklassifikationen mit den Klassifikatoren Support Vector Machine und Naïve Bayes durchgeführt. Im Vergleich dazu wurde Finetuning von state-of-the-art vortrainierten BERT-Modellen auf der Task der Textklassifikation betrieben, was sowohl für die monolingualen Fälle, als auch für den multilingualen Fall im besten Textklassifikationsmodell resultierte.

Inhaltsverzeichnis

| | |
|--|-----------|
| 1. Einleitung | 10 |
| 1.1. Motivation | 10 |
| 1.2. Ziel | 11 |
| 1.3. Methoden | 12 |
| 1.4. Aufbau der Arbeit | 12 |
| 2. Grundlagen | 13 |
| 2.1. Textrepräsentationen | 13 |
| 2.1.1. Weighted Words | 13 |
| 2.1.2. Word Embeddings | 14 |
| 2.1.3. Contextual Word Embeddings | 18 |
| 2.2. Themenmodellierung mit BERTTopic | 20 |
| 2.3. Textklassifikation mit Machine Learning Methoden | 22 |
| 2.3.1. Feature Extraction | 24 |
| 2.3.2. Wahl des Klassifikationsalgorithmus | 24 |
| 2.3.3. Evaluationsmetriken | 26 |
| 3. Forschungsstand | 27 |
| 3.1. Nutzung von Wikipedia für die Generierung von Trainingsdaten | 27 |
| 3.2. Vergleichsstudien zur multilingualen Textklassifikation | 28 |
| 3.3. Kombination von transformer-basiertem Topic Modeling und Word Embeddings für die Textklassifikation | 28 |
| 4. Exkurs zur Musikgeschichte: Komponisten im Nationalsozialismus | 30 |
| 5. Methodik | 32 |
| 5.1. Klassifikation von Texten eines geisteswissenschaftlichen Themas | 32 |
| 5.2. Wikipedia als Datenquelle | 32 |
| 5.3. Textklassifikation mit Word Embeddings und Machine Learning Methoden | 33 |
| 5.3.1. Wahl der Features | 33 |
| 5.3.2. Alignment der fastText-Embeddings | 34 |
| 5.3.3. Anreicherung der Word Embeddings durch Topic Modeling | 34 |
| 5.3.4. Wahl der Klassifikationsverfahren | 35 |
| 5.4. Wahl der vortrainierten BERT Modelle | 37 |
| 5.4.1. Die verwendeten Modelle | 37 |
| 5.4.2. Methodik für das Finetuning | 38 |
| 6. Umsetzung | 39 |
| 6.1. Verwendete Technologien | 39 |

| | | |
|------------------|--|-----------|
| 6.2. | Daten | 40 |
| 6.2.1. | Wikipedia Scraping | 40 |
| 6.2.2. | Filtern der relevanten Teile der Artikel | 42 |
| 6.2.3. | Analyse der Daten | 43 |
| 6.3. | Feature Extraction | 44 |
| 6.3.1. | Datenbereinigung und -vorverarbeitung | 44 |
| 6.3.2. | MUSE | 45 |
| 6.3.3. | BERTopic | 47 |
| 6.3.4. | Training auf um Themenwörter erweiterte Texte | 51 |
| 6.3.5. | Resampling von nicht-balancierten Trainingsdaten | 53 |
| 6.4. | Finetuning von BERT für die Textklassifikation | 54 |
| 7. | Evaluation | 55 |
| 7.1. | Ergebnisse der besten Modelle | 55 |
| 7.1.1. | Textklassifikation mit Support Vector Machine und Naïve Bayes | 55 |
| 7.1.2. | Textklassifikation mit BERT | 56 |
| 7.2. | Reflexion | 56 |
| 7.2.1. | Reflexion der Textklassifikationen mit SVM und NB | 56 |
| 7.2.2. | Reflexion der Textklassifikation mit BERT | 57 |
| 7.3. | Musikwissenschaftliches Abfragen des besten Modells | 58 |
| 8. | Schlussfolgerungen und Ausblick | 60 |
| A. | Sprachverteilungen der extrahierten Artikel, Paragraphen, Sätze und Wörter | 62 |
| B. | Teilmenge der von BERTopic generierten Topic-Cluster | 64 |
| C. | Vollständige Auswertungen der Textklassifikation mit Support Vector Machine und Naïve Bayes | 70 |
| D. | Vollständige Auswertungen der Textklassifikation mit BERT Modellen | 74 |
| Literatur | | 78 |

Abbildungsverzeichnis

| | | |
|-------|---|----|
| 2.1. | Word2Vec Architekturen CBOW und SG | 15 |
| 2.2. | Illustration der Vektorraum Alignment Methode | 17 |
| 2.3. | Pre-Training und Finetuning der BERT Modelle | 19 |
| 2.4. | Architektur eines Sentence Transformers | 20 |
| 2.5. | BERTopic nutzt c-TF-IDF für die Topic Repräsentation | 22 |
| 2.6. | Text Classification Pipeline | 23 |
| 2.7. | Support Vector Machine | 25 |
| 3.1. | Illustration der Anreicherung von Textrepräsentationen mit Topic Modeling | 29 |
| 4.1. | Titelbild der 1939 erschienenen Broschüre zur Ausstellung „Entartete Musik“ | 30 |
| 5.1. | Alignment der Word Embeddings | 34 |
| 5.2. | Anreicherung von Texten mit Topic Modeling | 35 |
| 5.3. | Limitationen des multilingualen BERT Modells auf einer deutschen Sequenz | 38 |
| 6.1. | Ausschnitt der „Liste der vom NS-Regime oder seinen Verbündeten verfolgten Komponisten“ | 40 |
| 6.2. | Konsolenoutput des Data Scrapings mit Filtern von Komponisten | 42 |
| 6.3. | Aufbau des Wikipedia-Artikels über Carl Orff | 42 |
| 6.4. | Ausgewählte Wortvektoren in Englisch und Deutsch im 2-dimensionalen Vektorraum vor dem Alignment | 45 |
| 6.5. | Ausgewählte Wortvektoren in Englisch und Deutsch im 2-dimensionalen Vektorraum nach dem Alignment | 46 |
| 6.6. | Ausgewählte vortrainierte alignierte Wortvektoren in Englisch und Deutsch im 2-dimensionalen Vektorraum | 46 |
| 6.7. | Generierte Topics und ihre Häufigkeiten des multilingualen BERTopic Modells | 47 |
| 6.8. | c-TF-IDF Scores der 8 häufigsten Topics im deutschen BERTopic Modell . . . | 48 |
| 6.9. | c-TF-IDF Scores der 8 häufigsten Topics im englischen BERTopic Modell . . | 48 |
| 6.10. | Themenzuordnungen der Sätze durch multilinguale Topic Modeling | 49 |
| 6.11. | Ausgewählte Topic Cluster im 2-dimensionalem Raum | 50 |
| 6.12. | Topic Cluster verfolgter Komponisten im 2-dimensionalem Raum | 51 |
| 6.13. | Balkendiagramm der Verteilung ausgewählter Topics pro Klasse | 52 |
| 6.14. | Konsolenoutput des Mappings von Themen auf Artikel | 52 |
| 6.15. | Redundanz bestimmter Topics bei Artikeln | 53 |
| 6.17. | Ausschnitt aus dem Trainingsdatensatz für das BERT Modell | 54 |
| 7.1. | Abfragen des besten Modells nach der Klassenzugehörigkeit des „gottbegnadeten“ Richard Strauss | 58 |

| | |
|---|----|
| 7.2. Abfrage des besten Modells nach der Klassenzugehörigkeit des ungesesehenen Felix Mendelssohn Bartholdy | 58 |
| 7.3. Abfragen des besten Modells nach der Klassenzugehörigkeit des ungesesehenen Richard Wagner | 59 |
| 7.4. Veränderter Input in der Abfrage des besten Modells nach Richard Wagner | 59 |

Tabellenverzeichnis

| | |
|---|----|
| 2.1. Exemplarische Wortvektoren einer 1-aus-10-Kodierung. | 13 |
| 5.1. Das BERT-Base Modell. | 37 |
| 6.1. Klassen- und Sprachverteilung der Wikipedia-Artikel. | 43 |
| 6.2. Durchschnittliche Paragraphen, Sätze und Wörter der extrahierten Datensätze. | 43 |
| 7.1. Performanz der besten Klassifikationsverfahren auf fastText- und Word2Vec-Embeddings. | 55 |
| 7.2. Konfiguration und Performanz der besten der jeweils verwendeten BERT Modelle auf den Datensätzen. | 56 |
| C.1. Performanz der Textklassifikation mit Support Vector Machine und Naïve Bayes auf dem deutschen Datensatz. | 71 |
| C.2. Performanz der Textklassifikation mit Support Vector Machine und Naïve Bayes auf dem englischen Datensatz. | 72 |
| C.3. Performanz der Textklassifikation mit Support Vector Machine und Naïve Bayes auf dem multilingualen Datensatz. | 73 |
| D.1. Konfiguration und Performanz von gBERT auf dem deutschen Datensatz. . . | 75 |
| D.2. Konfiguration und Performanz von mBERT auf dem deutschen Datensatz. . | 75 |
| D.3. Konfiguration und Performanz von eBERT auf dem englischen Datensatz. . | 76 |
| D.4. Konfiguration und Performanz von mBERT auf dem englischen Datensatz. . | 76 |
| D.5. Konfiguration und Performanz von mBERT auf dem multilingualen Datensatz. | 77 |

1. Einleitung

1.1. Motivation

Stehen Menschen vor der Aufgabe, einen Text anhand seines Themas, seines Genres oder seiner Grundstimmung (*sentiment*) bestimmten vordefinierten Klassen zuzuordnen, stellt dies häufig keine Schwierigkeit dar, vorausgesetzt sie beherrschen die natürliche Sprache, in welcher der Text verfasst ist. Je nach Themengebiet des Textes und gewählter Granularität der Klassen, kann zudem das Beherrschende von Expertensprache voneinander sein. Manchmal vertrauen Menschen aber auch auf ihr „Bauchgefühl“, beispielsweise wenn es um mehrdeutige, oder sarkastische Texte geht. Abgesehen vom intuitiven, schwerer zu fassenden Ansatz, lassen sich für den Vorgang einzelne logische Schritte definieren. Diese sind z.B. Lesen des Textes und Identifizierung von Merkmalen, die auf die Klassenzugehörigkeit hinweisen. Die automatische Textklassifikation durch eine Maschine folgt einer ähnlichen Logik. Allerdings sind beginnend mit der „maschinengerechten“ Repräsentation der Texte spezielle Schritte erforderlich. Das Ziel ist, Klassenzuordnungen zu erhalten, die möglichst den Ergebnissen der manuellen Textklassifikation durch einen Experten entsprechen.

Der musikhistorische Hintergrund für die in dieser Bachelorarbeit vorgenommene binäre Textklassifikation ist die Verfolgung von als „entartet“ verfemten Komponisten, und im Gegensatz dazu der Unterstützung von als „gottbegnadet“ angesehene Komponisten im Deutschland des Nationalsozialismus. Während die eine Gruppe ermordet, oder ins Exil getrieben wurde, florierte die Karriere anderer Komponisten unter den Nationalsozialisten und sie waren Schutzbefohlene, Täter oder Mitläufer.

Die akademische Musikwissenschaft begann die kritische Aufarbeitung ihrer eigenen politischen Verwicklungen im Nationalsozialismus erst verzögert. Fred K. Prieberg geht in seinem 1982 erschienenen Buch „Musik im NS-Staat“ (Prieberg 1982) als erster deutscher Musikwissenschaftler das Thema umfangreich an. Er beschreibt den Umstand, dass fast alle Komponisten nach 1945 in autobiografischen Angaben Werke verschwiegen haben, die zu politischen Anlässen komponiert wurden und Biografen der Komponisten dies unkritisch übernommen haben (Prieberg 1982). Dies stellte eine der Motivationen dar, die Wikipedia als Datenquelle für die Komponistenbiografien zu nutzen. Interessant ist außerdem vor dem Hintergrund der Unterschiedlichkeit der Musikwissenschaften verschiedener Länder, die Frage, wie sich die Dokumentation dieses Themengebietes in der deutschen Wikipedia zu der anderer Sprachen unterscheidet. Die mögliche Unvollständigkeit der Listen auf Wikipedia, insbesondere in Sprachen außerhalb des Deutschen ist ein Nachteil, der für diese Bachelorarbeit akzeptiert wurde. Eine Erweiterung des Korpus um fehlende Personen würde, die Existenz frei verfügbarer und mehrsprachiger Quellen vorausgesetzt, viel musikwissenschaftliche Recherchearbeit und informative Operationen darstellen, die den Rahmen dieser Arbeit sprengen würden.

Mit Respekt vor den Einzelschicksalen ins Exil gezwungener, oder ermordeter Komponisten ist zu bemerken, dass sich die Anfeindung von Komponisten im Nationalsozialismus weitgehend an biografischen Merkmalen, wie jüdischer Herkunft, oder Nähe zu oppositionellen Gruppierungen festmachen lässt.

Auch heutzutage existieren selbst in westlichen Ländern Fälle von Anfeindungen und Verfolgung von Musikern und Komponisten. Ein Beispiel betrifft die US-Country-Band „The Chicks“, die im Jahr 2003 durch kritische Aussagen über den damaligen Präsidenten George W. Bush konservative Fans der Country-Musik gegen sich aufbrachte. Es führte sogar zu einem Radio-Boykott ihrer Musik, sowie Morddrohungen gegenüber Bandmitgliedern (Rösinger 2007).

Anfeindungen von Gruppen könnten durch für diesen Zweck trainierte Algorithmen anhand von Texten im Internet aufgedeckt werden. Systematische Hetze auf bestimmte Gruppierungen durch einen objektiven Algorithmus nachzuweisen, ist besonders in der heutigen Zeit, in der viele hitzige Diskussionen im Internet stattfinden und Soziale Medien auch als Plattformen für den Austausch unter Hetzern genutzt werden, hochinteressant. Ein Ansatz dafür könnte sein, dass die Maschine lernt zwei Gruppen anhand von Sprache zu unterscheiden. Abgesehen von diesem speziellen Anwendungsfall gibt es ein großes kommerzielles Interesse an der Textklassifikation (Christopher D. Manning, Raghavan und Schütze 2008). Der bekannteste Anwendungsfall für die Textklassifikation liegt in der Erkennung von Spam-Mails. Die rapid steigende Anzahl an multilingualen Texten im Internet verstärkt die Notwendigkeit von multilingualen Textklassifikationssystemen.

1.2. Ziel

Das Ziel dieser Bachelorarbeit ist ein Klassifikationsmodell zu erhalten, das durch das Training auf Wikipedia-Artikeln über Komponisten gelernt hat, die „Klassenzugehörigkeit“ eines Komponisten zu entscheiden.

Folgende Unterfragen sollen dabei beantwortet werden:

- Kann ein Komponist im Vektorraum sprachübergreifend dargestellt werden und welchen Wörtern steht er nahe?
- Kann die Vermutung bestätigt werden, dass das Training auf um Themenwörter erweiterte Texte die Performanz der Textklassifikation verbessert?

1.3. Methoden

Zunächst wird mit Wikipedia Scraping ein geeigneter mehrsprachiger Korpus generiert. Eine wichtige Rolle für die Klassifikation spielt die Art der Textrepräsentation für den Computer, der natürlichsprachliche Texte nicht „verstehen“ kann. Etabliert haben sich Worteinbettungsmodelle wie Word2Vec und fastText. Da fastText als Erweiterung der Skip-Gram Architektur von Word2Vec auch die Morphologie der Wörter beachtet, wird diese Technik fokussiert. Monolinguale Word-Embeddings können durch Vektor Alignment auf den multilingualen Fall ausgeweitet werden. Das ermöglicht es semantisch ähnliche Wörter über verschiedene Sprachen hinweg im gemeinsamen Vektorraum abzubilden. Ob die Technik auch die Textklassifikationen verbessern kann, wird untersucht.

Eine Kombination des State-of-the-Art und traditionellen Methoden wird durch transformer-basiertes Topic Modeling und statischen Word Embeddings exploriert. Dafür werden aus den Texten extrahierten Topics als zusätzliche Worte den bereinigten Artikeln hinzugefügt, um darauf mit den Embedding Modellen Features für die Klassifikatoren zu generieren. Die These ist, dass um ihre Themenwörter erweiterten Texte biografische Merkmale eines Komponisten wiederspiegeln.

Letztlich beeinflusst die Wahl des Klassifikationsverfahrens die Güte der Vorhersagen. Für diese Bachelorarbeit wurden die Klassifikatoren Naïve Bayes und Support Vector Machine ausgewählt.

Im Vergleich zu dieser Art eine Textklassifikation durchzuführen, wird Finetuning von state-of-the-art vortrainierten BERT-Sprachmodellen auf der Task der Textklassifikation durchgeführt.

Durch Evaluation der Textklassifikationen soll für die monolingualen, sowie den multilingualen Datensatz das beste Modell ermittelt werden.

1.4. Aufbau der Arbeit

Zunächst wird im Grundlagenkapitel für diese Arbeit relevantes Hintergrundwissen vermittelt. Daran schließt sich der Forschungsstand zu Arbeiten an, in deren Kontext die Bachelorarbeit steht.

Interessierte Leser finden in Kapitel 4 einen kurzen Exkurs zur Musikgeschichte, der den Hintergrund der beiden Klassen benennt.

Im Kapitel 5 werden die gewählten Methoden begründet und in Bezug zum Forschungsstand gesetzt.

Das Kapitel 6 zur Umsetzung der Textklassifikationen zeichnet das praktische Vorgehen nach und zeigt Zwischenergebnisse der Schritte auf.

Nachfolgend wird in der Evaluation im Kapitel 7 der Vergleich zwischen den Ergebnissen mit den gewählten Techniken vorgestellt und reflektiert.

Außerdem wird das beste Modell nach Klassenzugehörigkeiten ausgewählter Komponisten abgefragt.

Zum Schluss steht ein Fazit, in dem die erreichten Ziele zusammengefasst werden und ein Ausblick auf mögliche Richtungen von Anschlussarbeiten gegeben wird.

2. Grundlagen

2.1. Textrepräsentationen

Natürlichsprachliche Texte, also Texte in Sprachen, die von Menschen gesprochen und zur Kommunikation genutzt werden, sind für einen Computer zunächst unverständliche Zeichenfolgen. Machine Learning Modelle benötigen daher eine „Übersetzung“.

Wortrepräsentationen betrachten Wörter als die atomaren Einheiten von Texten und repräsentieren diese mit Vektoren (Liu, Lin und Sun 2020). Ganze Texte können dann auf den Wortrepräsentationen aufbauend dargestellt werden.

Eine der einfachsten Möglichkeiten Wortrepräsentationen zu erhalten nennt sich **1-aus-n-Kodierung** (*One-Hot Encoding*). Dabei erhält jedes Wort im Korpus einen eigenen n -dimensionalen Vektor, wobei n die Korpusgröße ist. Die so entstehenden Vektoren werden *sparse vectors* genannt, aufgrund der Tatsache, dass jeder Vektor mit vielen Nullen und nur einer Eins spärlich besetzt ist.

Tabelle 2.1.: Exemplarische Wortvektoren einer 1-aus-10-Kodierung.

| Wort | Wortvektor |
|-----------|--------------|
| Komponist | (0010000000) |
| Musiker | (0000000010) |

Tabelle 2.1 zeigt exemplarische Wortvektoren der kodierten Wörter *Komponist* und *Musiker* einer 1-aus-n-Kodierung mit der Korpusgröße $n=10$. Dass aber die Wörter *Komponist* und *Musiker* mit hoher Wahrscheinlichkeit in einem ähnlichen Kontext verwendet werden, kann mit der 1-aus-n-Kodierung nicht ausgedrückt werden. Außerdem stellt die Dünnbesetztheit mit gleichzeitiger hohen Dimension der Vektoren ein Problem für Machine Learning Modelle dar. Letztlich ist die 1-aus-n-Kodierung sehr unflexibel, da durch jedes neue zu kodierende Wort ein neuer Index erstellt werden muss und somit alle bereits bestehenden Vektoren um eine Dimension vergrößert werden müssten (Liu, Lin und Sun 2020).

2.1.1. Weighted Words

Die Techniken *Bag of Words* (BoW) und *Term Frequency-Inverse Document Frequency* (TF-IDF), werden im Folgenden kurz vorgestellt, da ihre Grundideen für die Entwicklung der in dieser Bachelorarbeit genutzten Techniken der Textrepräsentation und des Themenmodellierungs mit BERTopic wichtig waren.

BoW und TF-IDF erweitern den *Term Frequency* (TF, zu deutsch: Vorkommenshäufigkeit) Ansatz, der jedes Wort auf die Häufigkeit seiner Vorkommen im Dokument mappt (Kowsari

u. a. 2019).

BoW zählt die Häufigkeit der Vorkommen der Wörter im gesamten Dokument. Ein Dokumentvektor wird gebildet, indem die 1-aus-n-kodierten Wörter das Dokument repräsentieren und mit der Vorkommenshäufigkeit dieser Wörter im Dokument gewichtet werden (Kowsari u. a. 2019). Da BoW die 1-aus-n-Kodierung nutzt, übertragen sich die oben genannten Nachteile dieser Methode.

TF-IDF verbindet den TF-Ansatz mit *Inverse Document Frequency* (IDF), einer Methode, die darauf abzielt, die Wirkung von übermäßig in allen Dokument vorkommenden Wörtern (wie „in“ und „das“) durch Gewichtungen zu verringern und gleichzeitig die Wirkung seltener, aber inhaltstragender Wörter verstärken soll (Kowsari u. a. 2019).

Sei $\|tf_{xd}\|$ die normalisierte Vorkommenshäufigkeit des Wortes x in einem Dokument d . Sei weiterhin df_x die Anzahl der Dokumente, die das Wort x enthalten. Dann ist das Gewicht W_{xd} des Wortes x im Dokument d definiert als

$$W_{xd} = \|tf_{xd}\| \cdot \log \frac{N}{df_x},$$

wobei N für die Gesamtzahl der Dokumente steht.

2.1.2. Word Embeddings

Word Embeddings (deutsch: Worteinbettungen, oft auch nur Wortvektoren genannt), haben eine feste Länge, die es dennoch erlauben soll, möglichst viel Information über das Wort aufzunehmen. Diese niedrigdimensionalen Vektoren enthalten Fließkommazahlen und können daher als *dense vectors* beschrieben werden. Die Vektoren können in den n-dimensionalen Raum als Datenpunkte geplottet werden.

Es wurden verschiedene Methoden vorgeschlagen um Word Embeddings aus einem Korpus zu generieren. Die für diese Arbeit relevanten Methoden werden im Folgenden vorgestellt.

Word2Vec

Die Erzeugung von Vektorrepräsentationen von Wörtern durch neuronale Netze wurde durch Word2Vec (Mikolov, Chen u. a. 2013) populär. Motiviert durch die Entwicklungen im Bereich Machine Learning, setzen sich die Autoren das Ziel eine effizientere Berechnung der Wortvektoren auf großen Datenmengen im Vergleich zu den damaligen auf neuronalen Netzen basierenden state-of-the-art Techniken zu erreichen. Mit Word2Vec führten sie zwei neue Architekturen ein: *Continuous Bag-of-Words* (CBOW) und *Skip-gram* (SG). Die Methoden setzen neuronale Netze mit einer versteckten Schicht (*hidden layer*) ein, um Wortvektoren aus *one-hot*-enkodierten Wörtern zu erlernen.

Beide Methoden iterieren über alle Wörter w des Korpus T . $w(t)$ bezeichnet das aktuelle Wort an Position t . Mit einem bestimmten Radius m werden die umliegenden Wörter (sog. „Kontextwörter“) $w(t-m), \dots, w(t-1), w(t+1), \dots, w(t+m)$ von $w(t)$ betrachtet. CBOW versucht das aktuelle Wort basierend auf den umliegenden Wörtern vorherzusagen, während SG den Ansatz umkehrt und aus den umliegenden Wörtern das aktuelle Wort berechnet.

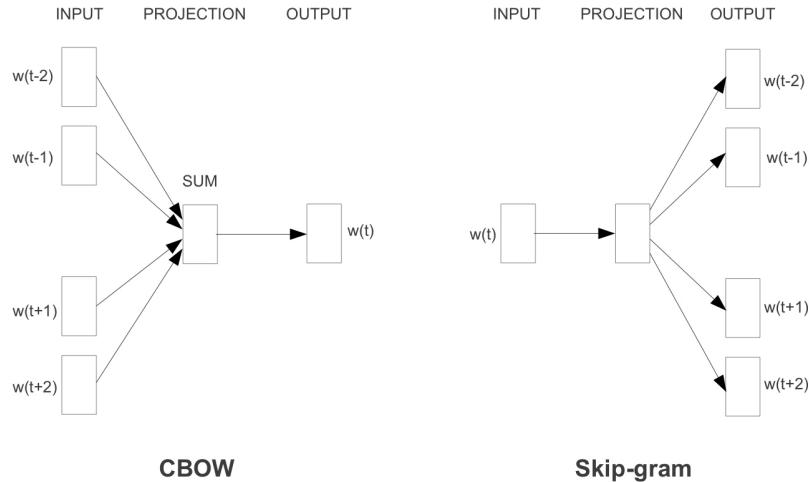


Abbildung 2.1.: Word2Vec Architekturen CBOW und SG aus Mikolov, Chen u. a. (2013, S. 5).

Bereits in vorherigen Studien konnten Mikolov, Yih und Zweig (2013) zeigen, dass semantisch ähnliche Wörter, aufgrund ihrer ähnlichen internen Repräsentation im Vektorraum nah beieinander liegen¹.

Ein vielzitiertes Beispiel dafür ist die algebraische Operation $\text{vector}(\text{„king“}) - \text{vector}(\text{„man“}) + \text{vector}(\text{„woman“}) \approx \text{vector}(\text{„queen“})$ (vgl. Mikolov, Yih und Zweig 2013, S. 1). Diese beruht auf der Analogie $\text{king} - \text{queen} = \text{man} - \text{woman}$ (gesprochen „king verhält sich zu queen wie man zu woman“).

Hervorzuheben ist, dass der bei der Addition bzw. Subtraktion von Wortvektoren entstandene Vektor nicht exakt einem Vektor aus dem Vokabular entsprechen muss. Im genannten Beispiel wird der Vektor für *queen* aus dem Vokabular aufgrund der Nähe zum resultierenden Vektor ausgegeben.

