



Erstellung eines multilingualen Text- und Bildabgleichmodells durch Contrastive Language-Image Pre-Training für die Bearbeitung der Wikipedia – Image/Caption Competition

Masterarbeit

zur

Erlangung des akademischen Grades

M. Sc.

Fakultät für Informatik

Professur Medieninformatik

Eingereicht von: Marten Rogall

Matrikel Nr.: 461832

Einreichungsdatum: Chemnitz, 11.11.2022

Betreuer: Prof. Dr. Maximilian Eibl

Dr. Stefan Kahl

Kurzzusammenfassung

Ein Großteil aller Bilder zu Artikeln auf Wikipedia ist mit keinem beschreibenden Kontext verknüpft. Dies ist ein Problem für die Informationsvermittlung im Web, aber auch für Menschen mit Sehbehinderung, die auf die Verwendung von Screenreader Software angewiesen sind. Um dieses Problem zu lösen, hat die Wikimedia Foundation die Wikipedia – Image/Caption Competition erstellt. Ziel dieses Wettbewerbs ist es, mit dem bereitgestellten WIT Datensatz ein multimodales Machine Learning Modell zu erstellen, das es ermöglicht, multilinguale Captions und Bilder zuzuordnen. In dieser Arbeit wird ein multilinguales Bild- und Textabgleichmodell auf Basis der monolingualen CLIP-Architektur entwickelt, das es ermöglicht das Ziel des Wettbewerbs zu verfolgen. Hierfür werden zunächst die theoretischen Grundlagen der NLP-Forschung dargestellt, die es ermöglichen, ein solches Modell zu erstellen. Die weitere Arbeit gliedert sich daraufhin in einen Konzeptionsabschnitt und einen daraus resultierenden Implementationabschnitt. Der Ablauf beider Abschnitte ist an das CRISP-DM Prozessmodell angelehnt. Hier wird als erster Schritt das Verstehen, das Bereinigen und das Bereitstellen des Datensatzes für das Training beschrieben. Als Nächstes wird, um einen Ausgangspunkt zu haben, mit dem CLIP-Modell eine Benchmark durch die Zero Shot Klassifikation vorgenommen. Daraufhin wird die Erstellung und das Training des multilingualen, multimodalen Modells durch die Verwendung des XLM-RoBERTa (XLM-R) Modells als Text Encoder und des CLIP eigenen RN50x4 Modells als Image Encoder erläutert. Als weiterer Optimierungsschritt, um die Benchmark zu verbessern, wird das multilinguale Sentence Transformer Modell verwendet, das von allen Modellen in dieser Arbeit am besten abschneidet. Weiterhin wird die Auswirkung von Prompt Engineering auf die Modelle untersucht. Um die Performanz der Modelle vergleichbar zu machen und zu evaluieren, werden sie durch die Top k Accuracy bewertet. Als Ergebnis dieser Arbeit wird gezeigt, dass durch die Verwendung eines multilingualen Dual Encoder Systems die Ziele des Wettbewerbs erfolgreich bearbeitet werden können.

Stichworte: Deep Learning, Natural Language Processing, Contrastive Language-Image Pre-Training(CLIP), Wikipedia-based Image Text Dataset(WIT)

Inhaltsverzeichnis

1 Einleitung	6
1.1 Ziel der Arbeit	7
1.2 Aufbau der Arbeit	8
1.3 Stand der Forschung	9
2. Grundlagen	11
2.1 WIT Datensatz	13
2.1.1 Datensatz des Wettbewerbs	15
2.2 Natural Language Processing	17
2.2.1 Transformer	20
2.2.2 Sprachmodelle, Pre-Training und Finetuning	25
2.3 CLIP	30
2.4 Multilinguale, multimodale Modelle	34
3. Konzeption	36
3.1 CRISP-DM	36
3.2 Data Understanding	39
3.3 Data Preparation	41
3.4 Benchmark Zero Shot CLIP	42
3.5 Modelling des multilingualen Modells	44
3.5.1 Training	46
3.6 Optimierungsstrategien	49
3.6.1 Sentence Transformer Modell	49
3.6.2 Multilinguale Prompt Engineering	51
3.7 Evaluation	53
4. Implementation	54
4.1 Data Understanding	58
4.1.1 Data Preparation	59
4.1.2 Data Visualization	61
4.3 Benchmark Zero Shot CLIP	65
4.3.1 Ergebnis	66
4.4 Multilinguale Modell	67
4.4.1 Training	69
4.4.2 Ergebnis	70
4.5 Optimierungsstrategien	71
4.5.1 Sentence Transformer Modell	71
4.5.2 Multilinguale Prompt Engineering	72
4.5.3 Ergebnis	73
5. Evaluation	75
6. Fazit	76
6.1 Limitation und Bias	78
6.2 Ausblick	79
Literaturverzeichnis	80
Selbstständigkeitserklärung	91

Abbildungsverzeichnis

Abbildung 1	7
Abbildung 2	13
Abbildung 3	16
Abbildung 4	21
Abbildung 5	24
Abbildung 6	27
Abbildung 7	28
Abbildung 8	32
Abbildung 9	37
Abbildung 10	43
Abbildung 11	50
Abbildung 12	51
Abbildung 13	53
Abbildung 14	60
Abbildung 15	60
Abbildung 16	61
Abbildung 17	62
Abbildung 18	63
Abbildung 19	64
Abbildung 20	66
Abbildung 21	69
Abbildung 22	70
Abbildung 23	74

Tabellenverzeichnis

Tabelle 1	75
-----------------	----

Abkürzungsverzeichnis

CLIP	-	Contrastive Language-Image