Sowohl Word2Vec, als auch weitere zeitlich nah darauf folgende Modelle, wie GloVe (Pennington, Socher und Christopher D Manning 2014), lernen Word Embeddings auf Wortebene, was den Nachteil hat, dass nur Wörter, die in der Trainingsmenge existierten, vorhergesagt werden können. An dieser Stelle setzt die folgende Worteinbettungstechnik an.

fastText

Im Jahr 2016 erschien der Artikel „Enriching Word Vectors with Subword Information“ von Forschungsgruppe des Facebook AI Research Labs unter Bojanowski u. a. (2016). Die Autoren

¹Die Semantik ist ein Teilgebiet in der Linguistik und bezeichnet die Studie um die inhaltliche Bedeutung von Zeichenfolgen wie sie in Wörtern, Phrasen, Sätzen oder Dokumenten vorkommen (Jurafsky und Martin 2009). Die Bedeutung von lexikalischen Einheiten, beispielsweise von Wörtern, wird lexikalische Semantik genannt. Semantisch ähnliche Wörter sind auf ihre Semantik bezogen ähnlich. Beispielsweise sind die Verben *fiedeln* und *geigen* bedeutungsähnlich (jedoch nicht bedeutungsgleich). *Musizieren* ist ebenso bedeutungsähnlich und würde im Vektorraum idealerweise nah bei den das Geigenspiel beschreibenden Wortvektoren liegen, während beispielsweise *marschieren* im Vektorraum weiter entfernt wäre.

stellten dort die Open-Source-Bibliothek fastText vor. FastText begreift sich als Erweiterung der Word2Vec Skip-gram-Architektur um das Training auf kleineren linguistischen Einheiten (Bojanowski u. a. 2016). FastText zerlegt ein Wort in seine n-Gramme, also Zeichensequenzen der Länge n, und berechnet für diese Vektorrepräsentationen. Für das Wort *Musik* und n=3 würde der Sack der zu *tri*-Grammen zusammengefassten Zeichen wie folgt aussehen: <Mu, Mus, usi, sik, ik>. Eine Wortrepräsentation setzt sich aus der Summe der Vektoren der zu n-Grammen zusammengefassten Zeichen des Wortes zusammen.

Der Vorteil dieser Methode ist, dass selbst Wörter, die nicht in den Trainingsdaten vorkommen, also *Out-Of-Vocabulary* (OOV) Wörter, betrachtet werden können. Besonders profitieren Sprachen wie Deutsch, mit sehr vielen morphologisch-komplexen² Wörtern, von dem Training auf n-Gramm-Ebene, da bestenfalls n-Gramme des unbekannten Wortes schon vom Modell gelernt wurden und daraus eine Vektorrepräsentation des ganzen Wortes gebildet werden kann. Aus den n-Grammen des Beispielworts *Musik* könnten dabei Vektorrepräsentationen vieler weiterer OOV-Wörter abgeleitet werden (*musisch*, *Musikwissenschaft*, *unmusikalisch* usw.).

Sprachübergreifende fastText-Embeddings

Das Training von Word Embeddings (s. Kapitel 2.1.2) findet auf sprachspezifischen Korpora statt. Im Jahr 2013 erkannten Mikolov, Le und Sutskever (2013), dass die Word2Vec-Embeddings verschiedener Sprachen in ihren jeweiligen Vektorräumen ähnliche Strukturen aufwiesen.

Sprachübergreifende Word Embeddings in einem gemeinsamen Vektorraum ermöglichen zum einen den sprachübergreifenden Vergleich von Wortbedeutungen und zum anderen den Transfer von gelerten semantischen Zusammenhängen von Sprachen mit einer reicherem Datengrundlage zu Sprachen mit weniger zur Verfügung stehenden Daten (Ruder, Vulic und Søgaard 2019). *Vector Space Alignment* von monolingualen Word Embeddings ist eine Methode sprachübergreifende Word Embeddings zu erhalten.

Ende 2017 hat Facebook AI Research (FAIR) eine mit „*MUSE: Multilingual Unsupervised and Supervised Embeddings*“ Methode für das lineare Mapping von monolingualen *source* auf monolinguale *target* fastText-Embeddings vorgestellt (Conneau u. a. 2017). Im Gegensatz zu vorher entwickelten Methoden gelingt dies auf eine unüberwachte Art, nämlich ohne einen parallelen Datensatz, oder bilinguale Lexika. Die Autoren stellen vortrainierte auf fastText basierenden Embeddings zum Download zur Verfügung (Conneau u. a. 2017). Außerdem lassen sich mit dem veröffentlichten Code selbst trainierte monolinguale fastText-Embeddings einander angleichen.

²Die Morphologie beschäftigt sich als Teilgebiet der Linguistik mit Wortformen.

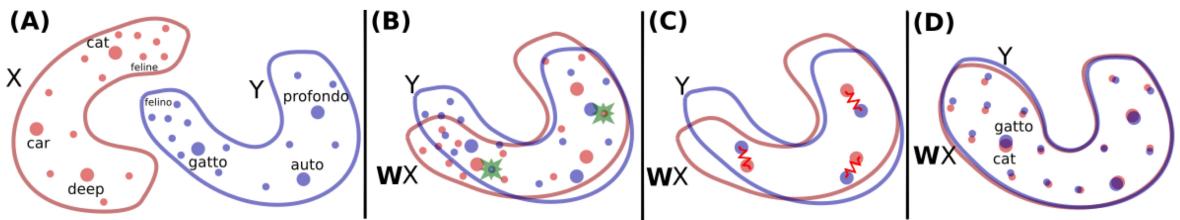


Abbildung 2.2.: Illustration der Vektorraum Alignment Methode aus Conneau u. a. (2017, S. 3).

Im Folgenden wird eine Intuition für die vorgenommenen Schritte anhand der Illustration in Abbildung 2.2 gegeben.

Bild (A) stellt die Ausgangssituation zweier Distributionen von monolingualen Word Embeddings X und Y im Vektorraum dar, wobei rote Punkte für englische und blaue Punkte für italienische Wörter stehen.

Seien $X = x_1, \dots, x_n$ und $Y = y_1, \dots, y_m$ die n bzw. m sprachspezifischen Word Embeddings. Das Ziel ist das Lernen eines linearen Mappings $WX = Wx_1, \dots, Wx_n = Y$. Dabei sollen die „inneren Strukturen“ der Distributionen X und Y erhalten bleiben (Conneau u. a. 2017, S. 3).

Zunächst wird eine Rotationsmatrix W gelernt, die zum groben Alignment von X und Y führt (B). Dafür wird *Adversarial Learning* implementiert, welches ein Ansatz des unsupervised Machine Learning ist. Einem *Discriminator* wird hierbei zufällig ein Datensatz aus WX oder Y übergeben und seine Aufgabe ist es zu entscheiden, von welcher Quelle der Datensatz kommt. Die zufällig ausgewählten Datensätze werden in Abb. 2.2 durch grüne Sterne dargestellt. Es entsteht ein „Wetteifern“, denn während der *Discriminator* seine Fähigkeit maximiert korrekt zu entscheiden, verbessert der sog. *Generator* (hier W) seine künstlichen Datensätze, indem er WX möglichst an Y annähert (Conneau u. a. 2017).

Die so entstandene Matrix W wird nun in einem iterativen Prozess verbessert (C). Dafür wird aus Wörtern, die häufig auftreten (in Abb. 2.2 fett dargestellt) und ihren k-nächsten Nachbarn ein bilinguales Wörterbuch gebildet. Sie dienen somit als Ankerpunkte für das Mapping. Auf dem Wörterbuch wird der Procrustes Algorithmus iterativ angewendet³, um die Qualität des Wörterbuchs zu erhöhen (Conneau u. a. 2017). Schließlich wird das verbesserte Mapping auf alle Wörter angewendet.

Im letzten Schritt (D) findet die Übersetzung mittels W unter der Benutzung des Ähnlichkeitsmaß *CSLS* (Cross-domain Similarity Local Scaling) statt. Letzteres dient der Entzerrung

³Die Autoren verweisen auf Artetxe, Labaka und Agirre (2017).

von dichten Stellen um bestimmte Wörter, wie um das Wort „cat“ auf dem ersten Bild von Abbildung 2.2 (Conneau u. a. 2017).

Sei Wx_s das Mapping eines Word Embeddings der *source* Sprache. Und sei $N_t(Wx_s)$ seine Nachbarschaft mit k Nachbarn. Dann notieren Conneau u. a. (vgl. 2017, S. 4) die durchschnittliche Ähnlichkeit einer *source* Embedding x_s zu seiner *target* Nachbarschaft als

$$r_T(Wx_s) = \frac{1}{K} \sum_{y_t \in N_t(Wx_s)} \cos(Wx_s, y_t),$$

wobei die Kosinus-Ähnlichkeit zum Einsatz kommt, ein Maß für die Ähnlichkeit zweier Vektoren. Die durchschnittliche Ähnlichkeit zu den Nachbarwörtern wird für alle *source* und *target* Embeddings ausgerechnet. Dann werden die erhaltenen Werte eingesetzt in die Berechnung für die Ähnlichkeit von Wx_s und einer *target* Embedding y_t :

$$CSLS(Wx_s, y_t) = 2 \cos(Wx_s, y_t) - r_T(Wx_s) - r_S(y_t)$$

CSLS verringert so die Ähnlichkeit von Wortvektoren in dichten Umgebungen (Conneau u. a. 2017).

Problematik statischer Word Embeddings

Alle bisher besprochenen Methoden berechnen für jedes Wort genau eine Vektorrepräsentation. Problematisch wird dies bei mehrdeutigen Wörtern. Beispielsweise kann das Wort *Ton* sowohl ein plastisch verformbares Material zur Herstellung von Gegenständen, als auch eine durch das Gehör wahrnehmbare Luftschwingung beschreiben.

Die *statischen* Word Embeddings von Word2Vec, GloVe und fastText könnten diesen Umstand nicht mit verschiedenen Vektoren für *Ton* modellieren.

2.1.3. Contextual Word Embeddings

Im Jahr 2018 revolutionierten *Embeddings from Language Models* (ELMo) (Peters u. a. 2018) die Word Embedding Modelle, indem jedes Wort eine eigene Vektorrepräsentation abhängig vom Kontext bekommt. Kontext bedeutet für ELMo nun auch die Position des Wortes im Dokument. ELMo nutzt dafür *Rekurrente Neuronale Netze* (RNNs), deren Zustände die Eigenschaft haben, abhängig von vorhergehenden Zuständen zu sein und folglich auch nachfolgende Zustände zu beeinflussen. Je nach Stelle im Satz entsteht durch das unidirektionale Training mit RNNs eine andere Worteinbettung.

Eine Anwendung von RNNs außerhalb des Natural Language Processing liegt beispielsweise auch in der Generierung von Notentexten, welche durch die positiven Eigenschaften der RNNs für die Verarbeitung von Sequenzen erleichtert wird (Dua u. a. 2020).

BERT

Im Jahr 2019 wurden *Bidirectional Encoder Representations from Transformers* (BERT) von Devlin u.a. (2019) eingeführt. Auch BERT generiert *Contextual Word Embeddings*. Allerdings

nutzt BERT dafür keine RNNs, sondern dem neuen Stand der Technik entsprechenden *Transformer*-Blöcke. Die Transformer-Architektur wurde schon ein Jahr vor ELMo von Vaswani u. a. (2017) vorgestellt. BERT verwendet für das Training die Enkodierer-Blöcke der Transformer mit einem tiefen Netzwerk aus entweder 12 oder 24 Schichten. Das Konzept der Selbstattention der Transformer hilft BERT dabei, den Fokus bei der Berechnung der Wortvektoren auf wichtige Wörter zu setzen.

BERT setzt auf das Nutzen von auf großen, unannotierten Datensätzen und für spezielle Tasks vortrainierten Architekturen. Durch *Finetuning*, also Feinabstimmungen bestimmter Hyperparameter, können diese für *Downstream Tasks*, wie der Task der Textklassifikation, angepasst werden (Devlin u. a. 2019). Im Kapitel 6.4 des Hauptteils dieser Arbeit werden die für das Finetuning infrage kommenden Hyperparameter beschrieben.

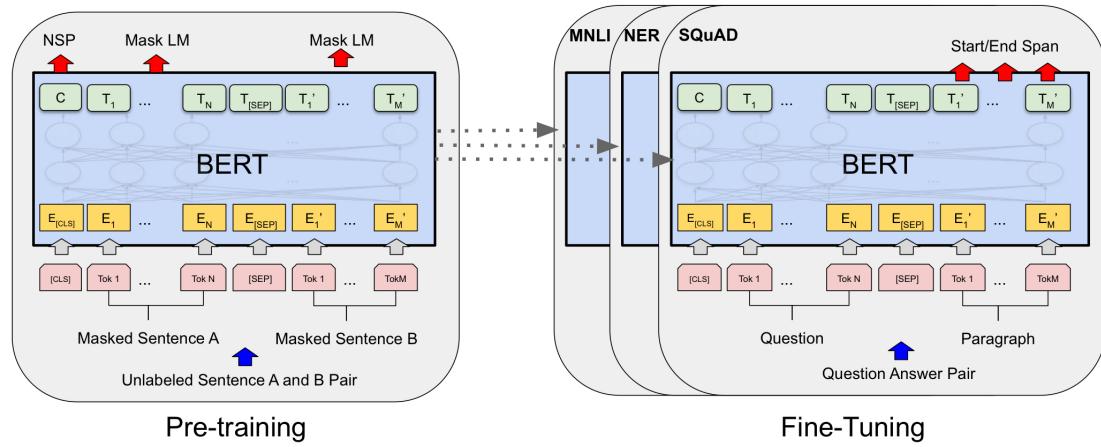


Abbildung 2.3.: Pre-Training und Finetuning der BERT Modelle aus Devlin u. a. (2019, S. 3).

BERT nutzt maskierte Sprachmodellierung (MLM), eine Technik, die zufällig 15% der Wörter in einer Eingabesequenz maskiert, um sie schließlich aus dem Kontext vorherzusagen (Devlin u. a. 2019). Dabei kann es sich bei der Eingabesequenz um einen einzelnen Satz, aber auch um einen Text bestehend aus mehreren Sätzen handeln (Devlin u. a. 2019). Im Gegensatz zu den RNNs wird die gesamte maskierte Sequenz auf einmal im Modell verarbeitet. Die zweite Technik, die BERT für das Training benutzt, ist die Vorhersage des Nächsten Satzes (NSP). Dafür erhält BERT zwei Sätze und sagt voraus, ob der zweite Satz zum Kontext des ersten gehört, wobei er in der Hälfte der Fälle ein zufällig gewählter Satz ist (Devlin u. a. 2019). Das durch diese Techniken entstehende bidirektionale Training führt zu einem tieferen Lernen des Kontexts eines Wortes, da auch weitreichende Zusammenhänge in der Eingabesequenz erfasst werden können.

Multilinguales BERT

Eine der vielen Erweiterungen von BERT ist das multilinguale mBERT, das auf dem konkatenierten Wikipedia-Korpus 104 Sprachen trainiert wurde (Devlin 2019). Das Modell wird in Kapitel 5.4 des Methodik-Teils beschrieben.

SBERT

Sentence-BERT (SBERT) (Reimers und Gurevych 2019) wurde im Jahr 2019 als Modifikation von BERT veröffentlicht. SBERT nutzt ein siamesisches Netzwerk, um Satzeinbettungen unter Beachtung der semantischen Kosinus-Ähnlichkeit generieren.

SBERT-Modelle werden auch Sentence Transformer genannt, da sie, mit BERT als Grundlage, ebenso auf Transformer-Blöcken basieren. Der Output von BERT für einen Satz geht bei SBERT über in ein *Pooling Layer*, welches eine Satzrepräsentation von fester Größe, z.B. 768 Dimensionen für BERT-Base (Devlin u. a. 2019), generiert. Als *mean-pooling* konfiguriert werden alle *Contextual Word Embeddings* für die Wörter des Satzes des BERT Outputs gemittelt.

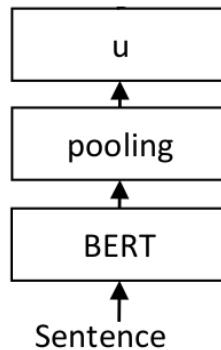


Abbildung 2.4.: Architektur eines Sentence Transformers,
<https://www.sbert.net/docs/training/overview.html>, letzter Zugriff:
21.11.2022 (Screenshot vom 21.11.2022, 20:34 Uhr).

Die Autoren zeigen, dass SBERT für Tasks, die die semantische Ähnlichkeit fokussieren, besser als BERT und die aus dem Durchschnitt von fastText-Wortvektoren gebildete Satzvektoren abschneidet (Reimers und Gurevych 2019). Genau wie BERT-Modelle werden auch SBERT-Modelle vorgenommen und *out-of-the-box* anwendbar zur Verfügung gestellt.

2.2. Themenmodellierung mit BERTopic

Themenmodellierung (*Topic Modeling*) beschreibt die Machine Learning Task des Findens von Themenwörtern zur Beschreibung der Inhalte von Dokumenten eines Textkorpus. Sie zählt zum unüberwachten Lernen (*unsupervised learning*) und ihre Ergebnisse benötigen menschliche Expertise, um gedeutet zu werden.

Da der Fokus der Bachelorarbeit nicht auf dem Topic Modeling liegt, wird die verwendete Topic Modeling Methode BERTopic nur grob vorgestellt. Für ein tiefergehendes Verständnis lohnt sich ein Blick in die hier zitierte Literatur.

BERTopic (Grootendorst 2022) nutzt state-of-the-art vortrainierte Embedding-Modelle, um Themen zu generieren.

die Themen aus Dokumenten extrahiert, indem sie Wörter mit ähnlichen Bedeutungen zu Clustern zusammenfasst.

BERTopic nutzt in seiner Standardkonfiguration folgende Pipeline (Grootendorst 2022):

- *Embeddings*
- *Dimensionality Reduction*
- *Clustering*
- *Topic Representation*

Für die Extraktion der Embeddings werden state-of-the-art vortrainierte Sentence-BERT Modelle (s. Kapitel 2.1.3) genutzt.

Bevor das Clustering durchgeführt werden kann, muss die Dimensionalität der Embeddings reduziert werden, da die großen Dimensionen der generierten Embeddings den Clustering Algorithmen Probleme bereiten können (Grootendorst 2022). In der Standardkonfiguration von BERTopic geschieht dies mit dem UMAP-Algorithmus, welcher die Kosinus-Ähnlichkeit als Distanzmetrik benutzt (McInnes, Healy und Melville 2018).

Der genutzte Clustering-Algorithmus ist HDBSCAN (McInnes, Healy und Astels 2017). Alle Dokumente eines Clusters werden dann zu einem Dokument zusammengefasst, welches das Cluster repräsentiert. Durch Zählen der Häufigkeiten des Vorkommens von Wörtern im Cluster wird eine BoW-Repräsentation gebildet. Im Kapitel 2.1.1 wurde die Idee von BoW beschrieben, allerdings wird die Technik hier auf die Cluster und nicht auf die einzelnen Dokumente angewendet, damit die Wörter auf Topic-Ebene herausgestellt werden können (Grootendorst 2022).

Die eigentliche Repräsentation der Topics wird schließlich mit c-TF-IDF, einer clusterbasierten Version von TF-IDF (s. Kapitel 2.1.1). TF-IDF kann auf einer Liste von Dokumenten angewendet, die Relevanz der Wörter zwischen den Dokumenten vergleichen. c-TF-IDF bekommt die BoW-Repräsentation eines Clusters als Eingabe und kann somit die Relevanz der Wörter in den verschiedenen Clustern bewerten und schließlich die relevantesten Wörter eines Clusters als Topic zusammenfassen (Grootendorst 2022).

c-TF-IDF

For a term x within class c :

$$W_{x,c} = \|\mathbf{tf}_{x,c}\| \times \log\left(1 + \frac{A}{f_x}\right)$$

$\mathbf{tf}_{x,c}$ = frequency of word x in class c

f_x = frequency of word x across all classes

A = average number of words per class

Abbildung 2.5.: BERTopic nutzt c-TF-IDF für die Topic Repräsentation,
<https://maartengr.github.io/BERTopic/algorithm/algorithm.html#3-cluster-documents>, letzter Zugriff: 16.11.2022 (Screenshot vom 16.11.2022, 14:01 Uhr).

BERTopic ermöglicht es, die beschriebene Pipeline *out-of-the-box* anzuwenden. Es ist je nach Nutzen für den Anwendungsfall allerdings auch möglich, die Standardkonfiguration zu verändern, etwa indem andere Word Embedding Modelle, oder PCA für die Reduzierung der Dimensionen genutzt werden.

Das Ergebnis des Topic Modelings mit BERTopic sind dichte Cluster von Wörtern, aus denen Topics interpretiert werden können. Die Ausgabe als „Worte-Cluster“ hat gegenüber einem einzigen Wort den Vorteil, dass der Kontext des Wortes erkennbar wird. Bezug nehmend auf das Beispiel im Abschnitt 2.1.2, könnte somit korrekt interpretiert werden, um welche Bedeutung es sich bei *Ton* handelt, wenn sich das Wort in demselben Cluster wie *Klang*, *Tonkunst* und *Musik* befindet.

2.3. Textklassifikation mit Machine Learning Methoden

Textklassifikation beschreibt die Task der Zuordnung von Dokumenten zu vordefinierten Kategorien und kann folgendermaßen formal notiert werden:

Für eine Menge von Dokumenten D und einer Menge von Kategorien c_1, c_2, \dots, c_n wird jedem Dokument d_i genau eine Kategorie c_j zugewiesen (Ikonomakis, Kotsiantis und Tampakas 2005, vgl. S. 1).

Ikonomakis, Kotsiantis und Tampakas (2005) unterscheiden dabei *text genre classification* von *text topic-based classification*. Ein prominenter Korpus für die *text genre classification* ist der 20 Newsgroups Korpus, welcher Dokumente aus 20 Nachrichten-Kategorien, wie Sport und Weltnachrichten, umfasst. Diese Bachelorarbeit beschäftigt sich mit der *text topic-based classification*. Die Texte dieser Textklassifikationsform haben die Eigenschaft aus ähnlichen Quellen zu stammen und in Format, Vokabular und Schreibstil weitgehend übereinzustim-

men (Ikonomakis, Kotsiantis und Tampakas 2005).

Die automatische (maschinelle) Textklassifikation ist eine Machine Learning Task des überwachten Lernens (*supervised learning*).

Anhand schon klassifizierter Daten berechnet ein lernender Algorithmus Strukturen. Diese ermöglichen es dem Klassifikationsalgorithmus auf unbekannten Daten, die Klasse mit möglichst hoher Gewissheit zu entscheiden. Die Gewissheit der Klassifikation wird bei der *weichen Klassifikation* mit einer bestimmten Prozentzahl angegeben. In dieser Bachelorarbeit wird allerdings eine binäre Entscheidung für eine Klasse erzwungen, was auch *harte Klassifikation* genannt wird.

Für das Training des lernenden Algorithmus wird üblicherweise eine Teilmenge der insgesamt zur Verfügung stehenden Daten mit bekanntem Label genutzt, während die Güte des Klassifizierers auf den restlichen Daten überprüft wird. Man spricht daher von *Trainings- und Testdaten*. Die Einteilung des Datensatzes wird *train-test-split* genannt, wobei auf einem Großteil der Daten trainiert und auf der kleineren Teilmenge getestet wird.

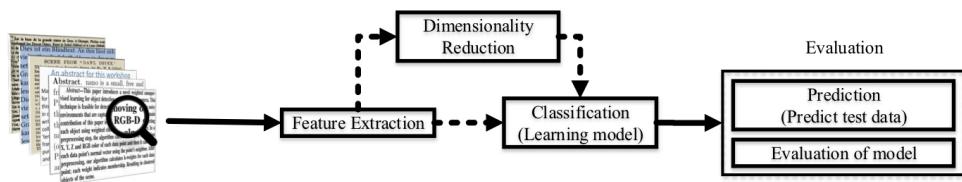


Abbildung 2.6.: Text Classification Pipeline aus Kowsari u. a. (2019, S. 2).

Kowsari u. a. (2019) teilen die Pipeline der Textklassifikation grob in die folgenden vier Schritte ein (siehe Abbildung 2.2.):

- *Feature Extraction*
- *Dimensionality Reduction*
- *Classifier Selection*
- *Evaluations*

Der zweite Schritt ist laut Kowsari u. a. (2019) optional und bezieht sich auf Techniken zur Reduzierung der Dimensionen, wie *Principal Component Analysis* (PCA) oder *Linear Discriminant Analysis* (LDA). Um den Rahmen dieser Bachelorarbeit nicht zu sprengen, wurde der Schritt nicht durchgeführt.

Das Textklassifikationssystem kann auf verschiedenen Textebenen angewendet werden, nach als Klassifikation von Dokumenten, Paragraphen, Sätzen oder Phrasen (Kowsari u. a. 2019). In dieser Bachelorarbeit wird die Textklassifikation auf Dokumentenebene⁴ durchgeführt.

⁴Da es sich bei den Texten um Wikipedia-Artikel handelt, wird im Hauptteil dieser Bachelorarbeit anstatt

2.3.1. Feature Extraction

Dieser Schritt dient der Umwandlung der natürlichsprachlichen Texte in eine Form, die einen geeigneten Input, für den Machine Learning Algorithmus darstellt, die sog. genannten *Features*. Dabei ist die Wahl der richtigen Features essenziell und bei zu vielen Features kann das Problem des Overfittings entstehen (Kowsari u. a. 2019).

Features können beispielsweise Word Embeddings sein, welche mit Feature-Extraction-Techniken wie Word2Vec (Mikolov, Chen u. a. 2013) erzeugt werden. Dies werden detailliert im Kapitel 2.1 des Grundlagenkapitels behandelt.

Doch bevor die Features erzeugt werden, ist es für viele dieser Techniken unerlässlich, die Daten zu bereinigen (Kowsari u. a. 2019). In dieser Arbeit wurden dafür Stopwörter und Punktierungszeichen entfernt. Außerdem wurden die NLP-Techniken Stemming und Tokenisierung angewendet. Im Kapitel 6.3.1 kann dieser Prozess mit Beispielen aus dem Datensatz nachvollzogen werden, weshalb die einzelnen Schritte hier nicht erklärt werden.

2.3.2. Wahl des Klassifikationsalgorithmus

Die Wahl des besten Klassifikationsalgorithmus (auch Klassifikator genannt), ist Kowsari u. a. (2019) zufolge der wichtigste Schritt der Klassifikationspipeline. Für diese Arbeit wurden zwei Klassifikationsalgorithmen ausgewählt, die für die Task der maschinellen Textklassifikation vielversprechende Ergebnisse erwarten lassen.

Support Vector Machines

Support Vector Machine (SVM) ist ein linearer Klassifikator, der für die Task der Textklassifikation bevorzugt gewählt wird (Christopher D. Manning, Raghavan und Schütze 2008). Das Ziel einer SVM ist es, in einem Vektorraum eine optimale Entscheidungsschranke zu ermitteln, um zwei Klassen voneinander zu trennen, und kann daher als binärer Klassifikator beschrieben werden. Optimal ist die Entscheidungsschranke, wenn die trennende Hyperebene von jedem Datenpunkt der Trainingsdaten maximal entfernt ist.

Die Erweiterung des Verfahrens für den Fall, dass die Trainingsdaten nicht linear trennbar sind, sieht vor, dass dennoch versucht wird, eine Hyperebene zwischen den Datenpunkten aufzuspannen und ein kleinstmöglicher Fehler toleriert wird. Alternativ könnten die Vektoren auf eine höhere Dimension gebracht werden, sodass sie linear trennbar sind (Christopher D. Manning, Raghavan und Schütze 2008).

von „Dokumentenebene“ von „Artikelebene“ gesprochen. „Dokumente“ steht hingegen für eine Liste von Texten, welche sowohl Artikel, Paragraphen, als auch Sätze sein können. Diese Terminologie nutzt auch das Topic Modeling (s. Kapitel 2.2)

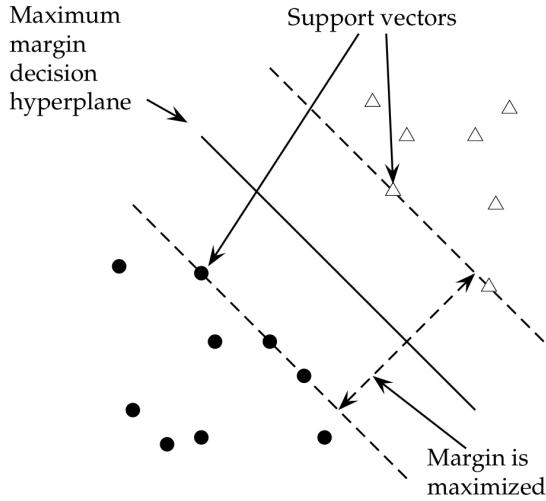


Abbildung 2.7.: Support Vector Machine aus Christopher D. Manning, Raghavan und Schütze (2008, S. 320).

Abbildung 2.7 illustriert das SVM-Verfahren für den linear separierbaren Fall. Die im n -dimensionalen Vektorraum geplotteten Datenpunkte zweier Klassen (Dreieck und Punkt) werden durch die zwischen den Stützvektoren (sog. *Support Vectors*) aufgespannte Entscheidungsebene getrennt. Je weiter der Radius (*margin*) um die Hyperebene, desto sicherer ist die Klassifikation eines Datenpunkts.

Naïve Bayes

Naïve Bayes (NB) ist ein statistischer Klassifikator, welcher auf dem Satz von Bayes fundiert. Naïve Bayes bezeichnet eine Familie von Algorithmen. Hier ist genauer der *Multinomial Naïve Bayes* gemeint.

Der Satz von Bayes berechnet für ein zu klassifizierendes Dokument \mathbf{d} und die Wahrscheinlichkeit in Klasse c zu sein wie folgt:

$$P(c \mid \mathbf{d}) = \frac{P(c) \cdot P(\mathbf{d} \mid c)}{P(\mathbf{d})},$$

wobei \mathbf{d} durch einen Vektor (x_1, \dots, x_n) bestehend aus n *features* repräsentiert wird und c eine mögliche Klasse der vordefinierten Klassen C ist.

$$P(c \mid \mathbf{d}) = P(c \mid x_1, \dots, x_n)$$

Ein *feature* x_i , mit $1 \leq i \leq n$ könnte etwa die Wahrscheinlichkeit des Vorkommens eines bestimmten Wortes im Dokument sein (Christopher D. Manning, Raghavan und Schütze 2008). *Naïve* wird der Klassifikator durch die Annahme genannt, dass die *features* unabhängig voneinander eintreten. Aus der naiven Annahme folgt:

$$P(c \mid x_1, \dots, x_n) = P(c \mid \mathbf{d}) = \frac{P(c) \cdot \prod_{i=1}^n P(x_i \mid c)}{P(x_1) \cdot P(x_2) \cdot \dots \cdot P(x_n)}$$

Das Ziel in der Textklassifikation ist, die beste Klasse für ein Dokument zu finden (Christopher D. Manning, Raghavan und Schütze 2008).

$$c_{best} = \operatorname{argmax}_{c \in C} P(c \mid \mathbf{d})$$

2.3.3. Evaluationsmetriken

Die Evaluation ist der letzte Schritt der Textklassifikationspipeline und dient dem Aufbauen eines Verständnisses dafür, wie gut die Vorhersagen des Modells sind (Kowsari u. a. 2019). Im Folgenden werden die in dieser Arbeit genutzten Evaluationsmetriken zur Bewertung der Güte der binären Klassifikation vorgestellt.

Accuracy gibt den Anteil der korrekten Vorhersagen unter allen Vorhersagen des Modells an.

$$\text{Accuracy} = \frac{\text{Anzahl der korrekten Vorhersagen}}{\text{Anzahl aller Vorhersagen}}$$

Das Maß beschreibt allerdings nur bei etwa gleich großen Klassen zuverlässig die Güte des Modells. Bei stark unausgeglichenen Datensätzen könnte das Modell lernen, immer die übermäßig vorhandene Klasse vorherzusagen. Auch wenn es nicht ein einziges Mal die andere Klasse richtig vorhersagt, ist die Accuracy hoch.

Deshalb wird zusätzlich der **F_1 -Score** verwendet. Er bemisst sich aus *Precision* und *Recall*.

$$\text{Precision} = \frac{\text{Anzahl korrekter positiver Vorhersagen}}{\text{Anzahl aller positiven Vorhersagen}}$$

$$\text{Recall} = \frac{\text{Anzahl korrekter positiver Vorhersagen}}{\text{Anzahl aller Vorkommen der positiven Klasse im Datensatz}}$$

Das harmonische Mittel von Precision und Recall ergibt den F_1 -Score.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

3. Forschungsstand

Es konnte keine Arbeit gefunden werden, die ein in allen Aspekten vergleichbares Thema wie diese Arbeit behandelt. Mit einzelnen Aspekten dieser Arbeit in Verbindung stehende Forschungsliteratur wird im Folgenden dargestellt.

3.1. Nutzung von Wikipedia für die Generierung von Trainingsdaten

Suissa, Elmalech und Zhitomirsky-Geffet (2022) beschreiben zwei Herausforderungen, auf die Wissenschaftler im Bereich der digitalen Geisteswissenschaften¹ (DH) treffen, wenn sie *supervised Machine Learning* und speziell das *supervised Deep Learning* in ihrer Forschung nutzen wollen. Zum einen sei dies die Nicht-Verfügbarkeit von angemessenen Trainingsdaten. Entweder seien die Textressourcen zu spezifisch, und daher nicht ausreichend verfügbar, oder aber die Klassen der Trainingsdaten seien nicht ausgewogen verteilt. Zum anderen sei das *Pre-Processing* der Daten im Bereich der digitalen Geisteswissenschaften notwendig, da die Textressourcen üblicherweise *noisy*, aber auch uneinheitlich in ihrem Format seien. Da große, frei verfügbare und bereits annotierte Datensätze bislang nicht von der DH-Community existierten, sei die beste Alternative, um Datensätze zu erstellen, immer noch der Mensch (Suissa, Elmalech und Zhitomirsky-Geffet 2022).

Wikipedia als Wissensbasis für semantische Informationen zu nutzen, ist keine neue Idee (s. bspw. Strube und Ponzetto 2006). Inwiefern der Wikipedia Korpus für die multilinguale Analyse genutzt werden kann, untersuchen Adafre und De Rijke (2006). Die Autoren erhoffen sich vom Alignment ähnlicher Sätze in zwei Sprachen Einsicht darüber zu erlangen, wie bestimmte Themen in der Wikipedia in den verschiedenen Sprachen dokumentiert sind. Es sei möglich, vergleichbare Datensätze in verschiedenen Sprachen über die Linkstruktur der Wikipedia zu generieren, da Wikipedia-Artikel dasselbe Konzept beschreiben, auch wenn sie meist keine genauen Übersetzungen voneinander sind. Ihre Forschung betont, dass die Wikipedia durch ihre Linkstruktur ein großes Potenzial für die Generierung paralleler Korpora birgt (Adafre und De Rijke 2006).

Aufwändig vortrainierte und frei zur Verfügung stehende Word Embedding Modelle, zum Beispiel von Pennington, Socher und Christopher D Manning (2014) und Mikolov, Grave u. a. (2018), sowie state-of-the-art Sprachmodelle (Devlin u. a. 2019) verwendeten (neben weiteren Korpora) den Wikipedia-Korpus für das *Pre-Training*.

¹engl. *Digital Humanities*; Der Begriff beschreibt das interdisziplinäre Aufeinandertreffen des weiten Feldes der Geisteswissenschaften (somit auch der Musikwissenschaft, und insbesondere der Linguistik) mit der Informatik. Für eine ausführliche Definition siehe Sahle (2011).

3.2. Vergleichsstudien zur multilingualen Textklassifikation

Mutuvi u. a. (2020) vergleichen für die multilinguale Textklassifikation von Nachrichten zu Krankheitsausbrüchen verschiedene Machine Learning Modelle auf TF-IDF Wort-Gewichten mit multilingualen Deep Learning Modellen. Die Autoren beobachten, dass die Textklassifikation mit Machine Learning Klassifikatoren sehr unausgeglichene Ergebnisse produziert, mit den höchsten *Precision*-Werten und den niedrigsten *Recall*-Werten. Das Deep-Learning Sprachmodell BERT resultiert in stabileren Werten.

Jiang u. a. (2019) nutzen in einem sprachübergreifenden Vektorraum alignierte Word Embeddings in den Sprachen Englisch und Französisch und führen auf diesen Embeddings die Textklassifikation mit zwei verschiedenen Klassifikatoren, welche auf neuronalen Netzen basieren, aus. Sie beobachten, dass sowohl die monolingualen, als auch die bilingualen Klassifikationen vom sprachübergreifenden Vektorraum profitieren (Jiang u. a. 2019).

Eine kürzlich erschienene Studie von Velankar, Patil und Joshi (2022) untersucht auf den Tasks der Textklassifikation, der Erkennung von Hassrede und der Sentimentanalyse, ob monolinguale (in Marathi) oder multilinguale vortrainierte auf BERT basierende Modelle besser abschneiden. Unter den untersuchten multilingualen Modellen befindet sich auch das in dieser Arbeit verwendete mBERT (Devlin u. a. 2019). Die Autoren beobachten, dass die monolingualen BERT-Modelle auf allen Datensätzen besser als die monolingualen BERT-Modelle abschneiden.

3.3. Kombination von transformer-basiertem Topic Modeling und Word Embeddings für die Textklassifikation

Alhaj u. a. (2022) nutzen den Output von BERTopic, um Textrepräsentationen von Tweets anzureichern. Zunächst bilden sie eine Textrepräsentation Embedding(doc_d) mit dem Durchschnitt der einzelnen Word2Vec-Embeddings aller Wörter im Dokument. Dann generieren sie die Verteilung von latenten Topics BERTTopic(doc_d) im Dokument d . Die Topic Repräsentation und die Word Embeddings eines Dokumentes werden nun konkateniert. Die so erhaltene Dokumentrepräsentation (CTE) wird schließlich in ein Machine Learning Modell für die Textklassifikation übergeben. Das Vorgehen wird in Abbildung 3.1 veranschaulicht.

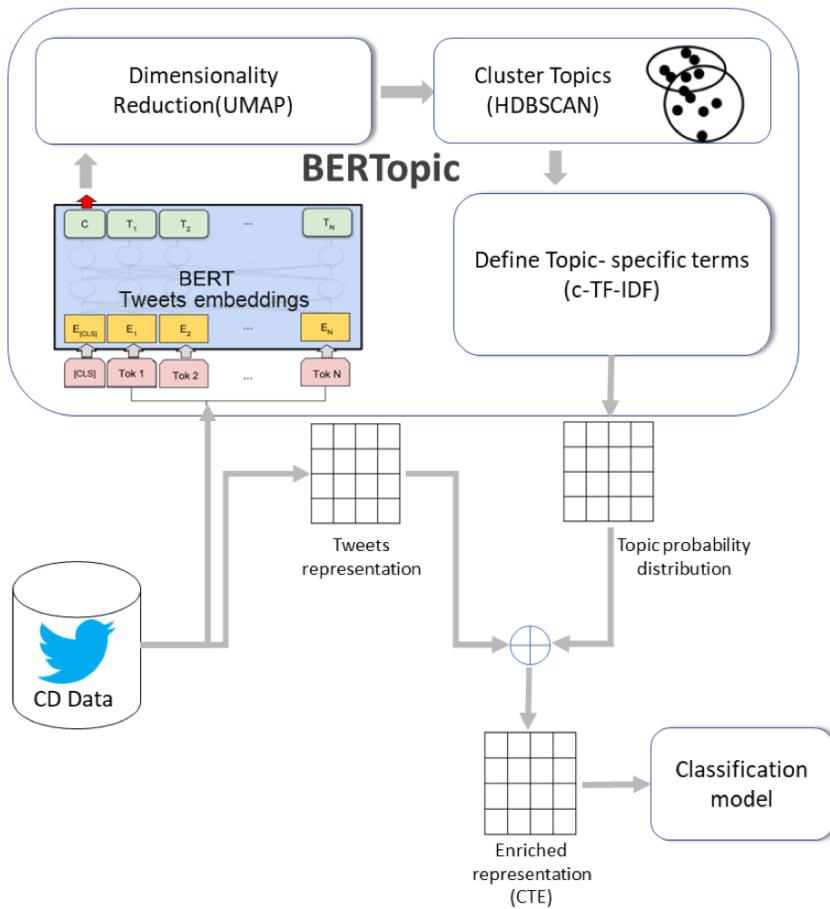


Abbildung 3.1.: Illustration der Anreicherung von Textrepräsentationen mit Topic Modeling aus Alhaj u. a. (2022, S. 4).

Schließlich vergleichen die Autoren die Performanz verschiedener Klassifikatoren auf entweder der nur auf Word2Vec basierenden Tweet-Repräsentation, oder der angereicherten Tweet-Repräsentation aus. Der SVM Klassifikator konnte hierbei (neben der Machine Learning Methode des *Stacking*) besonders gut abschneiden.

4. Exkurs zur Musikgeschichte: Komponisten im Nationalsozialismus

Der geschichtliche Hintergrund für die beiden Klassen, die vom Textklassifikationsmodell unterschieden werden sollen, ist einerseits das Verbot von als „entartet“ diffamierter Musik im Dritten Reich und andererseits die Förderung und Entlastung bestimmter Personen, die der politischen und ästhetischen Agenda der Nationalsozialisten besser entsprachen.

Der Begriff der „Entarteten Musik“ lässt sich auf die gleichnamige Ausstellung im Rahmen der Reichsmusiktage 1938 in Düsseldorf zurückführen, die von Hans Severus Ziegler in Anlehnung an Ausstellung „Entartete Kunst“ aus dem Jahre 1937 initiiert wurde.

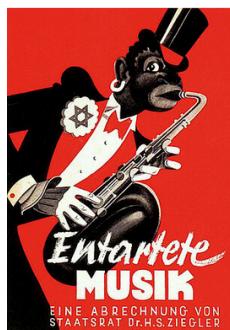


Abbildung 4.1.: Titelbild der 1939 erschienenen Broschüre zur Ausstellung „Entartete Musik“,
<https://holocaustmusic.org/de/politics-and-propaganda/third-reich/>,
letzter Zugriff: 12.11.2022 (Screenshot vom 12.11.2022, 21:45 Uhr).

Auf dem Titelbild der Broschüre zur Ausstellung (s. Abb. 4.1) wird die Hauptfigur der Oper „Johnny spielt auf“ von Ernst Krenek karikiert dargestellt und die Jazzmusik als ein Produkt der Vermischung unterschiedlicher Kulturen dargestellt, was dem Gebot der Rassenreinheit der Nationalsozialisten widersprach (Dümling 2015).

Neben der Jazzmusik wurde die Musik jüdischer Komponisten und atonale Musik abgewertet. In seiner Rede zur Eröffnung der Ausstellung sprach Ziegler:

Ich bekenne mich mit einer Reihe führender musikalischer Fachmänner und Kulturpolitiker zu der Anschauung, dass die Atonalität als Ergebnis der Zerstörung der Tonalität Entartung und Kunstbolschewismus bedeutet. Da die Atonalität zudem ihre Grundlage in der Harmonielehre des Juden Arnold Schönberg hat, so erkläre ich sie für das Produkt jüdischen Geistes. [...] Wer in die Schule Beethovens geht, kann unmöglich über die Schwelle der Werkstatt Schönbergs finden, wer sich aber länger in dieser Werkstatt Schönbergs aufgehalten hat, verliert notwendigerweise das Gefühl für die Reinheit Beethovens.

Aus: Dümling (2015, Rede Hans Severus Zieglers vom 25. Mai 1938, auf CD bereitgestellt vom Deutschen Rundfunkarchiv)

Schon etwa seit März 1933 begann Joseph Goebbels den Kunst- und Kulturbetrieb gleichzuschalten. Die bekannten Komponisten Arnold Schönberg und Franz Schreker wurden im selben Jahr von der Preußischen Akademie der Künste ausgeschlossen. Die Säuberung der Musikszene führte dazu, dass bis Ende 1935 viele jüdische Musiker und Komponisten ins Exil getrieben wurden und meist in den USA, oder in Palästina einen Neuanfang wagten.

Auf der anderen Seite gab es auch Profiteure des NS-Regimes, wie Carl Orff. Sein bis heute in den Opernhäusern präsentes Werk „Carmina Burana“ ist innerhalb des NS-Regimes entstanden und 1937 in Frankfurt am Main uraufgeführt worden.

Jene als „gottbegnadet“ geltende Komponisten, wurden aufgrund ihrer angeblich hohen Bedeutung für die deutsche Identität vom Kriegsdienst befreit.

Auch das Andenken bereits verstorbener Komponisten wurde angetastet. Beispielsweise konnten die Bayreuther Festspiele, welche von Richard Wagner begründet wurden, vom Nationalsozialismus profitieren, während Stücke des Komponisten Felix Mendelssohn Bartholdy aus den Konzertsälen verschwanden.

5. Methodik

5.1. Klassifikation von Texten eines geisteswissenschaftlichen Themas

Für diese Bachelorarbeit kann das von Suissa, Elmalech und Zhitomirsky-Geffet (2022) genannte Problem der Nicht-Verfügbarkeit von bereits annotierten fachspezifischen Korpora bestätigt werden (s. Kapitel 3.1). Daher werden die Annotationen über die von Nutzern durchgeführten Kategorisierungen von Wikipedia-Konzepten und deren Sammlung zu speziellen Listen generiert. Die über diesen Weg vorgenommene Einteilung der Komponisten in die zwei Klassen „begnadet“ und „verfolgt“ bedeutet eine starke Vereinfachung, welche allerdings für das Machine Learning notwendig ist. Es werden in dieser Kategorisierung die vielfältigen Schattierungen innerhalb dieser Klassen außer Acht gelassen, seien die Komponisten Opportunisten, Profiteure oder Mittäter, sowie auf der anderen Seite Ausgeschlossene, Verfolgte oder Ermordete gewesen.

Kirschenbaum (2007) stellt die gegenteiligen Kulturen der Geisteswissenschaften und der Informatik heraus. Die Geisteswissenschaften pflegten eine von Interpretation, Argumentation und Mehrdeutigkeit geprägte Kommunikationskultur, die eine *ground truth* kaum akzeptieren könne und weniger auf das Problemlösen fokussiert sei (vgl. Kirschenbaum 2007, S. 1). Die Textklassifikation durch eine Maschine könne in den digitalen Geisteswissenschaften dafür genutzt werden, um neue, unerwartete Erkenntnisse über bekannte Texte zu erlangen und nicht in erster Linie, um einen Experten durch eine Maschine zu ersetzen (Kirschenbaum 2007). Es wird daher auch für diese Arbeit angenommen, dass die radikale binäre Klassifikation der Komponisten zu interessanten Ergebnissen führen kann.

5.2. Wikipedia als Datenquelle

Ausschlaggebend für die Entscheidung Wikipedia als Datenquelle zu nutzen ist ihre Mehrsprachigkeit, die es ermöglicht vergleichbare Textkorpora in verschiedenen Sprachen zu erhalten (Adafre und De Rijke 2006). Die Wikipedia ist seit ihrer Gründung im Jahr 2001 eine rasant wachsende und einzigartige Basis für frei zugängliches Wissen und wird damit ihrem Namen gerecht, der als Schachtelwort das Wort „Wiki“ (hawaiisch für „schnell“) mit dem englischen Wort „encyclopedia“ (Wikipedia 2022d) kombiniert. Für den Bereich der Naturalen Sprachverarbeitung in der Informatik können Verlinkungen zwischen Wikipedia-Artikeln und die im Jahr 2004 eingeführten Kategorien zum Beispiel für die Korpuserstellung hilfreich sein. Dabei werden die Kategorienzuordnungen in dieser Bachelorarbeit als Annotationen der Seiten aufgefasst.

Durch die kollaborative Partizipation mehrerer Autoren an Wikipedia-Artikeln wird bei den annotierten Daten vom *Goldstandard* ausgegangen, d.h. von manuell annotierten Daten, de-

ren Annotation überprüft wurde und gegen die die automatische Klassifikation des Systems gegengeprüft wird (Bird, Klein und Loper 2009). Durch diese Annahme entfällt der aufwändige Schritt der selbst durchgeführten Annotation der Daten, die je nach Domäne, bestimmtes Expertenwissen (hier Wissen im Bereich der Musikgeschichte) voraussetzt.

Spätestens beim Schritt der Interpretation der Ergebnisse werden allerdings jene Kenntnisse wieder benötigt. Das genaue Vorgehen des Wikipedia Scrapings, sowie die betrachteten Seiten werden in Kapitel 6.2.1 beschrieben.

5.3. Textklassifikation mit Word Embeddings und Machine Learning Methoden

Die Textklassifikation mit statischen Word Embeddings und Machine Learning Algorithmen orientiert sich an der von Kowsari u. a. (2019) illustrierten Pipeline, deren Schritte im Kapitel 2.3 des Grundlagenteils vorgestellt wurden.

5.3.1. Wahl der Features

Es stellte sich die Frage, mit welcher Technik die Features für die Machine Learning Modelle generiert werden sollen. Die Wahl fiel auf Word Embeddings (s. Kapitel 2.1.2), da sie in der Literatur schon lange etabliert sind. Die Vorauswahl bestimmter Wörter hätte dazu geführt, dass der Vorteil der Objektivität von automatischen Methoden gegenüber musikwissenschaftlicher Expertenarbeit nivelliert wäre. Daher wurden die Wortvektoren aller Wörter in den Texten verwendet.

Ein Dokumentvektor repräsentiert dann den Text eines Dokumentes aus den einzelnen Wortvektoren, die im Vokabular des Embedding Modells nachgeschlagen werden und wird als Feature den Klassifikationsalgorithmen übergeben. Hier als *mean representation vector* definiert, berechnet sich der Dokumentvektor aus dem Durchschnitt aller Wortvektoren im Dokument.

Die gewählten Techniken zur Generierung von Words Embeddings waren Word2Vec und fastText. Es wurde mit verschiedenen Ansätzen Word Embeddings zu erhalten experimentiert. Der Ansatz die Word Embeddings selbst zu trainieren wurde verfolgt, um speziell auf die Texte angepasste Embeddings zu erhalten. Der Suchbegriff „Schönberg“ findet in Wikipedia beispielsweise etwa 80 Orte und sehr viele Namensträger. Wir möchten jedoch einen Wortvektor für den Komponisten (Arnold) Schönberg erhalten. Im Grundlagenteil wurde gezeigt, dass statische Word Embeddings nur einen Wortvektor für ein Wort berechnen können. Im genannten Beispiel könnte der Wortvektor für „Schönberg“ somit für mehrere Orte, Berge, oder Personen gleichzeitig stehen.

Allerdings ist der Datensatz für diese Bachelorarbeit sehr klein (s. Kapitel 6.2.3). Daher werden die selbst trainierten Embeddings mit vortrainierten Embeddings für die Textklassifikation verglichen.

Word2Vec

Als Baseline wurden Word2Vec Modelle *from-scratch*, also auf dem eigenen Datensatz von Grund auf neu trainiert. Da keine sprachübergreifenden vortrainierten Word2Vec Embeddings in Deutsch und Englisch gefunden werden konnten, wurden auch für den monolingualen Fall keine vortrainierten Word2Vec Embeddings genutzt.

Es wurden aufgrund der kleinen Größe des Datensatzes 100-dimensionale Word2Vec-Embeddings generiert. Im Vergleich dazu haben vortrainierte Word Embeddings meist 300 Dimensionen.

fastText

Es wurden 300-dimensionale fastText-Embeddings auf den eigenen Daten trainiert, da die MUSE Bibliothek (Conneau u. a. 2017), die später für das Alignment verwendet wurde, ebenso diese Dimensionen verwendet. Für den Kontext des Wortes wurden immer die fünf Wörter vor und nach dem Wort betrachtet. Es wurden Word Embeddings für alle Wörter im Korpus erstellt, die mindestens zweimal vorkamen.

5.3.2. Alignment der fastText-Embeddings

Abbildung 5.1 illustriert das Vorgehen für das Alignment der monolingualen fastText-Embeddings. Das Alignment von fastText-Embeddings mit MUSE wurde im Grundlagenteil beschrieben (s. 2.1.2). Das Vorgehen orientiert sich an dem Vorgehen von Jiang u. a. (2019). Die Autoren haben zum Vergleich auch die Methode der Machine Translation verwendet, die aber in dieser Bachelorarbeit nicht angewendet wurde.

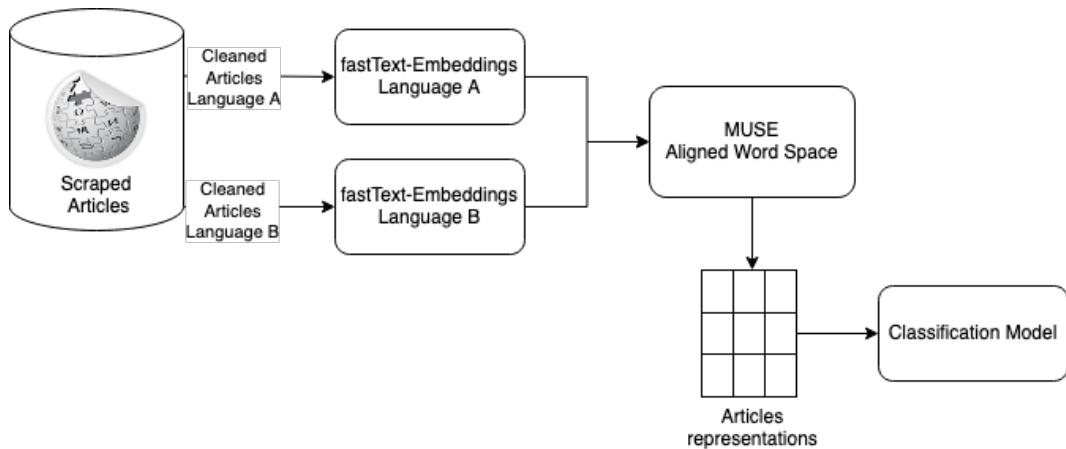


Abbildung 5.1.: Alignment der Word Embeddings, eigene Darstellung, angelehnt an Jiang u. a. (2019, S. 2).

5.3.3. Anreicherung der Word Embeddings durch Topic Modeling

Angelehnt an die Methode von Alhaj u. a. (2022), wurden in dieser Arbeit für jeden Artikel die n=3 wahrscheinlichsten Themenwörter des Artikels generiert. Da das BERTopic-

Modell auf Sätzen trainiert wurde (s. Kapitel 6.3.3), musste ein Mapping von den Satz-Thema-Zuordnungen auf den Artikel stattfinden. Die so erhaltenen Topics mit ihren jeweils 10 Themenwörtern wurden dem bereinigten Text angehängt, damit neue Features zu generieren. Die Methode wird in Abbildung 5.2 beschrieben.

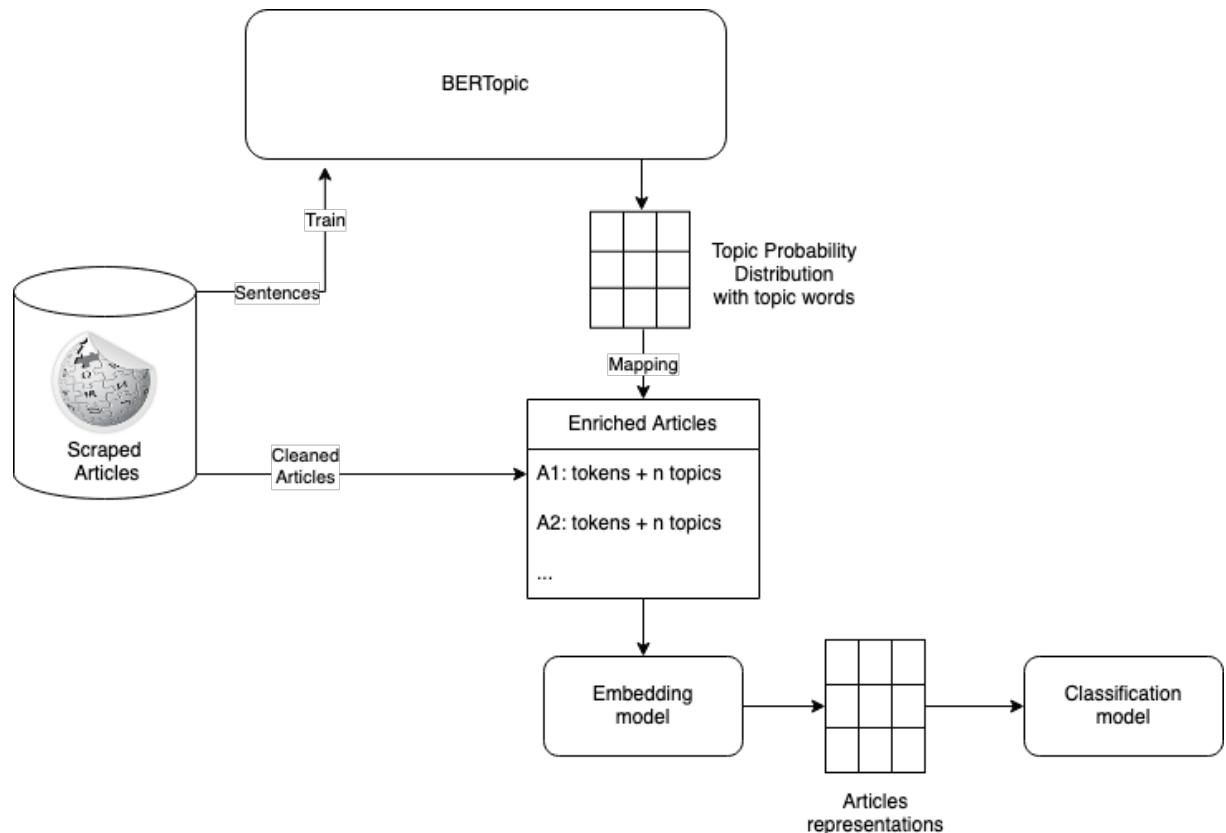


Abbildung 5.2.: Anreicherung von Texten mit Topic Modeling, eigene Darstellung, angelehnt an Alhaj u. a. (2022, S. 4).

5.3.4. Wahl der Klassifikationsverfahren

Ein wichtiger Punkt zur Wahl des richtigen Klassifikationsverfahrens ist die Frage, wieviel Daten vorhanden sind (Christopher D. Manning, Raghavan und Schütze 2008). Generell sei aus der Praxis die Daumenregel entstanden, dass mit einer Verdopplung der Menge der Trainingsdaten die Performanz der Klassifikationsverfahren linear ansteige (Christopher D. Manning, Raghavan und Schütze 2008). Da eine Korpuserweiterung um vergleichbare biografische Texte von Komponisten in verschiedenen Sprachen für diese Bachelorarbeit nicht möglich ist, kann auch die Zahl der Trainingsdaten auf diesem Weg nicht erhöht werden.

Naïve Bayes

Der im Kapitel 2.3.2 beschriebene Naïve Bayes Klassifikator wird in unserem Fall auf die Klassen $C = \text{"begnadet", "verfolgt"}$ angewendet. Das Problem des Findens der besten Klasse c_{best} für ein gegebenes Dokument d kann daher in folgender Weise formuliert werden:

$$c_{best} = \operatorname{argmax}_{c \in \{\text{"begnadet", "verfolgt"}\}} P(c | d)$$

Dabei werden die Wahrscheinlichkeiten $P(c)$ und $P(c | d)$ aus den Häufigkeiten im Trainingsdatensatz geschätzt.

Seine vereinfachten Annahmen (s. Kap. 2.3.2) machen Naïve Bayes zu einem effizienten Klassifikator (Kowsari u. a. 2019). Jedoch beschreiben Ikonomakis, Kotsiantis und Tampakas (2005) ihn als nicht ideal für die Modellierung von Texten, da er die Relationen zwischen den Features ignoriert. Sprachen werden somit als *bags-of-words* behandelt. Naïve Bayes wird aufgrund seines Einsatzes als Baseline in vielen Studien zur Textklassifikation angewendet (Ikonomakis, Kotsiantis und Tampakas 2005).

Support Vector Machine

Ikonomakis, Kotsiantis und Tampakas (2005) zufolge erzielt der SVM-Klassifikator für Texte eine sehr gute Precision, aber einen schlechten Recall. Nach Christopher D. Manning, Raghavan und Schütze (2008) eigne er sich besonders für Probleme mit einer geringen Menge an Trainingsdaten. Daher wurde er für diese Bachelorarbeit verwendet.

Oversampling von nicht-balancierten Trainingsdaten

Unausgeglichene Datensätze (*imbalanced data sets*) können für das Machine Learning problematisch sein. Einen Lösungsansatz bietet das *Resampling* von Datensätzen, welches auf das quantitative Angleichen der Klassen ausgerichtet ist.

Neben dem *Oversampling* zählt das *Undersampling* zum Resampling. Das Resampling der Daten für diese Arbeit hat experimentellen Charakter. Das Ziel war, ein Gefühl über das Potenzial des Resampling auf einem kleinen Datensatz zu erlangen. Daher wurde auf einen Einsatz und Vergleich verschiedener komplexer Techniken verzichtet und einem naiven Ansatz gefolgt.

Es wird die *Random Oversampling* Technik genutzt, bei der die Anzahl der Datensätze in der kleineren Klasse an die der größeren Klasse angeglichen wird, indem zufällige Datensätze (*samples*) dupliziert werden. Ebenso wird ihr Gegenstück das *Random Undersampling* implementiert, welches die Klassengrößen durch Reduzierung der Trainingsdaten der größeren Klasse angleicht.

Wo beim Undersampling die Gefahr des Informationsverlusts besteht, kann beim Oversampling das Hinzufügen von redundanten Informationen ein Problem darstellen (Leevy u. a. 2018).

5.4. Wahl der vortrainierten BERT Modelle

Wegen der geringen Anzahl an Trainingsdaten wurde der Ansatz gewählt state-of-the-art vortrainierte Sprachmodelle (PLMs) zu verwenden und Finetuning auf der Task der Textklassifikation zu betreiben.

Alle der verwendeten Modelle basieren auf der von Devlin u. a. (2019) vorgestellten BERT-Base Architektur.

Tabelle 5.1.: Das BERT-Base Modell.

| Layer | Hidden | Attention Heads | Parameter | Techniken für das Training |
|-------|--------|-----------------|-----------|----------------------------|
| 12 | 768 | 12 | 110M | NSP und MLM |

Es wurden stets Varianten gewählt, welche die Unterschiede zwischen Groß- und Klein-schreibung beachten. Leider ist BERT nur in der Lage Sequenzen von maximal 512 Token (in Englisch etwa 300-400 Wörter) einzulesen, wobei längere Sequenzen gekürzt werden. Kapitel 6.2.3 zeigt allerdings, dass die durchschnittliche Wortzahl eines extrahierten Artikels länger ist.

Es wurde dennoch entschieden, das Modell auf Artikelebene, und nicht auf Paragraph- oder gar Satzebene zu trainieren, da die Forschungsfrage dieser Arbeit nach den Klassenzuordnungen der Komponisten fragt.

5.4.1. Die verwendeten Modelle

- *bert-base-cased*¹ (im Weiteren frei mit eBERT abgekürzt)
- *bert-base-multilingual-cased*² (mBERT)
- *bert-base-german-cased*³ (gBERT)

Das für das Finetuning auf dem englischen Datensatz verwendete eBERT wurde neben einem zweiten Korpus auf der bereinigten englischen Wikipedia vortrainiert, wobei Tabellen, Listen und Referenzen der Artikel gefiltert wurden (Devlin u. a. 2019). Dieses Vorgehen wird auch in dieser Bachelorarbeit im Kapitel 6.2.1 gewählt.

Das multilinguale mBERT wurde auf der konkatenierten Wikipedia in 104 Sprachen vortrainiert, darunter sind auch die Sprachen Deutsch und Englisch (Devlin 2019).

Das deutsche Sprachmodell gBERT wurde von Chan u. a. (2019) auf dem deutschen Wikipedia Dump trainiert. Die Entwickler beschreiben ihre Motivation in der unzureichenden Performanz des multilingualen mBERT auf deutschen Texten (s. Abb. 5.3). Das Modell wurde dann auf 5 verschiedenen deutschen Datensätzen und den Downstream Tasks *Multiclass*

¹<https://huggingface.co/bert-base-cased> [Letzter Zugriff am 27.11.2022 um 13:02 Uhr]

²<https://huggingface.co/bert-base-multilingual-cased> [Letzter Zugriff am 27.11.2022 um 13:02 Uhr]

³<https://huggingface.co/bert-base-german-cased> [Letzter Zugriff am 27.11.2022 um 13:04 Uhr]

Classification, Binary Classification, Named Entity Recognition und Document Classification evaluiert und mit dem mBERT (s.u.) verglichen (Chan u. a. 2019), wobei es in allen Anwendungen eine bessere Performance als mBERT erreichte.

| | | |
|---------|-------------|---|
| uncased | arzt ##e | hatten unter anderem eine stunde sonne ##nl ##icht pro tag em ##pf ##oh ##len |
| cased | Ä ##rz ##te | hätten unter anderem eine Stunde Sonne ##nl ##icht pro Tag em ##pf ##oh ##len |
| German | Ärzte | hätten unter anderem eine Stunde Sonnen ##licht pro Tag empfohlen |

Abbildung 5.3.: Limitationen des multilingualen BERT Modells auf einer deutschen Sequenz, <https://www.deepset.ai/german-bert>, letzter Zugriff: 23.11.2022 (Screenshot vom 23.11.2022, 14:54 Uhr).

5.4.2. Methodik für das Finetuning

Devlin u. a. (2019) beschreiben welche Hyperparameter für das Finetuning angepasst werden können. Diese sind Batch-Size, Lernrate und Epochenanzahl. Für das Finetuning werden folgende etablierte Werte vorgeschlagen, welche allerdings für verschiedene Tasks variieren können (vgl. Devlin u. a. 2019, S. 13):

- Batchgröße: 16, 32
- Lernrate des standardmäßigen *Adam*-Optimierers: 5e-5, 3e-5, 2e-5
- Epochenzahl: 2, 3, 4

Allerdings betonen die Autoren, dass vor allem bei kleinen Datensätzen die Hyperparameterwahl sehr große Unterschiede bewirken können und daher viele Kombinationen ausprobiert werden sollten. Der genaue Ablauf des Finetunings wird in Kapitel 6.4 beschrieben.

6. Umsetzung

Zunächst werden die in der Implementierung benutzten Frameworks und Bibliotheken im Abschnitt 6.1 beschrieben. Der Abschnitt 6.2 befasst sich mit allen Operationen, die zur Vorbereitung des Korpus vorgenommen wurden, beginnend mit der Beschaffung der Texte aus der Wikipedia. Der Aufbau des Kapitels der Textklassifikation mit statischen Embeddings orientiert sich an den Schritten der Pipeline zur Textklassifikation (s. Kapitel 2.3) von Kowsari u. a. (2019). Danach folgt die Textklassifikation mit BERT.

6.1. Verwendete Technologien

Der Code ist in *Python* (V. 3.9) geschrieben, welches über die Open-Source-Distribution *Anaconda* in der Version 1.9. installiert wurde. Die für diese Arbeit genutzte Entwicklungsumgebung war DataSpell. DataSpell wurde aufgrund seiner Vorteile für die Entwicklung genutzt, darunter Code Completion und Type Hinting. In der browserbasierten Webapplikation *Jupyter Notebooks*, welche im Anaconda-Paket enthalten ist, lassen sich die einzelnen in der Bachelorarbeit vorgenommenen Durchläufe durch angepasste Parameter in den Notebooks steuern, was für die Wiederverwendbarkeit des Codes, aber auch für Code-Demonstrationen vorteilhaft ist. Da das verwendete Betriebssystem MacOs Probleme mit dem Import von den genutzten *spacy* Sprachmodellen aufwurf, wurde Docker genutzt.

Es wurde ein öffentliches GitHub-Repository für den Programmcode angelegt (Hassibi 2022). MediaWiki stellt eine API¹ für Wikipedia für den freien Zugriff auf Wikipedia-Inhalte bereit. Diese wurde mit dem Wrapper *Wikipedia-API* (Majlis 2017) angesprochen.

Pandas (McKinney 2010) erleichterte mit der Datenstruktur *DataFrames* die Operationen auf den Datensätzen, sowie das Lesen und Schreiben der Daten in CSV-Dateien.

Des Weiteren wurden *matplotlib* (Hunter 2007) und *NumPy* (Harris u. a. 2020) für die Analyse und Visualisierung der Datensätze genutzt.

Die für Tokenisierung und Lemmatisierung genutzte Python Bibliothek war *SpaCy* (Honnibal und Montani 2017). Mit SpaCy muss zuerst die Pipelines der jeweiligen Sprachen heruntergeladen und installiert werden. Für Deutsch wurde *de_core_news_sm*² und für Englisch *en_core_web_sm*³ genutzt.

Die Stopwortlisten für Englisch und Deutsch wurden für diese Arbeit mit *NLTK* (Bird, Klein und Loper 2009) geladen. Der Name steht für *Natural Language Processing Toolkit* und für die führende Plattform für Text Processing Bibliotheken und Korpora.

Das Topic Modeling wurde mit *BERTopic* (Grootendorst 2022) durchgeführt.

Die Implementierung und das Training der fastText-Modelle erfolgte mit *gensim* (Rehurek und Sojka 2011). Das Vector Alignment der fastText-Embeddings wurde mit *MUSE* (Conneau

¹https://www.mediawiki.org/wiki/API:Main_page [Letzter Zugriff am 3. November 2022 um 9:00 Uhr]

²<https://spacy.io/models/de> [Letzter Zugriff am 3. November 2022 um 9:15 Uhr]

³<https://spacy.io/models/en> [Letzter Zugriff am 3. November 2022 um 9:15 Uhr]

u. a. 2017) realisiert.

Die vortrainierten BERT Modelle und die zu ihnen passenden Tokenisierer mit der *Huggingface* (Wolf u. a. 2019) Bibliothek geladen.

Das Finetuning fand auf einer Tesla T4 GPU in *Google Colab* (Google LLC 2020) statt.

Mit *scikit-learn* (Pedregosa u. a. 2011) wurden die SVM- und NB-Klassifikatoren geladen und die Evaluierungen der Textklassifikationen durchgeführt.

Imbalanced-learn (Lemaître, Nogueira und Aridas 2017) ist eine Open Source Python Toolbox, die basierend auf *scikit-learn* Lösungen für die Klassifikation auf unausgeglichenen Datensätzen implementiert und hier das Oversampling ermöglichte.

6.2. Daten

6.2.1. Wikipedia Scraping

Für die Korpuserstellung wurden zwei deutsche Wikipedia-Seiten betrachtet, welche Auflistungen von Personen beinhalten.

Für die Texte, die über die „Gottbegnadeten-Liste“ (Wikipedia 2022b) erreicht wurden, wurde das Label „begnadet“ (abgekürzt für „gottbegnadet“) übernommen. Die Texte der „Liste der vom NS-Regime oder seinen Verbündeten verfolgten Komponisten“ (Wikipedia 2022c) bekamen das Label „verfolgt“.

Über die Linkstruktur und das Kategoriensystem von Wikipedia konnten die relevanten Seiten extrahiert werden.

The screenshot shows a Wikipedia page titled "Liste der vom NS-Regime oder seinen Verbündeten verfolgten Komponisten". The page content discusses musicians persecuted by the Nazi regime. It includes a quote from Sarah Nathan-Davis and a list of names. The sidebar on the left contains links to various Wikipedia sections like Hauptseite, Artikel verbessern, and Spaltenprojekte.

Inhaltsverzeichnis [Verbergen]

- 1 Von den Nationalsozialisten oder deren Verbündeten ermordete Komponisten
- 2 Von den Nationalsozialisten verfolgte oder ins Exil gezwungene Komponisten
- 3 Siehe auch
- 4 Weblinks

Von den Nationalsozialisten oder deren Verbündeten ermordete Komponisten

| | | | |
|--------------------------------------|----------------------------------|---------------------------------|-------------|
| • Heinz Alt (1922–1945) | • Gustav Ernest (1858–1941) | • Rudolf Karel (1880–1945) | • Valen... |
| • Richard Altmann (1888–1942) | • Ralph Erwin (1896–1943) | • Dick Kattenburg (1919–1944) | • Ruth I... |
| • Ernst Bachrich (1892–1942) | • Richard Fall (1882–1945) | • Franz Eugen Klein (1912–1944) | • Nico F... |
| • Elkan Bauer (1852–1942) | • Siegfried Fall (1877–1943) | • Gideon Klein (1919–1945) | • Andri... |
| • Emil Bauer (Komponist) (1874–1941) | • Mordechaj Gebirtig (1877–1942) | • Josef Koffler (1896–1944) | • Samu... |
| • David Beigelman (1887–1945) | • Sim Gokkes (1897–1943) | • Viktor Kohn (1901–1944) | • Zikmu... |

Abbildung 6.1.: Ausschnitt der „Liste der vom NS-Regime oder seinen Verbündeten verfolgten Komponisten“, aus: Wikipedia (2022c), letzter Zugriff: 10.11.2022 (Screenshot vom 10.11.2022, 09:20 Uhr).

Verlinkte Seiten

Für beide Listen wurde zunächst allen Verlinkungen gefolgt, wobei einige Seiten erreichbar waren, aber keinen Inhalt hatten (in Abbildung 6.1 rot hinterlegte Namen). Diese Fälle wurden abgefangen und verworfen.

Filtern der Seiten anhand von Kategorien

Verlinkte Seiten wie „Zeit des Nationalsozialismus“, oder „Musiker“, gehören nicht in den Korpus und sollten ebenso verworfen werden. Daher wurden alle Seiten nach ihren Kategorien abgefragt. Das Filtern nach Kategorien stellte sich allerdings als eine Herausforderung heraus, da die Kategorisierung von Wikipedia-Artikeln eine „Folksonomy“ ist und Wikipedia keine Regeln dafür vorschreibt.

Zum Beispiel wurde der bekannte Komponist Hanns Eisler sehr detailliert mit insgesamt 24 Kategorien u.a. den folgenden Kategorien zugeordnet: *Komponist klassischer Musik (20. Jahrhundert)* und *Emigrant aus dem Deutschen Reich zur Zeit des Nationalsozialismus*.

Der Komponist Walter Jurmann hingegen wurde von den beiden nur der letzten Kategorie zugeordnet, aber daneben in die drei Kategorien *Komponist (Österreich)*, *Filmkomponist*, *Komponist (Schlager)*.

Würde nun lediglich nach *Komponist klassischer Musik (20. Jahrhundert)* gefiltert werden, wäre Walter Jurmann verworfen.

Um alle Komponisten vollständig zu erfassen, musste also eine Strategie entworfen werden. Im ersten Schritt wurden die Kategorien der Seite abgefragt. Falls das Wort „Komponist“ oder „komponist“ als Teilstring in einer der Kategorien vorkommt, wurde die Seite als „composer“ gelabelt. Falls nicht, wurde die Zusammenfassung der Seite, welche üblicherweise 1-3 Sätze umfasst, aufgerufen und nach dem Wort „Komponist“ abgesucht.

Da nun aber auch in anderen musikbezogenen Artikeln das Wort „Komponist“ in der Zusammenfassung vorkommen kann, wurde noch eine letzte Überprüfung durchgeführt. Wurde also das Wort „Komponist“ in der Zusammenfassung des Artikels gefunden, musste versichert werden, dass der Artikel wirklich eine Person beschreibt. Falls eine der Kategorien *Mann*, *Frau*, *Person*, *Geboren*, *Gestorben* in den Kategorien vorkam, wurde die Seite als „composer“ gelabelt. Ansonsten wurde die Seite verworfen.

Nach der Überprüfung aller Verlinkungen der Listen mit dem beschriebenen Verfahren, wurden alle als „composer“ erkannten Artikel in den in der Konfigurationsdatei angegebenen Sprachen aus der Wikipedia gelesen, falls sie in der jeweiligen Sprache existierten.

```
In [2]: scrape_and_generate_data()

Collecting groups...

Processing wikipedia page: Liste der vom NS-Regime oder seinen Verbündeten verfolgten Komponisten
in language: de
Collecting sub-pages that contain Komponist categories only...
FOUND: Abel Ehrlich is Komponist
IGNORED: Miklós Horthy CATEGORIES DIDN'T MATCH...
FOUND: Alexander von Zemlinsky is Komponist
FOUND: Alexandre Tansman is Komponist
IGNORED: Alfons Josef Biron PAGE DOESN'T EXIST...
IGNORED: Alfred Kastner (Musiker) PAGE DOESN'T EXIST...
IGNORED: Alfred Rosé PAGE DOESN'T EXIST...
FOUND: Alfred Szendrei is Komponist
FOUND: Andre Asriel is Komponist
IGNORED: Andries de Rosa PAGE DOESN'T EXIST...
FOUND: Antal Doráti is Komponist
FOUND: Anton Webern is Komponist
```

Abbildung 6.2.: Konsolenoutput des Data Scrapings mit Filtern von Komponisten.

6.2.2. Filtern der relevanten Teile der Artikel

Wikipedia-Artikel über Komponisten beinhalten häufig viele Auflistungen beispielsweise von Werken, oder Ehrungen. Zur Verdeutlichung zeigt Abbildung 6.3 das Inhaltsverzeichnis der Seite über den Komponisten Carl Orff. Für diese Arbeit relevanter biografischer Text, bestehend aus vollständigen und zusammenhängenden Sätzen, ist nur im ersten Abschnitt „Leben“ und dessen Unterabschnitten vorhanden. Die Abschnitte 2-9 bestehen aus Auflistungen.

Carl Orff (* 10. Juli 1895 in München; † 29. März 1982 ebenda) war ein deutscher Komponist und Musikpädagoge der populärsten Chorwerke des 20. Jahrhunderts wurde.

| Inhaltsverzeichnis [Verbergen] | |
|---|--|
| 1 Leben | |
| 1.1 Verhältnis zum NS-Staat | |
| 1.2 Weitere Tätigkeiten | |
| 1.3 Privatleben | |
| 2 Werke | |
| 2.1 Bühnenwerke | |
| 2.2 Andere Werke | |
| 3 Ehrungen (Auswahl) | |
| 4 Nachwirken und Rezeption | |
| 5 Schüler | |
| 6 Siehe auch | |
| 7 Literatur | |
| 8 Weblinks | |
| 9 Einzelnachweise | |

Abbildung 6.3.: Aufbau des Wikipedia-Artikels über Carl Orff,
https://de.wikipedia.org/wiki/Carl_Orff, letzter Zugriff: 10.11.2022 (Screenshot vom 10.11.2022, 10:00 Uhr).

Obwohl auch vereinzelt längere Komponisten-Artikel existieren, deren relevanter Text sich über den ersten Abschnitt des Wikipedia-Artikels zieht, musste für das *Wikipedia Scraping* eine allgemeine Lösung entwickelt werden. Daher wurde für alle Artikel lediglich die Zusammenfassung, sowie der erste Hauptabschnitt mit all seinen Unterabschnitten ausgelesen. Da die Unterabschnitte wiederum Unterabschnitte haben konnten, wurde eine rekursive Funktion implementiert, die es ermöglichte den gesamten Text des ersten Abschnitts, mit variabler Ebenenanzahl der Unterabschnitte aufzunehmen.

6.2.3. Analyse der Daten

Insgesamt wurden die Texte von 457 Komponisten-Artikeln mit 1529 Paragraphen, 17245 Sätzen und 276023 Wörtern in sprachspezifische JSON-Dateien geschrieben.

Im Anhang A befinden sich die Visualisierungen als Tortendiagramme zu den Sprachverteilungen der extrahierten Artikel, Paragraphen, Sätze und Wörter.

Von den 457 Artikeln wurden 376 Artikel mit dem Label „verfolgt“ versehen, und die übrigen 81 Artikel mit dem Label „begnadet“.

Die Tabelle 6.1 zeigt die Klassen- und Sprachverteilung der Artikel.

Tabelle 6.1.: Klassen- und Sprachverteilung der Wikipedia-Artikel.

| Sprache | #Artikel | davon „begnadet“ | davon „verfolgt“ |
|----------|----------|------------------|------------------|
| Deutsch | 266 | 47 | 219 |
| Englisch | 191 | 34 | 157 |

Es wird deutlich, dass das Thema in der deutschen Wikipedia besser dokumentiert ist, trotz größeren Größe der englischen Wikipedia. Außerdem wird ersichtlich, dass es sich um einen unausgeglichenen Datensatz handelt. Tabelle 6.2 zeigt die durchschnittliche Anzahl der Paragraphen, Sätze und Wörter der extrahierten Datensätze in den verschiedenen Sprachen.

Tabelle 6.2.: Durchschnittliche Paragraphen, Sätze und Wörter der extrahierten Datensätze.

| Sprache | \varnothing Paragraphen pro Artikel | \varnothing Sätze pro Artikel | \varnothing Wörter pro Artikel |
|----------|---------------------------------------|---------------------------------|----------------------------------|
| Deutsch | 2,4 | 32,4 | 487,5 |
| Englisch | 2,5 | 25,4 | 548,2 |

Die durchschnittlichen Paragraphen, Sätze und Wörter der gesamten Wikipedia-Artikel (nicht nur der extrahierten Textanteile) konnten leider nicht mehr untersucht werden, da das Wikipedia Scraping zum Ende der Bearbeitungszeit immer wieder von der MediaWiki API abgebrochen wurde. In der Fehlermeldung wurden als Grund Wartungsarbeiten angegeben.

6.3. Feature Extraction

6.3.1. Datenbereinigung und -vorverarbeitung

Die Datenbereinigung und -vorverarbeitung zählt in der Pipeline von Kowsari u. a. (2019) zum Teil der *Feature Extraction*. Es wurden folgende Schritte durchgeführt:

- Tokenisierung
- Lemmatisierung
- Entfernung von Stoppwörtern, Punktierungen und Lautschrift

Tokenisierung bezeichnet die Segmentierung von Texten in Wörter (Jurafsky und Martin 2009). In SpaCy muss der Text zunächst in seine Sätze segmentiert werden, bevor auf diese Sätze Tokenisierung angewendet wird. Lemmatisierung bewirkt die Rückführung eines Wortes von seiner flektierten in seine unflektierte Form (auch „Stamm“ oder „Basisform“ gennant) durch morphologische Analyse des Wortes (Jurafsky und Martin 2009).

Beispiele für die Rückführung von Token zu ihren Lemmata sind:

sangen -> *singen* oder *Instrumente* -> *Instrument*.

Durch die Lemmatisierung kann die semantische Bedeutung eines Wortes verstärkt werden (Liu, Lin und Sun 2020). Als Stoppwörter werden in der NLP Wörter bezeichnet, die in Texten sehr häufig auftreten, aber wenig Inhalt liefern.

Punktierungen und Sonderzeichen, die häufig in Wikipedia-Artikeln vorkommen, sollten ebenso aus der Tokenliste entfernt werden. Daher wurde eine Liste dieser Zeichen, bzw. Zeichenkombinationen erstellt. Entfernt wurden:

, . ; : (; - – ” „ / ” ”() [] ! ? = & * †

Außerdem sollten auch Namen in phonetischer Schrift entfernt werden, beispielsweise [‘be:lɒ:bɔrto:k’] für den Komponisten Béla Bartók. Dies konnte durch eine selbsterstellte Regex über die Substitutionsfunktion der *re* Bibliothek erreicht werden.

Ergebnis

Nachfolgend wird das Ergebnis der beschriebenen Arbeitsschritte auf zwei Sätze aus Franz Schrekers Biografie aufgezeigt.

Bereits in den späten 1920er-Jahren war Schreker Angriffsobjekt der Kulturpolitik der Nationalsozialisten. 1932 wurde auf Grund des NS-Terrors die in Freiburg geplante Uraufführung seiner Oper *Christophorus* von Schreker selbst zurückgezogen, und er wurde zum Rücktritt von seinem Amt als Direktor der Berliner Musikhochschule gezwungen, die er seit 1920 geleitet hatte.
(Wikipedia 2022a)

Ergebnissätze:

- „bereits“, „spät“, „1920er-Jahr“, „Schreker“, „Angriffsobjekt“, „Kulturpolitik“, „Nationalsozialist“
- „1932“, „Grund“, „NS-Terror“, „Freiburg“, „geplant“, „Uraufführung“, „Oper“, „Christophorus“, „Schreker“, „zurückziehen“, „Rücktritt“, „Amt“, „Direktor“, „Berliner“, „Musikhochschule“, „zwingen“, „seit“, „1920“, „leiten“

6.3.2. MUSE

Es wurden monolinguale fastText Word Embeddings auf den bereinigten Texten der jeweiligen Sprache trainiert. Auf diesen wurde dann Word Embedding Alignment von der ressourcenreichereren Sprache Deutsch (*source*) auf die ressourcenärmere Sprache Englisch (*target*) durchgeführt. Der Code für die in diesem Abschnitt präsentierten Visualisierungen mit t-SNE stammt aus der MUSE Bibliothek ⁴. Abbildung 6.4 zeigt die selbst trainierten monolingualen fastText Wortvektoren im 2-dimensionalen Vektorraum vor dem Alignment.

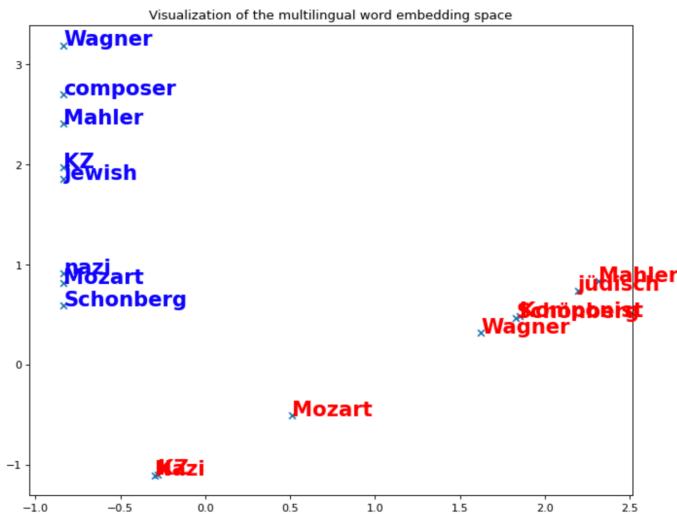


Abbildung 6.4.: Ausgewählte Wortvektoren in Englisch (blau) und Deutsch (rot) im 2-dimensionalen Vektorraum vor dem Alignment.

Der Vergleich zu der Abbildung 6.5 nach dem Alignment zeigt, dass der alignierte Wortvektorraum Komponisten sprachübergreifend darstellen kann.

⁴<https://github.com/facebookresearch/MUSE/blob/main/demo.ipynb> [Letzter Zugriff am 26.11.2022 um 9:30 Uhr]

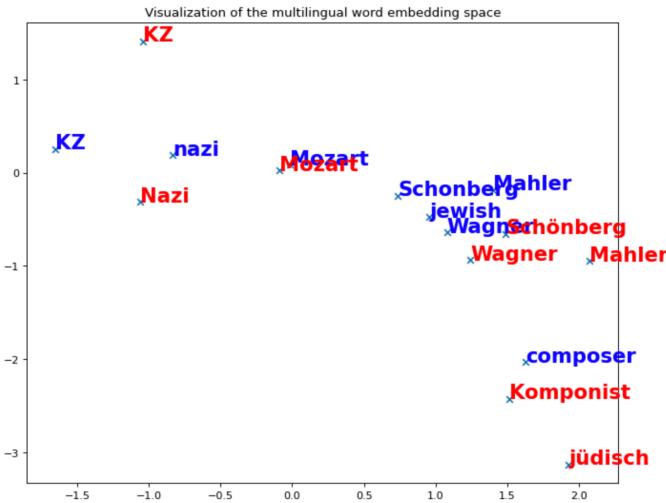


Abbildung 6.5.: Ausgewählte Wortvektoren in Englisch und Deutsch im 2-dimensionalen Vektorraum nach dem Alignment.

Die auf dem Wikipedia Korpus vortrainierten und von Conneau u. a. (2017) im gemeinsamen Vektorraum alignierten Word Embeddings zeigen für die ausgewählten Wörter ebenso gute Ergebnisse (s. Abb. 6.6). Allerdings wird hier beispielsweise zwischen dem Komponisten Wolfgang Amadeus Mozart und den in dieser Arbeit fokussierten Komponisten kein semantischer Unterschied deutlich. Zudem befand sich der deutsche Wortvektor für Schönberg nicht in der Nähe des englischen, und darüber hinaus nicht in der Nähe der „Komponistenwolke“.

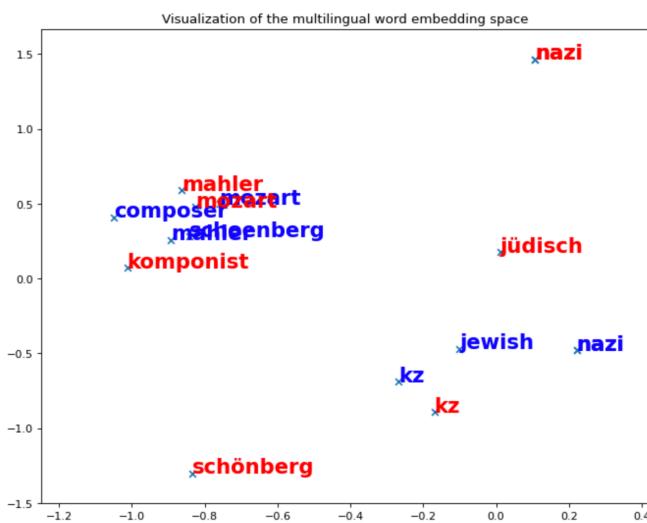


Abbildung 6.6.: Ausgewählte vortrainierte alignierte Wortvektoren in Englisch und Deutsch im 2-dimensionalen Vektorraum.

6.3.3. BERTopic

BERTopic nimmt eine Liste von Dokumenten, als Eingabe und generiert aus diesen Themenwörtern, die in Clustern ähnlicher Themenwörter zusammengefasst werden (s. Kapitel 2.2). Im Durchlauf des Trainings auf Artikelebene, hat sich herausgestellt, dass keine den vielschichtigen Inhalt eines Artikels angemessen repräsentierende Themen entstehen konnten. Daher wurde entschieden das BERTopic Modell auf Satzebene zu trainieren. Für die Embeddings wurden die standardmäßigen Embedding Modelle von BERTopic gewählt, welche Sentence Transformer (Reimers und Gurevych 2019) für die jeweilige Sprache sind. Für den multilingualen Fall ist dies beispielsweise das multilinguale *paraphrase-multilingual-MiniLM-L12-v2*⁵. Daher konnte das multilinguale Modell auf den Sätzen aller Sprachen gleichzeitig trainiert werden. Die monolingualen BERTopic-Modelle wurden nur auf den Sätzen der jeweiligen Sprache trainiert. Die offizielle Dokumentation von BERTopic empfiehlt, die Texte in ihrer ursprünglichen Form einzugeben und lediglich Stoppwörter zu entfernen. Daher wurden die konkatenierten Stopwortlisten der verwendeten Sprachen, welche bereits für die Datenvorverarbeitung benutzt wurden, dem BoW-Modell im BERTopic-Algorithmus übergeben. Für den zum Clustern verwendeten UMAP-Algorithmus wurde die minimale Clustergröße 10 definiert. Insgesamt generierte das multilinguale Modell 285 Topics. Im Anhang B befindet sich eine Liste der 26 häufigsten Topics des multilingualen Modells mit ihren 10 wahrscheinlichsten Topic-Wörtern anhand ihres c-TF-IDF Scores, wobei die meisten die minimale Clustergröße haben und ihre Reihenfolge somit bei wiederholtem Training mit denselben Konfigurationen variiert. Das Topic „-1“ umfasst Dokumente, die nicht eingeordnet werden konnten, sog. *outliers*. Üblicherweise ist dies das häufigste Topic.

| Topic | Count | Name |
|-------|-------|--|
| 0 | -1 | 4814 -1_wurde_berlin_wien_music |
| 1 | 0 | 249 0_schoenberg_schönberg_arnold_schönbergs |
| 2 | 1 | 206 1_adorno_horkheimer_adornos_sozialforschung |
| 3 | 2 | 168 2_jedoch_aufgeführt_mehr_kam |
| 4 | 3 | 162 3_deutscher_komponist_dirigent_amerikanischer |
| ... | ... | ... |
| 281 | 280 | 10 280_goldner_goldschlag_goldschmidt_berthold |
| 282 | 281 | 10 281_riemann_godowsky_sondershausen_baudelaire |
| 283 | 282 | 10 282_meisel_illiard_eliza_verleger |
| 284 | 283 | 10 283_exhausted_barn_fractured_straw |
| 285 | 284 | 10 284_beethoven_requiem_sonatas_32 |

Abbildung 6.7.: Generierte Topics und ihre Häufigkeiten im des multilingualen BERTopic Modells.

⁵https://www.sbert.net/docs/pretrained_models.html [Letzter Zugriff am 23.11.2022 um 14:15 Uhr]

Abbildung 6.8 stellt die 8 häufigsten Topics des deutschen BERTopic Modells in einem Balkendiagramm dar. Im Vergleich zu den Topics des multilingualen Modells (s. Abb 6.7) sind Parallelen der gefundenen Topics erkennbar. Beispielsweise ist in beiden Modellen das Thema um den Komponisten Arnold Schönberg wichtig.

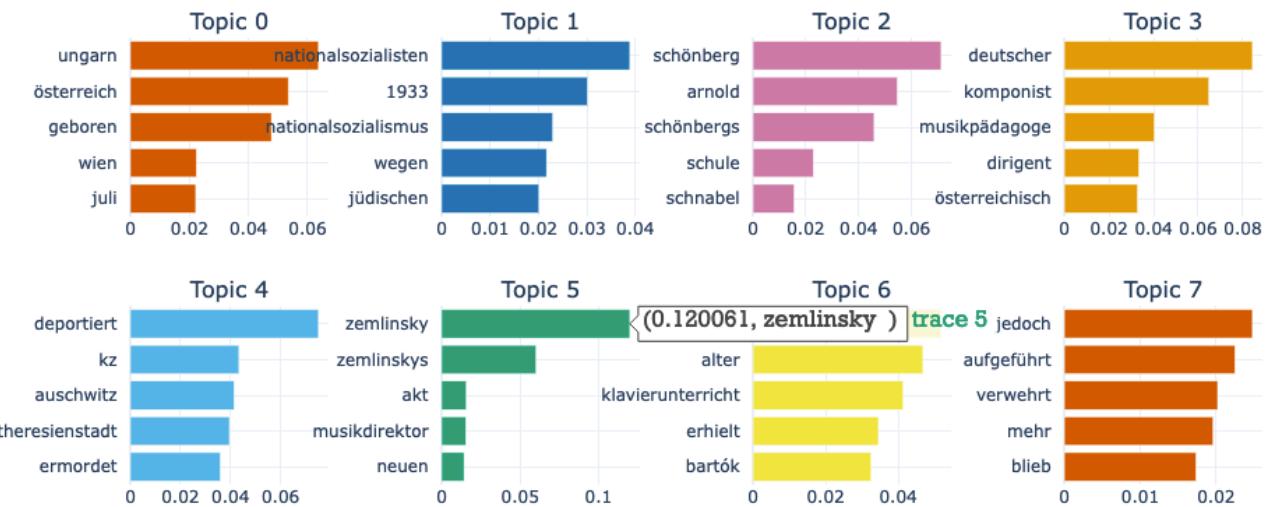


Abbildung 6.8.: c-TF-IDF Scores der 8 häufigsten Topics im deutschen BERTopic Modell.

Zieht man allerdings noch die 8 häufigsten Topics des englischen Modells zum Vergleich hinzu (s. Abb. 6.9) ist das Thema um Arnold Schönberg nicht dabei.

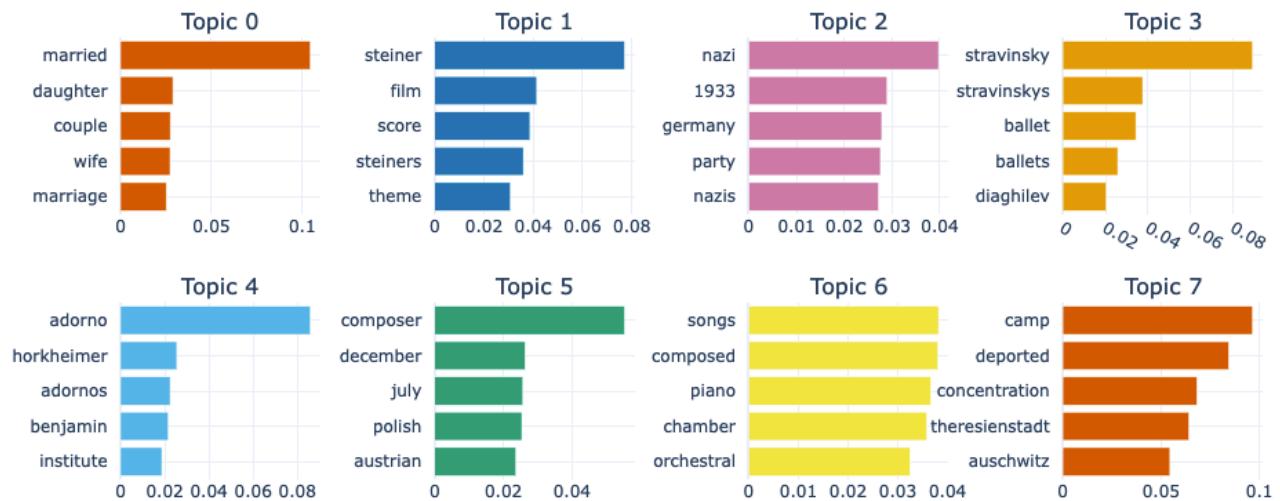


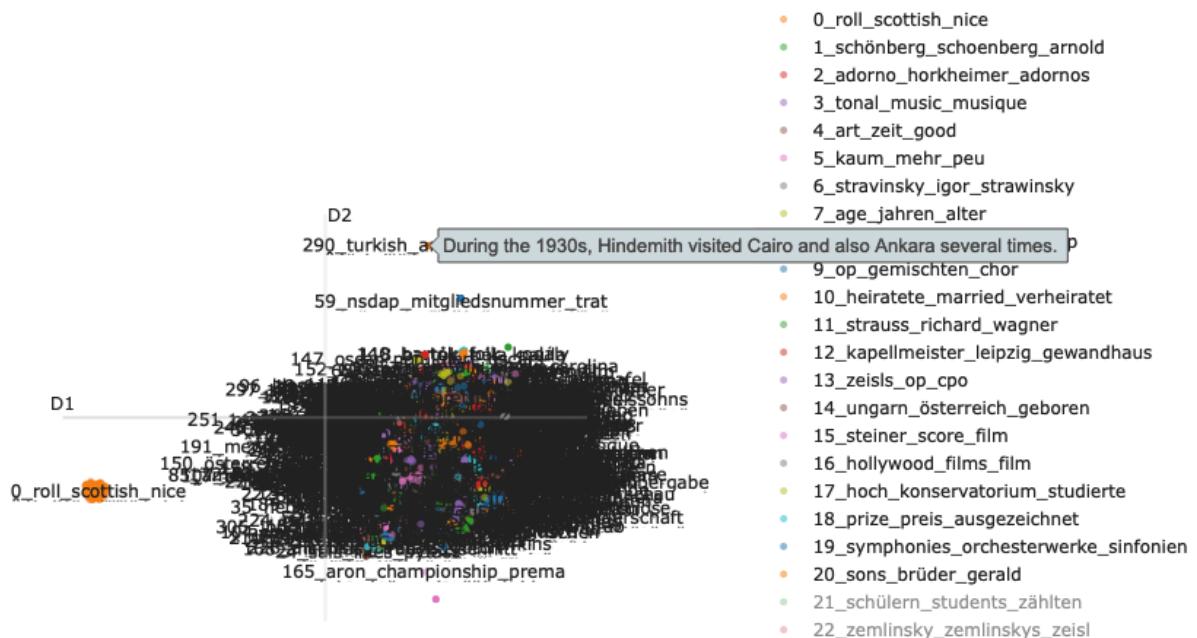
Abbildung 6.9.: c-TF-IDF Scores der 8 häufigsten Topics im englischen BERTopic Modell.

Schönberg scheint dennoch auch in den englischen Artikeln häufig erwähnt worden zu sein, denn im multilingualen Modell ist das Thema auf dem ersten Platz gelandet (s. Abb. 6.7).

Visualisierung der Themenzuordnungen von Sätzen

BERTopic implementiert eine Methode zur Visualisierung der Themenzuordnungen von Dokumenten im 2-dimensionalen Raum auf Grundlage der gelernten Embeddings des Topic Models. Der so entstandene Plot lässt sich interaktiv durch Herüberfahren mit der Maus (*Hover-Effekt*) nach zugeordneten Dokumenten erkunden. Die nachfolgenden Visualisierungen wurden auf einem multilingualen BERTopic Modell mit den Sprachen Deutsch, Englisch und Französisch erzeugt. Französische Komponistenartikel wurden im Laufe der Bearbeitungszeit verworfen, aufgrund der erhöhten Komplexität des Sprachenvergleichs der Textklassifikation. Dennoch wurden die Screenshots beibehalten, weil sie anschaulich zeigen, dass BERTopic in der Lage war, musikhistorische Zusammenhänge im 2-dimensionalen Raum abzubilden.

Es konnten interessante Entdeckungen gemacht werden, beispielsweise in Abbildung 6.10. Dort befindet sich ein Cluster von Sätzen, welches weit entfernt von der „großen schwarzen Wolke“ der Satz-Thema-Zuordnungen befand. Das entfernte Topic lässt sich mit dem Thema „Türkei“ benennen. Ein Beispielsatz handelte von Türkeibesuchen des Komponisten Paul Hindemith und dem großen Respekt, den er bei türkischen Musikern innehatte.



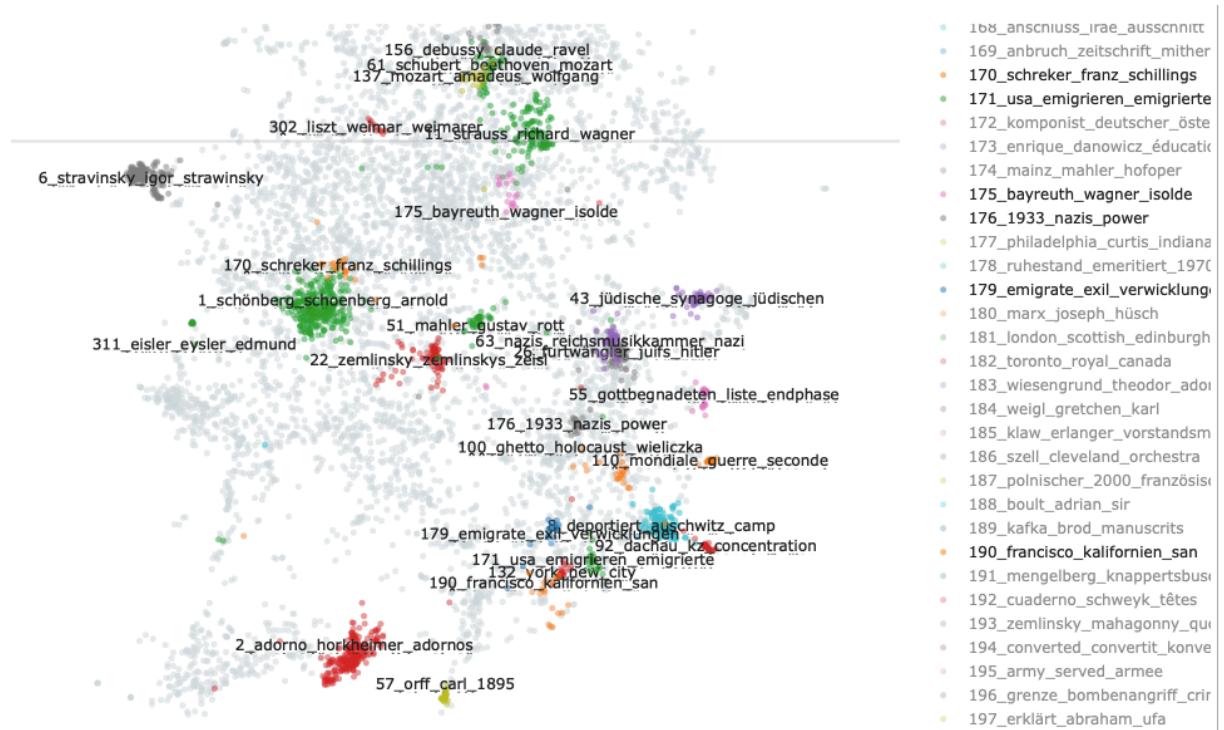


Abbildung 6.11.: Ausgewählte Topic Cluster im 2-dimensionalem Raum.

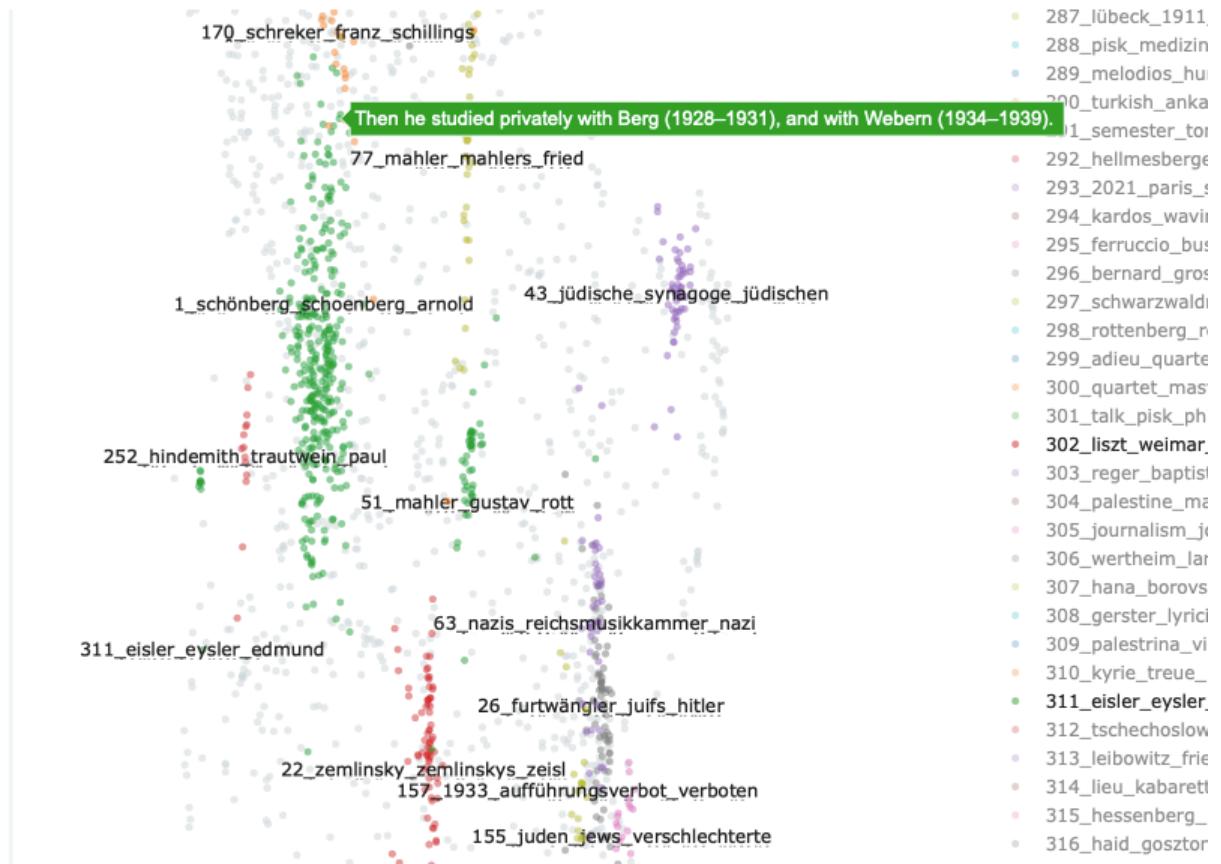


Abbildung 6.12.: Topic Cluster verfolgter Komponisten im 2-dimensionalem Raum.

6.3.4. Training auf um Themenwörter erweiterte Texte

Mit einer von BERTopic implementierten Methode kann eine Visualisierung der Verteilung der Topics pro Klasse erstellt werden. Es können somit bestimmte Topics ausgewählt werden, von denen erwartet wird, dass sie häufiger in einer bestimmten Klasse auftreten. Dabei waren die meisten Topics aufgrund des Ungleichgewichts der Klassen im Datensatz in der Klasse „verfolgt“ (0) vorzufinden. Es gab jedoch Ausreißer, die häufiger in der Klasse der „gottbegnadeten“ Komponisten vorzufinden waren und aus musikwissenschaftlicher Sichtweise dort auch besser passen.

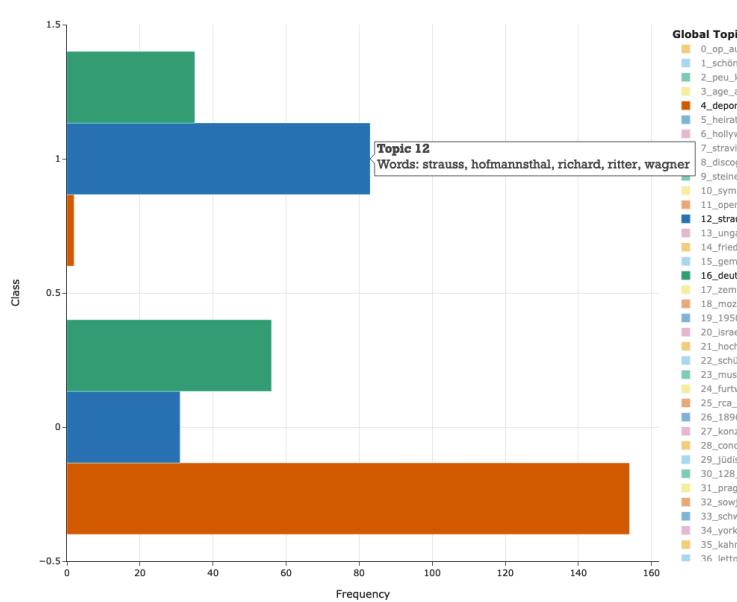


Abbildung 6.13.: Balkendiagramm der Verteilung ausgewählter Topics pro Klasse (0=„verfolgt“, 1=„begnadet“).

Daher wurde der Versuch unternommen, die zu einem Text passenden Topics mit ihren Themenwörtern und den bereinigten Text selbst zu konkatenieren. Mithilfe Topic Models, das auf Sätzen trainiert wurde, mussten dafür zunächst die häufigsten Topics den einzelnen Sätzen eines Artikels zugeordnet werden.

Es wurden die n=3 wahrscheinlichsten Topics für jeden Artikel abgefragt. Die 10 Topic Wörter eines Topics wurden dann an die Liste der Wörter des bereinigten Artikels gehangen. Das führte zu einer Verlängerung jedes Textes von 30 Wörtern.

```

Analysing sentence= Ab 1925 war Heller Privatstudent bei Franz Schreker, 1931 kam im Berliner Rundfunk erstmals ein
es seiner Werke zur Aufführung.
Analysing sentence= 1933 musste die Familie vor den Nazis nach Frankreich fliehen, Heller wurde aber nach Ausbruch
des Zweiten Weltkrieges verhaftet und musste Zwangsarbeit für die Organisation Todt leisten.
Analysing sentence= Der Deportation nach Auschwitz entkam er durch Flucht und wurde von der Résistance versteckt.
Analysing sentence= 1946 emigrierten die Hellers in die USA, er konnte sich dort als Komponist aber nie durchsetze
n.
Analysing sentence= 1955 kehrten sie nach Deutschland zurück.
Analysing sentence= Hans Heller starb 1969, sein Werk geriet bald darauf in Vergessenheit.

```

Abbildung 6.14.: Konsolenoutput des Mappings von Themen auf Artikel.

Leider musste festgestellt werden, dass bestimmte Themenwörter sich oft wiederholten, da sie zu besonders häufigen Topics gehörten. In Abbildung 6.15 ist ein Ausschnitt aus der CSV-Datei der bereinigten Datensätze gegeben, der diesen Umstand verdeutlicht.

| cleaned_texts_topics | topics |
|---|---|
| ['Abel', 'Ehrlich', '3.', 'September', '1915', 'Cranz', ...] | ['schoenberg', 'schönberg', 'arnold', 'schönbergs', 'sch...'] |
| ['Alexander', 'Zemlinsky', 'Pseudonym', 'Al', 'Roberts', ...] | ['israel', 'jerusalem', 'aviv', 'tel', 'rubin', 'palesti...'] |
| ['Alexandre', 'Tansman', 'französisch', 'Alexandr  ', 'Ta...] | ['juden', 'jews', 'angesichts', 'people', 'fortschrittso...'] |
| ['Alfred', 'Szendrei', 'Alfred', 'Sendrey', 'Alad  r', 'S...] | ['musiktheorie', 'harmony', 'legte', '1929', 'musiklehre...'] |
| ['Andre', 'Asriel', '22.', 'Februar', '1922', 'Wien', '2...] | ['violinunterricht', 'cello', 'violin', 'five', 'organ', ...] |
| ['Antal', 'Dor  ti', ' ', '9.', 'April', '1906', 'Budape...] | ['israel', 'jerusalem', 'aviv', 'tel', 'rubin', 'palesti...'] |
| ['Anton', 'Webern', '3.', 'Dezember', '1883', 'Wien', '1...] | ['israel', 'jerusalem', 'aviv', 'tel', 'rubin', 'palesti...'] |
| ['Arnold', 'Sch  nberg', '13.', 'September', '1874', 'Wie...] | ['israel', 'jerusalem', 'aviv', 'tel', 'rubin', 'palesti...'] |
| ['Arthur', 'Oskar', 'Chitz', '5.', 'September', '1882', ...] | ['sinfonien', 'symphonies', 'orchesterwerke', 'orchestra...'] |

Abbildung 6.15.: Redundanz bestimmter Topics bei Artikeln.

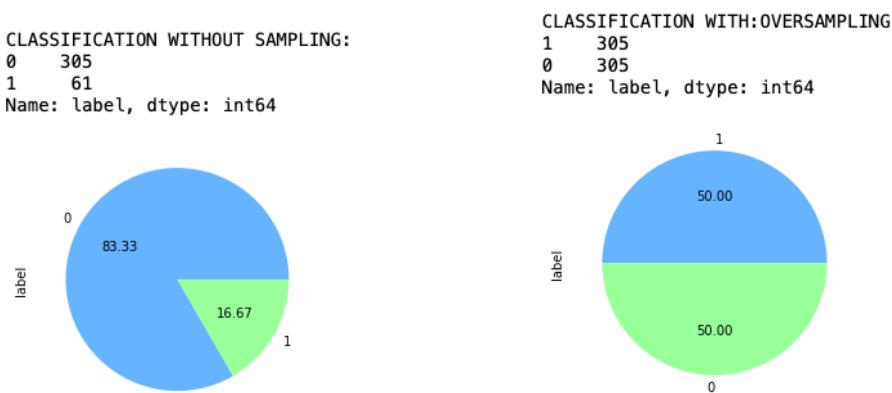
Auf den so entstandenen Texten wurde dann ebenso eine Variante der Word Embeddings trainiert, die für die Textklassifikation mit dem SVM und Na  ve Bayes Klassifikator verwendet werden.

6.3.5. Resampling von nicht-balancierten Trainingsdaten

Zuletzt wurden aufgrund des Ungleichgewichts der Klassen im Datensatz Resamplingstrategien für die Textklassifikation mit statischen Word Embeddings verwendet.

F  r die Textklassifikation mit BERT Modellen (siehe n  chstes Kapitel) hat daf  r die Bearbeitungszeit nicht mehr ausgereicht. Allerdings zeigt Kapitel 7, dass die BERT Modelle trotz des unausgeglichenen Datensatzes eine sehr gute Performanz aufwiesen.

Die Abbildungen 6.16a und 6.16b veranschaulichen f  r den Fall des Oversamplings die Distribution der Klassen in den Trainingsdaten des multilingualen Datensatzes vor und nach dem Resampling.



- (a) Distribution der Klassen in den Trainings-(b) Distribution der Klassen in den Trainings-
daten vor dem Oversampling (0=„ver- daten nach dem Oversampling
folgt“, 1=„begnadet“).

6.4. Finetuning von BERT für die Textklassifikation

Das Finetuning wurde auf der Task der Textklassifikation betrieben. Dafür wurde der veröffentlichte Code⁶ von Barbieri, Anke und Camacho-Collados (2021) für das Finetuning von XLM-T, einem multilingualen Sprachmodell trainiert auf Twitter Daten, auf den eigenen Anwendungsfall angepasst.

Die Texte und ihre korrespondierenden Label wurden in Trainings-, Validierungs- und Testdatensätze eingeteilt. Dabei wurden die unbereinigten Texte als Input ausgewählt, da die gewählten BERT Modelle selbst auf Wikipediadaten trainiert wurde. Da die maximale Sequenzlänge einer Eingabe auf 512 Token beschränkt ist, wurden längere Artikel gekürzt, während kürzere Artikel mit *Padding* versehen wurden. Die notwendigen Tokenisierungsschritte wurden dann vom zum vorgenannten Modell passenden Tokenisierer vorgenommen, der durch die *AutoTokenizer* Klasse von *Huggingface* geladen wurde.



```
1 Józef Koffler (28 November 1896 – 1944) was a Polish composer, music teacher, musicologist and musical columnist.  
2 Lex van Delden (eigentlich Alexander Zwapp, * 10. September 1919 in Amsterdam; † 1. Juli 1988 ebenda) war ein nie  
3 Simon (Sim) Gokkes (* 21. März 1897 in Amsterdam; † 5. Februar 1943 im KZ Auschwitz) war ein niederländischer Kom  
4 Hermann Leopoldi (born Hersch Kohn; 15 August 1888 – 28 June 1959) was an Austrian composer and cabaret star who  
5 Karol Rathaus (Karl Leonhard Bruno Rathaus; Pseudonym Leonhard Bruno; geboren 16. September 1895 in Tarnopol, Öst  
6 Franz Kinzl (* 2. Juli 1895 in Mettmach; † 23. April 1978 in Lambach) war ein österreichischer Militärkapellmeist  
7 Richard Fall (3 April 1882 – January 1945) was an Austrian composer and conductor of Jewish descent. One of his m  
8 Karel Reiner (* 27. Juni 1910 in Saaz als Karl Reiner; † 17. Oktober 1979 in Prag) war ein tschechischer Komponis  
9 Berthold Goldschmidt (geboren 18. Januar 1903 in Hamburg; gestorben 17. Oktober 1996 in London) war ein deutsch-b  
10 Bruno Bernhard Granichstaedten (* 1. September 1879 in Wien; † 30. Mai 1944 in New York) war ein österreichischer  
11 Ernst Steffan (* 22. Jänner 1890 in Wien als Ernst Steiner; † 26. September 1967 in Berlin) war ein österreichisc  
12 Fried Walter (* 19. Dezember 1907 in Ottendorf-Okrilla als Walter Emil Schmidt; † 8. April 1996 in Berlin) war ei  
13 Paul Amadeus Pisk (* 16. Mai 1893 in Wien, Österreich-Ungarn; † 12. Januar 1990 in Los Angeles) war ein österreic  
14 Guillermo Graetzer, geboren als Wilhelm Grätzer (* 5. September 1914 in Wien; † 22. Januar 1993 in Buenos Aires)
```

Abbildung 6.17.: Ausschnitt aus dem Trainingsdatensatz für das BERT Modell.

Da die Anzahl der Daten sehr gering ist, war es möglich, mit vielen verschiedenen Hyperparametern zu experimentieren. Es wurden dabei die von Devlin u. a. (2019) angegebenen Hyperparameter für das Finetuning beachtet. Für die Epochenzahl wurden aber teilweise weitaus höhere Epochen ausprobiert, da diese gute Ergebnisse aufzeigten und die Berechnung auf dem Datensatz relativ schnell ging.

Es wurde beobachtet, dass eine Erhöhung der Batch-Größe die Accuracy verbesserte. Die Erhöhung der Epochenanzahl führte zu einer höheren Stabilität der Ergebnisse. Allerdings ließ sich nach einer je nach Konfiguration variierender Epochenanzahl beobachten, dass der Loss auf dem Validationsdatensatz wieder anstieg, was auf Overfitting hindeuten könnte. Beispielsweise war das für die Lernrate 5e-5 ab 5 Epochen der Fall.

⁶<https://colab.research.google.com/drive/1IAA1h8u53O1hi9807u7oOFuT3728N0-n?usp=sharing#scrollTo=hqeWwkSCXaKV> [Letzter Zugriff am 27.11.2022 um 08:20 Uhr]

7. Evaluation

In diesem Kapitel wird die Güte der verschiedenen Klassifikationsmodelle betrachtet, indem die Vorhersagen der Modelle mit den tatsächlichen Labels verglichen werden.

Die für die Evaluation verwendeten Metriken Accuracy, Precision, Recall, und der F1-Score wurden im Grundlagenteil (s. Kapitel 2.3.3) beschrieben. Es werden die Abkürzungen „P“, „R“, „F1“ und „Acc“ verwendet. Sie wurden über die Methode `classification_report` von `scikit.metrics` der `scikit-learn` (Pedregosa u. a. 2011) Bibliothek berechnet.

Im Folgenden berechneten sich die Werte aus den *macro averages* der jeweiligen Werte für die beiden Klassen, um die Auswirkungen der Unausgeglichenheit des Datensatzes aufzuzeigen. Eine Ausnahme stellt die Accuracy dar, die sich nicht klassenspezifisch verhält, sondern generell die Anzahl der korrekten Vorhersagen unter allen Vorhersagen bemisst.

7.1. Ergebnisse der besten Modelle

Für die Vergleichbarkeit der Ergebnisse wurde für die jeweiligen Durchführungen mit derselben Aufteilung der Datensätze in Trainings-, (Validierungs-), und Testdaten gearbeitet. Für beide Ansätze der Textklassifikation wurden umfassende Auswertungen für den deutschen, englischen und den deutsch-englischen Datensatz erstellt, welche sich im Anhang C und D befinden. Die besten Modelle wurden anhand des höchsten *macro average* F1-Scores bestimmt.

7.1.1. Textklassifikation mit Support Vector Machine und Naïve Bayes

Für die verschiedenen Datensätze wurde die Aufteilung in 80% Trainings- und 20% Testdaten vorgenommen. In Tabelle 7.1 werden für jeden Datensatz die besten Textklassifikationen mit Support Vector Machine und Naïve Bayes in einer Tabelle zusammengefasst. Für jeden Datensatz wird jeweils das beste Word2Vec- und das beste fastText-Modell gezeigt, da Word2Vec die *Baseline* darstellt.

Tabelle 7.1.: Performanz der besten Klassifikationsverfahren auf fastText- und Word2Vec-Embeddings.

| Datensatz | Embeddings | Features | Res. | Klass. | P | R | F1 | Acc |
|-----------|------------|---------------|---------|--------|-------------|-------------|-------------|-------------|
| de | W2V | text | Unders. | NB | 0.59 | 0.56 | 0.49 | 0.50 |
| de | fT (p,a) | text | Overs. | SVM | 0.83 | 0.77 | 0.79 | 0.87 |
| en | W2V | text | Overs. | NB | 0.53 | 0.53 | 0.53 | 0.64 |
| en | ft (p,a) | text + topics | Overs. | SVM | 0.75 | 0.67 | 0.69 | 0.77 |
| de + en | W2V | text + topics | Unders. | NB | 0.61 | 0.56 | 0.51 | 0.59 |
| de + en | fT (p,a) | text | Over. | SVM | 0.69 | 0.60 | 0.59 | 0.67 |

Die beste Textklassifikation fand auf dem deutschen Datensatz mit dem Klassifikationsverfahren Support Vector Machine und nach dem Oversampling statt. Die Features für die SVM waren Dokument-Embeddings ohne Texterweiterung mit BERTopic. Diese wurden mit den von Conneau u. a. (2017) im Vektorraum alignierten vortrainierten fastText-Embeddings (p, a für *pretrained* und *aligned*) generiert.

7.1.2. Textklassifikation mit BERT

Für das Finetuning wurde die Aufteilung der Datensätze in 70% Trainings-, 15% Validierungs- und 15% Testdaten vorgenommen. Auf den monolingualen Datensätzen wurde jeweils ein monolinguales BERT-Modell, als auch ein multilinguales BERT-Modell verwendet. Auf dem multilingualen Datensatz fand das Finetuning nur mit einem multilingualen Modell statt. Tabelle 7.2 stellt für jeden Datensatz das jeweils beste der verwendeten Modelle dar.

Tabelle 7.2.: Konfiguration und Performanz der besten der jeweils verwendeten BERT Modelle auf den Datensätzen.

| Datensatz | Modell | Epochen | Batchgröße | Lernrate | Loss | P | R | F1 | Acc |
|-----------|--------|---------|------------|----------|------|-------------|-------------|-------------|-------------|
| de | gBERT | 5 | 16 | 5,00E-05 | 0.35 | 0.98 | 0.94 | 0.96 | 0.97 |
| de | mBERT | 20 | 16 | 5,00E-05 | 0.13 | 0.85 | 0.90 | 0.87 | 0.90 |
| en | eBERT | 20 | 8 | 3,00E-05 | 0.11 | 0.75 | 0.96 | 0.81 | 0.93 |
| en | mBERT | 5 | 8 | 3,00E-05 | 0.46 | 0.73 | 0.73 | 0.73 | 0.93 |
| de + en | mBERT | 5 | 16 | 3,00E-05 | 0.46 | 0.93 | 0.98 | 0.96 | 0.97 |

Auf dem deutschen Datensatz konnte *bert-base-german-cased* (gBERT) dieselben besten Werte für Accuracy und F1-Score wie *bert-base-multilingual-cased* (mBERT) auf dem multilingualen Datensatz erreichen. Der höchste Wert für die Precision ist in der deutschen, während der höchste Wert für den Recall in der multilingualen Textklassifikation ist.

7.2. Reflexion

Im direkten Vergleich der beiden Varianten die Textklassifikation durchzuführen wird deutlich, dass das Finetuning von BERT auf der Task der Textklassifikation für alle Datensätze in einer viel höheren Performanz resultiert.

7.2.1. Reflexion der Textklassifikationen mit SVM und NB

Die beste deutsche, englische und deutsch-englische Textklassifikation nutzte stets den SVM-Klassifikator, was sich vielleicht auf die von Christopher D. Manning, Raghavan und Schütze (2008) erwähnte Eigenschaft der SVM zurückführen lässt, gut auf kleinen Datensätzen zu funktionieren. Wie von den Autoren prophezeit, war die Precision dieses Klassifikators wesentlich höher als der Recall. In den Klassifikationen mit NB war der Unterschied zwischen Precision und Recall wesentlich geringer.

Obwohl die besten Textklassifikationen mit SVM und NB stets auf vortrainierten alignierten fastText-Embeddings verliefen, kann nicht sicher gesagt werden, ob der Umstand, dass sie im gemeinsamen Vektorraum angeglichen waren, für dieses gute Ergebnis sorgten. Das liegt daran, dass nicht zusätzlich vortrainierte monolinguale fastText-Embeddings in den Vergleich mit einbezogen wurden. Der Blick in die Tabellen im Anhang C zeigt, dass die vortrainierten und von Conneau u. a. (2017) im gemeinsamen Vektorraum angeglichenen Word Embeddings stets besser F1-Werte erzielten, als die auf den Datensätzen der Bachelorarbeit trainierten und dann mit MUSE alignierten fastText Vektoren (in den Tabellen mit $fT(a)$ notiert).

Die gute Performanz von $fT(p,a)$ scheint daher eher an der Eigenschaft der Vektoren zu liegen, umfangreich auf der Wikipedia vortrainiert zu sein, was sich insbesondere gut auf die zu klassifizierenden Texte dieser Bachelorarbeit auszuwirken scheint, die ebenso aus der Wikipedia stammen.

Bessere Ergebnisse durch linear transformierte fastText, bzw. MUSE-Vektoren, aufgrund der Tatsache, dass sie einen gemeinsamen Vektorraum teilen, analog zu den Ergebnissen von Jiang u. a. (2019), konnten daher nicht beobachtet werden.

Des Weiteren zeigt Tabelle 7.1, dass die Techniken Oversampling und Undersampling für den F1-Score förderlich waren, da sie der kleineren Klasse der „gottbegnadeten“ Komponisten zu Gute kam.

Es kann insgesamt für die Textklassifikation mit den Machine Learning Algorithmen SVM und NB gesagt werden, dass sie ausreichend gut funktionierte, wenn man nur die größere Klasse „verfolgt“ mit hoher Sicherheit vorhersagen möchte. Wenn beide Klassen gleichermaßen wichtig waren, wie in dieser Evaluation, waren die Ergebnisse sehr von dem Vorkommen des Labels „begnadet“ im Testdatensatz abhängig.

fastText zeigte in fast allen Fällen eine bessere Performanz als Word2Vec und insbesondere basieren die besten Textklassifikationen unabhängig von der Sprache der Datensätze immer auf fastText-Embeddings. Der Grund dafür wird in der Eigenschaft von fastText vermutet, die Morphologie einer Sprache besser als Word2Vec und zudem *Out-of-Vocabulary-Words* modellieren zu können (s. Kapitel 2.1.2 in den Grundlagen).

Letztlich konnte keine klare Tendenz zu besseren Ergebnissen durch die Kombination von BERTopic und Word Embeddings wie bei Alhaj u. a. (2022) beobachtet werden. Zumindest nicht mit der für diese Bachelorarbeit adaptierten Methode, die die Texte um ihre wahrscheinlichsten Themenwörter erweiterte.

7.2.2. Reflexion der Textklassifikation mit BERT

In der Tabelle 7.2 für die besten Textklassifikationen mit BERT ist erkennbar, dass auf den monolingualen Datensätzen die monolingualen Modelle besser als das multilinguale BERT-Modell abschneiden. Diese Beobachtung deckt sich mit den Ergebnissen von Velankar, Patil und Joshi (2022) und Chan u. a. (2019).

Anhand der Tabelle der besten Modelle lässt sich keine optimale Konfiguration der BERT-Modelle feststellen. Es könnte aber argumentiert werden, dass die beiden besten Modelle beide im Finetuning mit 5 Epochen und einer Batchgröße von 16 trainiert wurden.

In den Tabellen für den englischen Datensatz in Anhang D zeigt sich, dass sich die Per-

formanz relativ schnell „eingefahren“ hat auf P=0.47, R=0.50, F1=0.48 und Acc=0.93, da das Modell nicht in der Lage war, Komponisten der kleinen Klasse „begnadet“ korrekt zu bestimmen. Dies änderte sich aber beim monolingualen *bert-base-cased*-Modell mit 20 Epochen. Es wäre auch für die Textklassifikation mit BERT und besonders für den englischen Datensatz eventuell vorteilhaft gewesen das Oversampling anzuwenden. Jedoch ließ sich das in Anbe tracht der verfügbaren Bearbeitungszeit nicht mehr realisieren.

7.3. Musikwissenschaftliches Abfragen des besten Modells

Im Folgenden wurden interessengeleitet ausgewählte Passagen von Wikipedia-Artikeln über Komponisten kopiert und dem besten multilingualen BERT Modell als Input übergeben. Es wurde darauf geachtet, dass sowohl der Name des Komponisten, als auch ein repräsentativer Satz über dessen Beziehung zum Nationalsozialismus im Text vorkam.

Es steht immer das mit höchster Wahrscheinlichkeit vorhergesagte Label an erster Stelle.

```
#####
Would the composer described in the following text rather be
considered as 0 = 'entartet' or 1 = 'gottbegnadet' by the nazis?
#####
```

Richard Georg Strauss (* 11. Juni 1864 in München; † 8. September 1949 in Garmisch-Partenkirchen) war der vor allem für seine orchestrale Programmmusik (Tondichtungen), sein Liedschaffen und seine Opern Nach der Machtübernahme der Nationalsozialisten gelang es diesen, den international bekannten Komponi Im April 1933 gehörte Strauss zu den Unterzeichnern des „Protests der Richard-Wagner-Stadt München“ § Am 15. November wurde Strauss zum Präsidenten der Reichsmusikkammer ernannt.
 1) LABEL_1 0.9871
 2) LABEL_0 0.0129

Abbildung 7.1.: Abfragen des besten Modells nach der Klassenzugehörigkeit des „gottbegna deten“ Richard Strauss.

Obwohl Richard Strauss aus der „Gottbegnadeten-Liste“ stammt, aus welcher weniger Daten sätze generiert wurden, hat das Modell mit einer sehr hohen Gewissheit die richtige Klasse vorhergesagt (s. Abb. 7.1).

```
#####
Would the composer described in the following text rather be
considered as 0 = 'entartet' or 1 = 'gottbegnadet' by the nazis?
#####
Jakob Ludwig Felix Mendelssohn Bartholdy[1] (* 3. Februar 1809 in Hamburg; † 4. November 1847 in Leip Er zählt zu den bedeutendsten Musikern der Romantik und setzte als Dirigent Maßstäbe, die das Dirigie Nach der Machtübernahme des NS-Regimes im Jahr 1933 wurden die Werke Mendelssohns kaum noch gespielt. Ein offizielles Verbot existierte zwar nicht, die antisemitische Kampagne der Reichsmusikkammer veran Ein Gegenbeispiel hierzu stellte Wilhelm Furtwängler dar, der im Februar 1934 anlässlich Mendelssohns
  1) LABEL_0 0.9952
  2) LABEL_1 0.0048
```

Abbildung 7.2.: Abfrage des besten Modells nach der Klassenzugehörigkeit des ungesehenen Felix Mendelssohn Bartholdy.

Der Komponist Felix Mendelssohn Bartholdy aus der Abfrage in Abbildung 7.2 war nicht im Datensatz vorhanden, da er nicht in der Zeit des Nationalsozialismus gelebt hat. Obwohl er zu Lebzeiten eine große internationale Karriere hinlegte, wurde sein Andenken von den Nationalsozialisten geschmäht, weil er Jude war. Das Modell war in der Lage, auf Grundlage des übergebenen Textes die Entscheidung zu treffen, dass Mendelssohn von den Nationalsozialisten als „entartet“ eingestuft worden wäre.

Wenn über das Verhältnis von Komponisten zum Antisemitismus gesprochen wird, muss auch Richard Wagner erwähnt werden, der mit seiner Schrift „Das Judentum in der Musik“ aus dem Jahre 1850 das angeblich Jüdische in der Musik aufs Übelste beschimpfte und dessen Musik von Adolf Hitler vergöttert wurde.

Zunächst war die Klassenentscheidung des Modells auf einer Beispielpassage über Richard Wagner enttäuschend, da das Resultat „entartet“ war (s. Abb. 7.3).

```
#####
Would the composer described in the following text rather be
considered as 0 = 'entartet' or 1 = 'gottbegnadet' by the nazis?
#####
```

```
Wilhelm Richard Wagner (* 22. Mai 1813 in Leipzig; † 13. Februar 1883 in Venedig) war ein deutscher Komponist, D
Mit seinen durchkomponierten Musikdramen gilt er als einer der bedeutendsten Komponisten der Romantik. In der Ze
Mit seiner Schrift Das Judentum in der Musik und weiteren Äußerungen gehört Wagner geistesgeschichtlich zu den
In der Zeit des Nationalsozialismus wurde Wagners Werk zum Staatskult erhoben. Mit seiner Schrift Das Judentum
Ob sich der Antisemitismus in seinen musikdramatischen Werken niedergeschlagen hat, ist umstritten und wird bis
1) LABEL_0 0.9952
2) LABEL_1 0.0048
```

Abbildung 7.3.: Abfragen des besten Modells nach der Klassenzugehörigkeit des ungesiehenen Richard Wagner.

Dann fiel auf, dass im Text die Wörter „Antisemitismus“, „Judentum“ und „jüdisch“ vorkamen. Es war zu vermuten, dass das Modell aufgrund dieser Wörter die Klassenentscheidung getroffen hat.

Daher wurde ein neuer Abschnitt der Biografie Richard Wagners in der Wikipedia kopiert, welcher u.a. von Wagners Beschäftigung mit germanischer Mythologie handelte. Nun entschied das Modell die entgegengesetzte Klasse (s. Abb. 7.4).

```
#####
Would the composer described in the following text rather be
considered as 0 = 'entartet' or 1 = 'gottbegnadet' by the nazis?
#####
Wilhelm Richard Wagner (* 22. Mai 1813 in Leipzig; † 13. Februar 1883 in Venedig) war ein deutscher Komponist,
Mit seinen durchkomponierten Musikdramen gilt er als einer der bedeutendsten Komponisten der Romantik. In der z
Wagner beschäftigte sich intensiv mit Stoffen der germanischen Mythologie und Sagenwelt wie dem Schwanenritter,
In Lohengrin, der Ring-Tetralogie und dem Spätwerk Parsifal kreisen seine Gedanken um das Motiv der Erlösung, c
1) LABEL_1 0.9757
2) LABEL_0 0.0243
```

Abbildung 7.4.: Veränderter Input in der Abfrage des besten Modells nach Richard Wagner.

8. Schlussfolgerungen und Ausblick

In dieser Bachelorarbeit konnte gezeigt werden, dass es möglich ist, ein Klassifikationsmodell zu trainieren, das in der Lage ist, die „Klassenzugehörigkeit“ eines Komponisten gemäß nationalsozialistischer Denkweise zu bestimmen.

Durch das Scraping von Wikipedia-Artikeln zu „gottbegnadeten“ und „entarteten“ bzw. in der Zeit des Nationalsozialismus verfolgt oder ermordeten Komponisten und das Filtern der biografischen Teile dieser Artikel konnte zwar nur ein kleiner, aber mehrsprachiger Korpus für diesen Zweck generiert werden, der zudem bereinigt wurde.

Das entstandene beste Klassifikationsmodell war sowohl für die monolingualen, als auch für den multilingualen Datensatz ein vortrainiertes BERT-Base-Modell, welches mit Finetuning auf dem Datensatz und der Task der Textklassifikation angepasst wurde.

Interessengeleitete Abfragen dieses Modells zeigten eine hohe Gewissheit in seinen Entscheidungen, die musikwissenschaftlich diskutiert werden könnten, da sie die Komplexität des Themas und somit die vielen Schattierungen innerhalb der „Komponistenklassen“ ignorieren.

Versuche, die multilinguale Textklassifikation mit Machine Learning Klassifikatoren durch verschiedene auf Word Embeddings basierende Features zu verbessern, führten zwar nicht zum Erfolg, jedoch konnten mit den verwendeten Methoden die Daten besser exploriert werden.

Beispielsweise konnten mittels transformer-basiertem Topic Modeling mit BERTopic sprachübergreifende Themen in den Komponistenbiografien gefunden werden. An dieser Stelle wäre es interessant gewesen weiter zu forschen und beispielsweise mehrere ähnliche aus denen vielen generierten Topics zu finden und zu weniger größeren Topics zusammenzufassen. Außerdem konnte mit linearer Transformation von deutschen auf englische Embeddings anhand von exemplarischen Wortvektoren gezeigt werden, dass Wortvektoren, z.B. von Komponistennamen im Vektorraum sprachübergreifend dargestellt werden können. An dieser Stelle könnten noch weitere Zusammenhänge zwischen den Wörtern und Sprachen exploriert werden.

Die Autoren Kowsari u. a. (2019), sowie Ikonomakis, Kotsiantis und Tampakas (2005) erwähnen außerdem die potenziell verbesserte Performanz durch Kombination verschiedener Klassifikatoren beispielsweise durch *Boosting* und *Bagging*. Neben der Methode, die Textklassifikation auf fastText-Embeddings als Features durchzuführen, wäre es auch sinnvoll für den Fall monolingualer Embeddings den fastText-Klassifizierer (Joulin u. a. 2016) zu verwenden. Des Weiteren könnten andere Features, anstatt lediglich des Durchschnitts der Word Embeddings aller Wörter im Dokument den Klassifikationsverfahren übergeben werden, etwa Features, die anhand der Verlinkungen von und zu einem Artikel gebildet werden.

Es wird deutlich, dass es viele Richtungen gibt, in die diese Arbeit weiterentwickelt werden könnte.

Aufgrund der guten Ergebnisse der Textklassifikation mit dem multilingualen BERT-Modell,

scheint besonders attraktiv zu sein die Texte vieler weiterer Sprachen zu untersuchen. Mit dem entstandenen Code, der generisch ist, können beliebige Sprachen angegeben werden, in denen die Wikipedia-Artikel extrahiert und die Schritte dieser Arbeit durchgeführt werden. Lediglich das Alignment der Worteinbettungen findet auf jeweils zwei Sprachen statt und könnte übersprungen werden. Die Durchführung der Textklassifikation mit BERT-Modellen auf (Wikipedia-)Texten in Sprachen, die das Thema der Musikgeschichte weniger umfassend als die deutsche und die englische Wikipedia dokumentieren und die Evaluierung des besten Modells für diese Sprachen wäre eine interessante Erweiterung. Dabei muss erwähnt werden, dass zum Ende der Bearbeitungszeit für diese Arbeit das Wikipedia Scraping immer wieder von der MediaWiki Api abgebrochen wurde und neue Daten nicht in zu vielen Sprachen auf einmal abgefragt werden sollten.

Letztlich wäre es möglich, weitere Quellen für Personen zu erforschen, die den Korpus anreichern könnten.

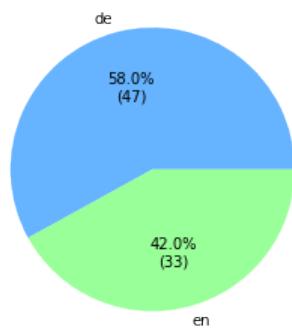
Beispielsweise wurde während der Recherche zu dieser Arbeit das „Handbuch deutsche Musiker 1933-1945“ des bereits in der Einleitung zitierten Musikwissenschaftlers Fred K. Priesberg als frei verfügbarer Volltext¹ der Version aus dem Jahr 2004 online entdeckt. Es werden unter anderem umfangreich Komponisten und Musiker aufgelistet, denen eine Mitgliedschaft in der NSDAP nachgewiesen werden konnte. Der Volltext könnte mit informatischen Methoden eingelesen werden, um die Anzahl der Datensätze in der kleineren Klasse zu erhöhen, was die in dieser Arbeit genutzten Methoden des Over- bzw. Undersamplings überflüssig machen könnte.

Da das Handbuch nur deutsche Texte umfasst, könnte dann erneut die Alignment Methode ausprobiert werden, um Wissen von der ressourcenreichen Sprache Deutsch auf ressourcenärmere Sprachen zu transferieren.

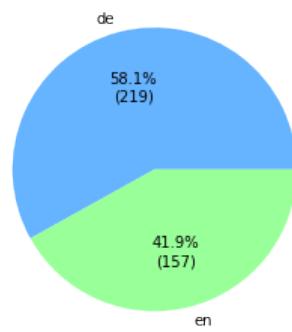
¹https://archive.org/stream/bib130947_001_001/bib130947_001_001_djvu.txt [Letzter Zugriff am 26.11.2022 um 10:39 Uhr]

A. Sprachverteilungen der extrahierten Artikel, Paragraphen, Sätze und Wörter

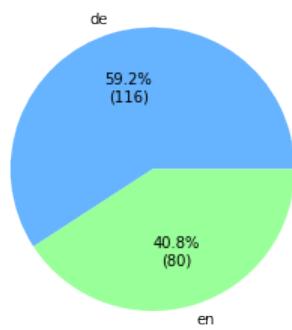
begnadet - Distribution of articles



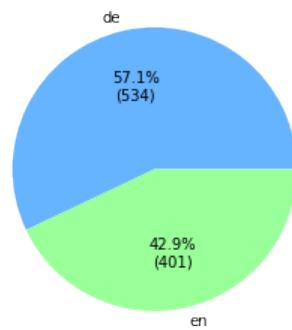
verfolgt - Distribution of articles



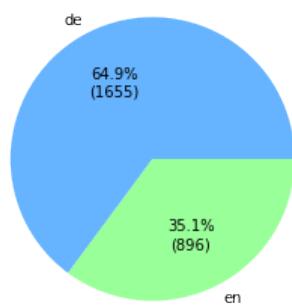
begnadet - Distribution of paragraphs



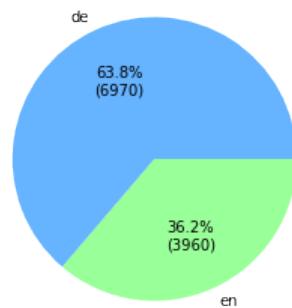
verfolgt - Distribution of paragraphs



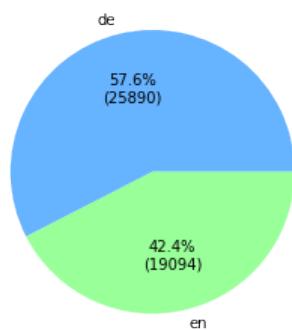
begnadet - Distribution of sentences



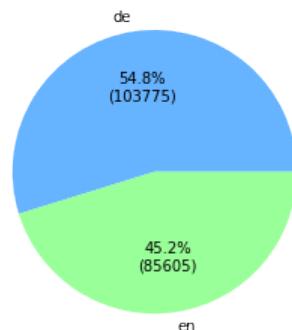
verfolgt - Distribution of sentences



begnadet - Distribution of words



verfolgt - Distribution of words



(a) Sprachverteilungen Klasse „begnadet“.

(b) Sprachverteilungen Klasse „verfolgt“.

B. Teilmenge der von BERTopic generierten Topic-Cluster

```

0: [(['ungarn', 0.0637555381131527),
  ('österreich', 0.05350740553627704),
  ('geboren', 0.047791968313417574),
  ('wien', 0.022368078101795117),
  ('juli', 0.022175839343705282),
  ('november', 0.01995663900677902),
  ('oktober', 0.01961389135328258),
  ('1893', 0.01952354423904368),
  ('18', 0.017988500169485364),
  ('april', 0.017773129221828166)],
1: [(['nationalsozialisten',
  0.038897985602613594),
  ('1933', 0.0301311245259771),
  ('nationalsozialismus', 0.022942919530786005),
  ('wegen', 0.021663266085087462),
  ('jüdischen', 0.0201125310731393),
  ('machtergreifung', 0.01985470879622143),
  ('abstammung', 0.01983864087795909),
  ('deutschland', 0.019647578682573857),
  ('verboten', 0.016943195097820076),
  ('machtübernahme', 0.016860719673347693)],
2: [(['schönberg', 0.07130271262131144),
  ('arnold', 0.054679285417095665),
  ('schönbergs', 0.04591454263339078),
  ('schule', 0.022874726102509173),
  ('schnabel', 0.015603853441206515),
  ('wiener', 0.013556622208074902),
  ('schüler', 0.01256932940774007),
  ('zwölftontechnik', 0.01199801650542621),
  ('schächter', 0.010545424927435267),
  ('berg', 0.010446282369677092)],
3: [(['deutscher', 0.08435763659151817),
  ('komponist', 0.06481370641198983),
  ('musikpädagoge', 0.04041942025728754),
  ('dirigent', 0.033439292248074924),
  ('österreichisch', 0.03284077895904613),
  ('ebenda', 0.028309749028220303),
  ('pianist', 0.02800713202203761),
  ('britischer', 0.026466298249537855),
  ('österreichischer', 0.023002566090340574),
  ('juni', 0.022785379668386497)],
4: [(['deportiert', 0.07518401332857623),
  ('kz', 0.043416172065791826),
  ('auschwitz', 0.04144186205560232),
  ('theresienstadt', 0.03957967875243317),
  ('ermordet', 0.03590475232166855),
  ('1942', 0.031853770740605414),
  ('interniert', 0.03081452959909918),
  ('1943', 0.030226125802303713),
  ('ghetto', 0.02982362048870421),
  ('verhaftet', 0.026165651353836375)],
5: [(['zemlinsky', 0.1200609622115642),
  ('zemlinskys', 0.06014999579691877),
  ('akt', 0.01566967167487192),
  ('musikdirektor', 0.015393431571790865),
  ('neuen', 0.014278858076497886),
  ('prag', 0.013644693262289759),
  ('nachdem', 0.01316170666434307),
  ('volksoper', 0.013046534908220402),
  ('mahagonny', 0.013015404003846012),
  ('zeisl', 0.012234103331295388)]]

```

```

6: [ ('jahren', 0.05187473740532184),
  ('alter', 0.04685803149665023),
  ('klavierunterricht', 0.041378254835822144),
  ('erhielt', 0.034588511807415394),
  ('bartók', 0.03255617538362245),
  ('ersten', 0.027372314719980532),
  ('bereits', 0.027355919690304615),
  ('violinunterricht', 0.0270654097313179),
  ('erziehung', 0.0270654097313179),
  ('elf', 0.02544070235998404)],
7: [ ('jedoch', 0.02480028516808423),
  ('aufgeführt', 0.022484057108758858),
  ('verwehrt', 0.020216865226299327),
  ('mehr', 0.019615907486595436),
  ('blieb', 0.017349045665369506),
  ('weder', 0.016792353667087426),
  ('vergessenheit', 0.015784326777570478),
  ('orchestrieren', 0.014576255185619716),
  ('erschienen', 0.01431938513943831),
  ('durfte', 0.01431938513943831)],
8: [ ('kammermusik', 0.03877402058087664),
  ('sinfonien', 0.03809040806892459),
  ('lieder', 0.03481126544030153),
  ('komponierte', 0.032974013724641135),
  ('orchesterwerke', 0.03136342566174107),
  ('opern', 0.029811804240956974),
  ('orchester', 0.028009380271600295),
  ('ballette', 0.023918726203472098),
  ('darunter', 0.022725534398254192),
  ('umfasst', 0.02153643477853113)],
9: [ ('kompositionen', 0.04001186880089959),
  ('hrsg', 0.02342250159762004),
  ('erfolge', 0.021738217080688897),
  ('eigenen', 0.021034965635428487),
  ('erste', 0.02042364412885158),
  ('theoretischen', 0.017921224968584375),
  ('composition', 0.017921224968584375),
  ('stück', 0.017244625117523775),
  ('zeit', 0.017186581429078363),
  ('literatur', 0.01690951926129746)],
10: [ ('heiratete', 0.09481638736438687),
   ('verheiratet', 0.057933506056170446),
   ('ehe', 0.04242168720840333),
   ('tochter', 0.030802730032435006),
   ('elisabeth', 0.021414891386201552),
   ('schauspielerin', 0.019496096804397466),
   ('zweiter', 0.019496096804397466),
   ('kennen', 0.018607177118544063),
   ('henriette', 0.0181408057165082),
   ('künstlerin', 0.0181408057165082)],
11: [ ('schülern', 0.0441276213000461),
   ('zählten', 0.025970913497276305),
   ('gehörten', 0.017779109990146955),
   ('helmut', 0.016304921879418403),
   ('hans', 0.014480623993166129),
   ('pierre', 0.012632556559514787),
   ('heinz', 0.012631479959131615),
   ('hermann', 0.012452325494409932),
   ('peter', 0.011903829671541223),
   ('robert', 0.011527752393924465)],
12: [ ('jazz', 0.04239897516649728),

```

```

('musik', 0.022572504080873735),
('tonalität', 0.022005120331883635),
('rhythmen', 0.021138431732341464),
('tonaler', 0.019919135974044395),
('harmonik', 0.01897562197776862),
('stil', 0.01787908792712254),
('klassische', 0.016997190049353798),
('neoklassizismus', 0.016503840248912726),
('kompositionen', 0.015879354199331376)],
13: [('new', 0.07775769292812142),
('york', 0.0697056913607017),
('school', 0.05885454697784406),
('music', 0.04444461817218381),
('university', 0.03929058202083286),
('of', 0.03288571923078198),
('philadelphia', 0.03166318024252442),
('city', 0.028654131083425467),
('juilliard', 0.025633752705754284),
('indiana', 0.020507002164603428)],
14: [('hollywood', 0.06938068728397792),
('filmen', 0.04857200158171714),
('filme', 0.04181072270079431),
('filmmusik', 0.03787457712438303),
('stummfilmen', 0.028834718779670806),
('filmmusiken', 0.02573817503543895),
('filmindustrie', 0.025108594037400515),
('original', 0.023660293492012117),
('schrieb', 0.023070496253192723),
('film', 0.022910131957431676)],
15: [('universität', 0.0420781017192808),
('musikakademie', 0.0359977437326269),
('wien', 0.03538049215302486),
('studierte', 0.03262503876108419),
('guido', 0.02998695775277524),
('musikwissenschaft', 0.028728446737443892),
('adler', 0.02826896412709475),
('wiener', 0.027151922368241314),
('studium', 0.02630327527182029),
('promovierte', 0.025543519545761395)],
16: [('schubert', 0.06146293250842803),
('beethoven', 0.04852831690691646),
('mozart', 0.04852831690691646),
('brahms', 0.04436539668603004),
('mozarts', 0.023745328188350977),
('bach', 0.02341626814335825),
('strauss', 0.022374022952625184),
('amadeus', 0.02008951840595937),
('verdi', 0.018396399390510704),
('wolfgang', 0.017406065311625016)],
17: [('pianist', 0.03168755642236947),
('sonnenschein', 0.02726120886495236),
('trat', 0.026970058144907997),
('herz', 0.02600215631785371),
('erscheinung', 0.024755760373369084),
('pianistin', 0.023678240037934745),
('il', 0.02286104979966853),
('klavierspiel', 0.02157047649357817),
('perl', 0.021138431732341464),
('häufig', 0.020113973918012857)],
18: [('preis', 0.0904145527051789),
('ausgezeichnet', 0.07240519717791034),

```

```

('american', 0.05571182670526813),
('stadt', 0.05149317211723213),
('arts', 0.04921327115834458),
('erhielt', 0.047086761313985606),
('academy', 0.045452720508440146),
('and', 0.04374512991852852),
('gewählt', 0.04081320394583501),
('kunstpreis', 0.03951002455180916)],
19: [('oper', 0.0914138598048984),
('manon', 0.037488561538281936),
('opern', 0.037212804435896966),
('operette', 0.03544688003380682),
('puccini', 0.024992374358854624),
('tote', 0.024992374358854624),
('uraufführung', 0.0241254626259401),
('esther', 0.02380855518926465),
('spiel', 0.022843590883090414),
('italienischen', 0.022029647170048565)],
20: [('hoch', 0.04227547811510487),
('konservatorium', 0.03953265801760743),
('schen', 0.03778493828464314),
('studierte', 0.0376604302225629),
('engelbert', 0.033762321789826084),
('humperdinck', 0.033762321789826084),
('komposition', 0.028651943932705638),
('sekles', 0.02680540000236338),
('iwan', 0.02658859606465783),
('klavier', 0.024562762408147722)],
21: [('geboren', 0.04488637694845089),
('familie', 0.04058709642931277),
('odessa', 0.03723518274882899),
('eltern', 0.03695789308905645),
('stammte', 0.03443392959029574),
('sohn', 0.0334167888829802),
('jüdischen', 0.02705806428620851),
('vater', 0.026583971847666966),
('cohnheim', 0.022429709237410632),
('ukraine', 0.022429709237410632)],
22: [('dresdner', 0.03184296020124928),
('leiter', 0.030023240309274276),
('dirigenten', 0.028145916454775335),
('orchester', 0.026304461472459406),
('jahrhunderts', 0.02437786116411174),
('20', 0.023897635225526064),
('verein', 0.02215176148648916),
('stärke', 0.022059734652051282),
('übernahm', 0.02056179534959853),
('leitung', 0.01931777754930267)],
23: [('gemischten', 0.07042714167843389),
('chor', 0.0642356055432098),
('para', 0.05766208159360431),
('klavier', 0.037488527791268454),
('de', 0.0335213275613004),
('vierstimmigen', 0.027174391326792273),
('coros', 0.027174391326792273),
('la', 0.025356490583763745),
('solisten', 0.02520205922928199),
('klarinette', 0.025143997534969754)],
24: [('weltkrieg', 0.06697233848889976),
('meldete', 0.0636676107729909),
('ersten', 0.05595126159841803),

```

```
('1918', 0.05243105977110722),  
('armee', 0.04528761236065026),  
('eingezogen', 0.043147005305009006),  
('militärdienst', 0.042210556997380404),  
('soldat', 0.04106341879225409),  
('1915', 0.037759176725607066),  
('weltkrieges', 0.03612051110950037)],  
25: [('familie', 0.07091591573912889),  
('zog', 0.05731601553449981),  
('vater', 0.03483668838279435),  
('mutter', 0.029176062782131085),  
('übersiedelte', 0.027069313604225702),  
('eltern', 0.0242155425871977),  
('weinhandlung', 0.02351422265108763),  
('main', 0.020690438858619542),  
('fabrik', 0.020488254144731374),  
('tochter', 0.019777577053792652)],  
26: [('kz', 0.07819826893553787),  
('auschwitz', 0.07264649268158153),  
('ghetto', 0.06454982211582191),  
('gestorben', 0.059076826551642675),  
('komponist', 0.05891087505365964),  
('1944', 0.05627839259337023),  
('niederländischer', 0.05139098040966221),  
('polnischer', 0.0481076691362696),  
('theresienstadt', 0.043611627335624965),  
('birkenau', 0.04282581700805184)],
```

C. Vollständige Auswertungen der Textklassifikation mit Support Vector Machine und Naïve Bayes

Tabelle C.1.: Performanz der Textklassifikation mit Support Vector Machine und Naïve Bayes auf dem deutschen Datensatz.

| Embeddings | Features | Res. | NB | | | ACC | SVM | | | ACC |
|------------|---------------|---------|------|------|------|------|------|------|------|------|
| | | | P | R | F1 | | P | R | F1 | |
| W2V | text | - | 0.52 | 0.54 | 0.52 | 0.72 | 0.50 | 0.39 | 0.44 | 0.78 |
| W2V | text | Overs. | 0.61 | 0.58 | 0.52 | 0.54 | 0.56 | 0.62 | 0.31 | 0.31 |
| W2V | text | Unders. | 0.59 | 0.56 | 0.49 | 0.50 | 0.57 | 0.44 | 0.44 | 0.58 |
| W2V | text + topics | - | 0.50 | 0.41 | 0.45 | 0.81 | 0.50 | 0.41 | 0.45 | 0.81 |
| W2V | text + topics | Overs. | 0.45 | 0.49 | 0.41 | 0.54 | 0.44 | 0.48 | 0.40 | 0.52 |
| W2V | text + topics | Unders. | 0.29 | 0.41 | 0.34 | 0.52 | 0.44 | 0.48 | 0.40 | 0.52 |
| fT | text | - | 0.63 | 0.58 | 0.58 | 0.69 | 0.50 | 0.42 | 0.45 | 0.83 |
| fT | text | Overs. | 0.62 | 0.57 | 0.57 | 0.67 | 0.67 | 0.60 | 0.58 | 0.67 |
| fT | text | Unders. | 0.67 | 0.60 | 0.58 | 0.67 | 0.70 | 0.61 | 0.54 | 0.57 |
| fT (a) | text | - | 0.50 | 0.50 | 0.46 | 0.63 | 0.50 | 0.41 | 0.45 | 0.81 |
| fT (a) | text | Overs. | 0.54 | 0.52 | 0.46 | 0.57 | 0.62 | 0.55 | 0.49 | 0.59 |
| fT (a) | text | Unders. | 0.51 | 0.51 | 0.46 | 0.52 | 0.51 | 0.51 | 0.46 | 0.52 |
| fT (p,a) | text | - | 0.50 | 0.42 | 0.45 | 0.83 | 0.50 | 0.42 | 0.45 | 0.83 |
| fT (p,a) | text | Overs. | 0.79 | 0.69 | 0.71 | 0.80 | 0.83 | 0.77 | 0.79 | 0.87 |
| fT (p,a) | text | Unders. | 0.82 | 0.70 | 0.71 | 0.78 | 0.71 | 0.62 | 0.55 | 0.56 |
| fT (p, a) | text + topics | - | 0.57 | 0.66 | 0.57 | 0.80 | 0.50 | 0.40 | 0.44 | 0.80 |
| fT (p, a) | text + topics | Overs. | 0.63 | 0.60 | 0.60 | 0.69 | 0.74 | 0.67 | 0.68 | 0.74 |
| fT (p, a) | text + topics | Unders. | 0.60 | 0.57 | 0.56 | 0.63 | 0.64 | 0.61 | 0.47 | 0.48 |

Tabelle C.2.: Performanz der Textklassifikation mit Support Vector Machine und Naïve Bayes auf dem englischen Datensatz.

| Embeddings | Features | Res. | NB | | | ACC | SVM | | | ACC |
|------------|---------------|---------|------|------|------|------|------|------|------|------|
| | | | P | R | F1 | | P | R | F1 | |
| W2V | text | - | 0.48 | 0.38 | 0.43 | 0.74 | 0.50 | 0.38 | 0.43 | 0.77 |
| W2V | text | Overs. | 0.53 | 0.53 | 0.53 | 0.64 | 0.50 | 0.50 | 0.40 | 0.41 |
| W2V | text | Unders. | 0.54 | 0.53 | 0.50 | 0.54 | 0.55 | 0.54 | 0.47 | 0.49 |
| W2V | text + topics | - | 0.50 | 0.42 | 0.46 | 0.85 | 0.50 | 0.42 | 0.46 | 0.85 |
| W2V | text + topics | Overs. | 0.66 | 0.58 | 0.50 | 0.54 | 0.59 | 0.59 | 0.31 | 0.31 |
| W2V | text + topics | Unders. | 0.58 | 0.54 | 0.46 | 0.51 | 0.50 | 0.08 | 0.13 | 0.15 |
| fT | text | - | 0.50 | 0.38 | 0.43 | 0.77 | 0.50 | 0.38 | 0.43 | 0.77 |
| fT | text | Overs. | 0.52 | 0.51 | 0.43 | 0.44 | 0.52 | 0.51 | 0.43 | 0.44 |
| fT | text | Unders. | 0.53 | 0.52 | 0.48 | 0.51 | 0.52 | 0.51 | 0.43 | 0.44 |
| fT (a) | text | - | 0.50 | 0.38 | 0.43 | 0.77 | 0.40 | 0.38 | 0.43 | 0.77 |
| fT (a) | text | Overs. | 0.57 | 0.56 | 0.46 | 0.46 | 0.53 | 0.62 | 0.26 | 0.28 |
| fT (a) | text | Unders. | 0.51 | 0.51 | 0.46 | 0.49 | 0.57 | 0.56 | 0.46 | 0.46 |
| fT (p,a) | text | - | 0.50 | 0.41 | 0.45 | 0.82 | 0.50 | 0.41 | 0.45 | 0.81 |
| fT (p,a) | text | Overs. | 0.65 | 0.65 | 0.65 | 0.79 | 0.62 | 0.60 | 0.61 | 0.74 |
| fT (p,a) | text | Unders. | 0.62 | 0.60 | 0.61 | 0.74 | 0.46 | 0.48 | 0.41 | 0.49 |
| fT (p, a) | text + topics | - | 0.50 | 0.41 | 0.45 | 0.82 | 0.50 | 0.41 | 0.45 | 0.82 |
| fT (p, a) | text + topics | Overs. | 0.59 | 0.56 | 0.57 | 0.69 | 0.75 | 0.67 | 0.69 | 0.77 |
| fT (p, a) | text + topics | Unders. | 0.65 | 0.60 | 0.60 | 0.69 | 0.61 | 0.61 | 0.36 | 0.36 |

Tabelle C.3.: Performanz der Textklassifikation mit Support Vector Machine und Naïve Bayes auf dem multilingualen Datensatz.

| Embeddings | Features | Res. | NB | | | ACC | SVM | | | ACC |
|------------|---------------|---------|------|------|------|------|------|------|------|------|
| | | | P | R | F1 | | P | R | F1 | |
| W2V | text | - | 0.47 | 0.48 | 0.47 | 0.65 | 0.50 | 0.42 | 0.46 | 0.85 |
| W2V | text | Overs. | 0.50 | 0.50 | 0.45 | 0.54 | 0.47 | 0.48 | 0.44 | 0.54 |
| W2V | text | Unders. | 0.50 | 0.50 | 0.46 | 0.55 | 0.43 | 0.47 | 0.40 | 0.49 |
| W2V | text + topics | - | 0.48 | 0.48 | 0.48 | 0.77 | 0.50 | 0.42 | 0.46 | 0.85 |
| W2V | text + topics | Overs. | 0.53 | 0.52 | 0.47 | 0.55 | 0.53 | 0.52 | 0.47 | 0.55 |
| W2V | text + topics | Unders. | 0.61 | 0.56 | 0.51 | 0.59 | 0.39 | 0.44 | 0.40 | 0.57 |
| fT | text | - | 0.55 | 0.53 | 0.51 | 0.64 | 0.50 | 0.42 | 0.46 | 0.85 |
| fT | text | Overs. | 0.60 | 0.55 | 0.50 | 0.57 | 0.64 | 0.57 | 0.50 | 0.54 |
| fT | text | Unders. | 0.61 | 0.56 | 0.51 | 0.59 | 0.64 | 0.57 | 0.50 | 0.54 |
| fT (a) | text | - | 0.52 | 0.51 | 0.52 | 0.73 | 0.50 | 0.42 | 0.46 | 0.85 |
| fT (a) | text | Overs. | 0.52 | 0.51 | 0.43 | 0.49 | 0.59 | 0.57 | 0.35 | 0.35 |
| fT (a) | text | Unders. | 0.52 | 0.51 | 0.42 | 0.8 | 0.60 | 0.59 | 0.33 | 0.33 |
| fT (p,a) | text | - | 0.50 | 0.43 | 0.46 | 0.86 | 0.50 | 0.43 | 0.46 | 0.86 |
| fT (p,a) | text | Overs. | 0.59 | 0.55 | 0.54 | 0.68 | 0.69 | 0.60 | 0.59 | 0.68 |
| fT (p,a) | text | Unders. | 0.45 | 0.48 | 0.41 | 0.50 | 0.57 | 0.53 | 0.46 | 0.53 |
| fT (p, a) | text + topics | - | 0.56 | 0.58 | 0.57 | 0.80 | 0.50 | 0.42 | 0.46 | 0.85 |
| fT (p, a) | text + topics | Overs. | 0.46 | 0.48 | 0.41 | 0.48 | 0.62 | 0.59 | 0.59 | 0.75 |
| fT (p, a) | text + topics | Unders. | 0.42 | 0.46 | 0.36 | 0.41 | 0.48 | 0.49 | 0.44 | 0.52 |

D. Vollständige Auswertungen der Textklassifikation mit BERT Modellen

Tabelle D.1.: Konfiguration und Performanz von gBERT auf dem deutschen Datensatz.

| Epochen | Batchgröße | Lernrate | Loss | P | R | F1 | Acc |
|---------|------------|----------|------|-------------|-------------|-------------|-------------|
| 1 | 8 | 3,00E-05 | 0.51 | 0.34 | 0.50 | 0.44 | 0.78 |
| 3 | 8 | 3,00E-05 | 0.40 | 0.85 | 0.94 | 0.87 | 0.90 |
| 5 | 8 | 3,00E-05 | 0.23 | 0.88 | 0.95 | 0.90 | 0.93 |
| 20 | 8 | 3,00E-05 | 0.07 | 0.91 | 0.97 | 0.93 | 0.95 |
| 1 | 16 | 3,00E-05 | 0.52 | 0.39 | 0.50 | 0.44 | 0.78 |
| 3 | 16 | 3,00E-05 | 0.50 | 0.39 | 0.50 | 0.44 | 0.78 |
| 5 | 16 | 3,00E-05 | 0.34 | 0.88 | 0.95 | 0.90 | 0.93 |
| 20 | 16 | 3,00E-05 | 0.10 | 0.91 | 0.97 | 0.93 | 0.95 |
| 5 | 8 | 5,00E-05 | 0.26 | 0.86 | 0.86 | 0.86 | 0.90 |
| 20 | 8 | 5,00E-05 | 0.07 | 0.88 | 0.95 | 0.90 | 0.93 |
| 1 | 16 | 5,00E-05 | 0.56 | 0.39 | 0.50 | 0.44 | 0.78 |
| 3 | 16 | 5,00E-05 | 0.48 | 0.39 | 0.50 | 0.44 | 0.78 |
| 5 | 16 | 5,00E-05 | 0.35 | 0.98 | 0.94 | 0.96 | 0.97 |
| 20 | 16 | 5,00E-05 | 0.09 | 0.82 | 0.92 | 0.85 | 0.88 |

Tabelle D.2.: Konfiguration und Performanz von mBERT auf dem deutschen Datensatz.

| Epochen | Batchgröße | Lernrate | Loss | P | R | F1 | Acc |
|---------|------------|----------|------|------|------|------|------|
| 1 | 8 | 3,00E-05 | 0.55 | 0.39 | 0.50 | 0.44 | 0.78 |
| 3 | 8 | 3,00E-05 | 0.50 | 0.39 | 0.50 | 0.44 | 0.78 |
| 5 | 8 | 3,00E-05 | 0.39 | 0.74 | 0.60 | 0.61 | 0.80 |
| 20 | 8 | 3,00E-05 | 0.44 | 0.39 | 0.50 | 0.44 | 0.78 |
| 1 | 16 | 3,00E-05 | 0.64 | 0.39 | 0.50 | 0.44 | 0.78 |
| 3 | 16 | 3,00E-05 | 0.57 | 0.39 | 0.50 | 0.44 | 0.78 |
| 5 | 16 | 3,00E-05 | 0.65 | 0.82 | 0.92 | 0.85 | 0.88 |
| 20 | 16 | 3,00E-05 | 0.13 | 0.56 | 0.52 | 0.51 | 0.75 |
| 5 | 8 | 5,00E-05 | 0.35 | 0.80 | 0.90 | 0.82 | 0.85 |
| 20 | 8 | 5,00E-05 | 0.10 | 0.83 | 0.80 | 0.81 | 0.88 |
| 1 | 16 | 5,00E-05 | 0.73 | 0.39 | 0.50 | 0.44 | 0.78 |
| 3 | 16 | 5,00E-05 | 0.43 | 0.39 | 0.50 | 0.44 | 0.78 |
| 5 | 16 | 5,00E-05 | 0.42 | 0.39 | 0.50 | 0.44 | 0.78 |
| 20 | 16 | 5,00E-05 | 0.13 | 0.85 | 0.90 | 0.87 | 0.90 |

Tabelle D.3.: Konfiguration und Performanz von eBERT auf dem englischen Datensatz.

| Epochen | Batchgröße | Lernrate | Loss | P | R | F1 | Acc |
|---------|------------|----------|------|-------------|-------------|-------------|-------------|
| 1 | 8 | 3,00E-05 | 0.53 | 0.47 | 0.50 | 0.48 | 0.93 |
| 3 | 8 | 3,00E-05 | 0.58 | 0.47 | 0.50 | 0.48 | 0.93 |
| 5 | 8 | 3,00E-05 | 0.41 | 0.47 | 0.50 | 0.48 | 0.93 |
| 20 | 8 | 3,00E-05 | 0.11 | 0.75 | 0.96 | 0.81 | 0.93 |
| 1 | 16 | 3,00E-05 | 0.99 | 0.3 | 0.5 | 0.06 | 0.07 |
| 3 | 16 | 3,00E-05 | 0.66 | 0.47 | 0.50 | 0.48 | 0.93 |
| 5 | 16 | 3,00E-05 | 0.62 | 0.47 | 0.50 | 0.48 | 0.93 |
| 20 | 16 | 3,00E-05 | 0.19 | 0.73 | 0.73 | 0.73 | 0.93 |
| 5 | 8 | 5,00E-05 | 0.45 | 0.47 | 0.50 | 0.48 | 0.93 |
| 20 | 8 | 5,00E-05 | 0.13 | 0.65 | 0.71 | 0.67 | 0.90 |
| 1 | 16 | 5,00E-05 | 0.93 | 0.03 | 0.50 | 0.06 | 0.07 |
| 3 | 16 | 5,00E-05 | 0.65 | 0.47 | 0.50 | 0.48 | 0.93 |
| 5 | 16 | 5,00E-05 | 0.61 | 0.58 | 0.68 | 0.59 | 0.83 |
| 20 | 16 | 5,00E-05 | 0.17 | 0.73 | 0.73 | 0.73 | 0.93 |

Tabelle D.4.: Konfiguration und Performanz von mBERT auf dem englischen Datensatz.

| Epochen | Batchgröße | Lernrate | Loss | P | R | F1 | Acc |
|---------|------------|----------|------|------|------|------|------|
| 1 | 8 | 3,00E-05 | 0.58 | 0.47 | 0.50 | 0.48 | 0.93 |
| 3 | 8 | 3,00E-05 | 0.47 | 0.47 | 0.50 | 0.48 | 0.93 |
| 5 | 8 | 3,00E-05 | 0.46 | 0.73 | 0.73 | 0.73 | 0.93 |
| 20 | 8 | 3,00E-05 | 0.18 | 0.73 | 0.73 | 0.73 | 0.93 |
| 1 | 16 | 3,00E-05 | 0.70 | 0.47 | 0.50 | 0.48 | 0.93 |
| 3 | 16 | 3,00E-05 | 0.54 | 0.46 | 0.44 | 0.45 | 0.83 |
| 5 | 16 | 3,00E-05 | 0.57 | 0.47 | 0.50 | 0.48 | 0.93 |
| 20 | 16 | 3,00E-05 | 0.20 | 0.65 | 0.71 | 0.67 | 0.90 |
| 5 | 8 | 5,00E-05 | 0.44 | 0.46 | 0.44 | 0.45 | 0.83 |
| 20 | 8 | 5,00E-05 | 0.13 | 0.60 | 0.69 | 0.63 | 0.86 |
| 1 | 16 | 5,00E-05 | 0.68 | 0.46 | 0.46 | 0.46 | 0.86 |
| 3 | 16 | 5,00E-05 | 0.60 | 0.47 | 0.50 | 0.48 | 0.93 |
| 5 | 16 | 5,00E-05 | 0.55 | 0.47 | 0.50 | 0.48 | 0.93 |
| 20 | 16 | 5,00E-05 | 0.21 | 0.47 | 0.50 | 0.48 | 0.93 |

Tabelle D.5.: Konfiguration und Performanz von mBERT auf dem multilingualen Datensatz.

| Epochen | Batchgröße | Lernrate | Loss | P | R | F1 | Acc |
|---------|------------|----------|------|-------------|-------------|-------------|-------------|
| 1 | 8 | 3,00E-05 | 0.51 | 0.41 | 0.50 | 0.45 | 0.81 |
| 3 | 8 | 3,00E-05 | 0.48 | 0.93 | 0.65 | 0.70 | 0.87 |
| 5 | 8 | 3,00E-05 | 0.38 | 0.85 | 0.89 | 0.89 | 0.91 |
| 15 | 8 | 3,00E-05 | 0.12 | 0.88 | 0.96 | 0.92 | 0.94 |
| 20 | 8 | 3,00E-05 | 0.38 | 0.88 | 0.96 | 0.92 | 0.94 |
| 1 | 16 | 3,00E-05 | 0.63 | 0.41 | 0.50 | 0.45 | 0.81 |
| 3 | 16 | 3,00E-05 | 0.44 | 0.83 | 0.94 | 0.86 | 0.90 |
| 5 | 16 | 3,00E-05 | 0.46 | 0.93 | 0.98 | 0.96 | 0.97 |
| 15 | 16 | 3,00E-05 | 0.14 | 0.89 | 0.88 | 0.88 | 0.93 |
| 20 | 16 | 3,00E-05 | 0.11 | 0.89 | 0.94 | 0.91 | 0.94 |
| 30 | 16 | 3,00E-05 | 0.08 | 0.86 | 0.86 | 0.86 | 0.91 |
| 5 | 8 | 5,00E-05 | 0.30 | 0.86 | 0.96 | 0.90 | 0.93 |
| 15 | 8 | 5,00E-05 | 0.15 | 0.85 | 0.89 | 0.87 | 0.91 |
| 20 | 8 | 5,00E-05 | 0.10 | 0.93 | 0.98 | 0.96 | 0.97 |
| 30 | 8 | 5,00E-05 | 0.14 | 0.91 | 0.94 | 0.91 | 0.94 |
| 1 | 16 | 5,00E-05 | 0.52 | 0.41 | 0.50 | 0.45 | 0.81 |
| 3 | 16 | 5,00E-05 | 0.44 | 0.83 | 0.94 | 0.86 | 0.90 |
| 5 | 16 | 5,00E-05 | 0.33 | 0.88 | 0.96 | 0.2 | 0.94 |
| 10 | 16 | 5,00E-05 | 0.20 | 0.93 | 0.88 | 0.90 | 0.94 |
| 15 | 16 | 5,00E-05 | 0.18 | 0.86 | 0.96 | 0.90 | 0.93 |
| 20 | 16 | 5,00E-05 | 0.16 | 0.98 | 0.89 | 0.92 | 0.96 |
| 30 | 16 | 5,00E-05 | 0.09 | 0.92 | 0.94 | 0.93 | 0.96 |

Literatur

- Adafre, Sisay Fissaha und Maarten De Rijke (2006). „Finding similar sentences across multiple languages in Wikipedia“. In: *Proceedings of the Workshop on NEW TEXT Wikis and blogs and other dynamic text sources*.
- Alhaj, Fatima, Ali Al-Haj, Ahmad Sharieh und Riad Jabri (2022). „Improving Arabic cognitive distortion classification in Twitter using BERTopic“. In: *International Journal of Advanced Computer Science and Applications* 13.1, S. 854–860.
- Artetxe, Mikel, Gorka Labaka und Eneko Agirre (Juli 2017). „Learning bilingual word embeddings with (almost) no bilingual data“. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, S. 451–462. DOI: 10.18653/v1/P17-1042. URL: <https://aclanthology.org/P17-1042>.
- Barbieri, Francesco, Luis Espinosa Anke und Jose Camacho-Collados (2021). *XLM-T: Multilingual Language Models in Twitter for Sentiment Analysis and Beyond*. DOI: 10.48550/ARXIV.2104.12250. URL: <https://arxiv.org/abs/2104.12250>.
- Bird, Steven, Ewan Klein und Edward Loper (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.”
- Bojanowski, Piotr, Edouard Grave, Armand Joulin und Tomás Mikolov (2016). „Enriching Word Vectors with Subword Information“. In: *CoRR* abs/1607.04606. arXiv: 1607 . 04606. URL: <http://arxiv.org/abs/1607.04606>.
- Chan, Branden, Timo Möller, Malte Pietsch und Tanay Soni (Juni 2019). *German BERT*. [Online; Veröffentlicht am 14. Juni 2019; Letzter Zugriff am 25. November um 13:32 Uhr]. URL: <https://www.deepset.ai/german-bert>.
- Conneau, Alexis, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer und Hervé Jégou (2017). „Word translation without parallel data“. In: *arXiv preprint arXiv:1710.04087*.
- Devlin, Jacob (2019). *Multilingual Models*. [Online; Letzter Commit am 17. Oktober 2019; Letzter Zugriff am 23. November 2022 um 23:55 Uhr]. URL: <https://github.com/google-research/bert/blob/master/multilingual.md#list-of-languages>.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee und Kristina Toutanova (Juni 2019). „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, S. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: <https://aclanthology.org/N19-1423>.
- Dua, Mohit, Rohit Yadav, Divya Mamgai und Sonali Brodiya (2020). „An Improved RNN-LSTM based Novel Approach for Sheet Music Generation“. In: *Procedia Computer Science* 171. Third International Conference on Computing and Network Communications (CoCoNet'19), S. 465–474. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2020.04.049>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050920310152>.

- Dümling, Albrecht (2015). *Das verdächtige Saxophon: „Entartete Musik“ im NS-Staat - Dokumentation und Kommentar*. 5. Aufl. ConBrio. ISBN: 978-3-940768-52-0.
- Google LLC (30. März 2020). *G Suite*. URL: <https://gsuite.google.com>.
- Grootendorst, Maarten (2022). „BERTopic: Neural topic modeling with a class-based TF-IDF procedure“. In: *arXiv preprint arXiv:2203.05794*.
- Harris, Charles R. u. a. (Sep. 2020). „Array programming with NumPy“. In: *Nature* 585.7825, S. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: <https://doi.org/10.1038/s41586-020-2649-2>.
- Hassibi, Jessica Jasmin (Nov. 2022). *Multi- und monolinguale Textklassifikation von Wikipedia-Artikeln zu einem Thema der Musikgeschichte (Bachelorarbeit Repository)*. Version 1.0. URL: <https://github.com/jessicajhassibi/Bachelor-Thesis>.
- Honnibal, Matthew und Ines Montani (2017). „spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing“.
- Hunter, John D. (2007). „Matplotlib: A 2D graphics environment“. In: *Computing in science & engineering* 9.03, S. 90–95.
- Ikonomakis, Emmanouil, Sotiris Kotsiantis und V. Tampakas (Aug. 2005). „Text Classification Using Machine Learning Techniques“. In: *WSEAS transactions on computers* 4, S. 966–974.
- Jiang, Jun u. a. (2019). „Cross-lingual data transformation and combination for text classification“. In: *arXiv preprint arXiv:1906.09543*.
- Joulin, Armand, Edouard Grave, Piotr Bojanowski und Tomas Mikolov (2016). *Bag of Tricks for Efficient Text Classification*. DOI: 10.48550/ARXIV.1607.01759. URL: <https://arxiv.org/abs/1607.01759>.
- Jurafsky, Daniel und James H. Martin (2009). *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. ISBN: 0131873210.
- Kirschenbaum, Matthew G. (2007). „The Remaking of Reading : Data Mining and the Digital Humanities“. In.
- Kowsari u. a. (Apr. 2019). „Text Classification Algorithms: A Survey“. In: *Information* 10.4, S. 150. DOI: 10.3390/info10040150. URL: <https://doi.org/10.3390%2Finf010040150>.
- Leevy, Joffrey L., Taghi M. Khoshgoftaar, Richard A. Bauder und Naeem Seliya (Nov. 2018). „A survey on addressing high-class imbalance in big data“. In: *Journal of Big Data* 5.1, S. 42. ISSN: 2196-1115. DOI: 10.1186/s40537-018-0151-6. URL: <https://doi.org/10.1186/s40537-018-0151-6>.
- Lemaître, Guillaume, Fernando Nogueira und Christos K. Aridas (2017). „Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning“. In: *Journal of Machine Learning Research* 18.17, S. 1–5. URL: <http://jmlr.org/papers/v18/16-365.html>.
- Liu, Zhiyuan, Yankai Lin und Maosong Sun (2020). „Word Representation“. In: *Representation Learning for Natural Language Processing*. Singapore: Springer Singapore, S. 13–41. ISBN: 978-981-15-5573-2. DOI: 10.1007/978-981-15-5573-2_2. URL: https://doi.org/10.1007/978-981-15-5573-2_2.
- Majlis, Martin (2017). *Wikipedia-API*. [Online; Letzter Zugriff am 14. November 2022 um 18:00 Uhr]. URL: <https://github.com/martin-majlis/Wikipedia-API/>.

- Manning, Christopher D., Prabhakar Raghavan und Hinrich Schütze (2008). *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press. ISBN: 978-0-521-86571-5. URL: <http://nlp.stanford.edu/IR-book/information-retrieval-book.html>.
- McInnes, Leland, John Healy und Steve Astels (2017). „hdbscan: Hierarchical density based clustering.“ In: *J. Open Source Softw.* 2.11, S. 205.
- McInnes, Leland, John Healy und James Melville (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. DOI: 10.48550/ARXIV.1802.03426. URL: <https://arxiv.org/abs/1802.03426>.
- McKinney, Wes (2010). „Data Structures for Statistical Computing in Python“. In: *Proceedings of the 9th Python in Science Conference*. Hrsg. von Stéfan van der Walt und Jarrod Millman, S. 51–56.
- Mikolov, Tomas, Kai Chen, Greg Corrado und Jeffrey Dean (2013). *Efficient Estimation of Word Representations in Vector Space*. DOI: 10.48550/ARXIV.1301.3781. URL: <https://arxiv.org/abs/1301.3781>.
- Mikolov, Tomas, Edouard Grave, Piotr Bojanowski, Christian Puhrsch und Armand Joulin (2018). „Advances in Pre-Training Distributed Word Representations“. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.
- Mikolov, Tomas, Quoc V Le und Ilya Sutskever (2013). „Exploiting similarities among languages for machine translation“. In: *arXiv preprint arXiv:1309.4168*.
- Mikolov, Tomas, Wen-tau Yih und Geoffrey Zweig (Juni 2013). „Linguistic Regularities in Continuous Space Word Representations“. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, S. 746–751. URL: <https://aclanthology.org/N13-1090>.
- Mutuvi, Stephen u. a. (2020). „Multilingual epidemiological text classification: a comparative study“. In: *COLING, International Conference on Computational Linguistics*, S. 6172–6183.
- Pedregosa, F. u. a. (2011). „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12, S. 2825–2830.
- Pennington, Jeffrey, Richard Socher und Christopher D Manning (2014). „Glove: Global vectors for word representation“. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, S. 1532–1543.
- Peters, Matthew E. u. a. (2018). *Deep contextualized word representations*. DOI: 10.48550/ARXIV.1802.05365. URL: <https://arxiv.org/abs/1802.05365>.
- Prieberg, Fred K. (1982). *Musik im NS-Staat*. Fischer Taschenbücher, Bd. 6901. Fischer Taschenbuch Verlag. ISBN: 9783596269013. URL: <https://books.google.de/books?id=VwKgAAAAMAAJ>.
- Rehurek, Radim und Petr Sojka (2011). „Gensim—python framework for vector space modelling“. In: *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3.2.
- Reimers, Nils und Iryna Gurevych (Nov. 2019). „Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks“. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. URL: <https://arxiv.org/abs/1908.10084>.

- Rösinger, Christiane (2007). „Chicks gegen den Krieg“. In: *taz.de*. [Online; Veröffentlicht am 9. August 2007 um 2:00 Uhr; Letzter Zugriff am 3. November 2022 um 14:20 Uhr]. URL: <https://taz.de/Country-Doku/!5196917/>.
- Ruder, Sebastian, Ivan Vulić und Anders Søgaard (2019). „A survey of cross-lingual word embedding models“. In: *Journal of Artificial Intelligence Research* 65, S. 569–631.
- Sahle, Patrick (Nov. 2011). *Was sind die digitalen Geisteswissenschaften?* Broschüre. URL: https://dig-hum.de/sites/dig-hum.de/files/cceh_broschuereweb.pdf.
- Strube, Michael und Simone Paolo Ponzetto (2006). „WikiRelate! Computing semantic relatedness using Wikipedia“. In: *AAAI*. Bd. 6, S. 1419–1424.
- Suissa, Omri, Avshalom Elmalech und Maayan Zhitomirsky-Geffet (2022). „Text analysis using deep neural networks in digital humanities and information science“. In: *Journal of the Association for Information Science and Technology* 73.2, S. 268–287. DOI: <https://doi.org/10.1002/asi.24544>. URL: <https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.24544>.
- Vaswani, Ashish u. a. (2017). *Attention Is All You Need*. DOI: 10.48550/ARXIV.1706.03762. URL: <https://arxiv.org/abs/1706.03762>.
- Velankar, Abhishek, Hrushikesh Patil und Raviraj Joshi (Nov. 2022). „Mono vs Multilingual BERT for Hate Speech Detection and Text Classification: A Case Study in Marathi“. In: *Artificial Neural Networks in Pattern Recognition*. Springer International Publishing, S. 121–128. DOI: 10.1007/978-3-031-20650-4_10. URL: https://doi.org/10.1007%2F978-3-031-20650-4_10.
- Wikipedia (2022a). *Franz Schreker*. [Online; Zuletzt bearbeitet am 4. Juni 2022 um 15:11 Uhr; Letzter Zugriff am 14. November 2022 um 23:18 Uhr]. URL: https://de.wikipedia.org/wiki/Franz_Schreker.
- (2022b). *Gottbegnadeten-Liste*. [Online; Zuletzt bearbeitet am 2. November 2022 um 16:32 Uhr; Letzter Zugriff am 3. November 2022 um 14:01 Uhr]. URL: <https://de.wikipedia.org/wiki/Gottbegnadeten-Liste>.
 - (2022c). *Liste der vom NS-Regime oder seinen Verbündeten verfolgten Komponisten*. [Online; Zuletzt bearbeitet am 7. April 2022 um 14:46 Uhr; Letzter Zugriff am 10. November 2022 um 14:01 Uhr]. URL: https://de.wikipedia.org/wiki/Liste_der_vom_NS-Regime_oder_seinen_Verb%C3%BCndeten_verfolgten_Komponisten.
 - (2022d). *Wikipedia*. [Online; Zuletzt bearbeitet am 2. November 2022 um 16:16 Uhr; Letzter Zugriff am 5. November 2022 um 22:58 Uhr]. URL: <https://de.wikipedia.org/wiki/Wikipedia>.
- Wolf, Thomas u. a. (2019). *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. DOI: 10.48550/ARXIV.1910.03771. URL: <https://arxiv.org/abs/1910.03771>.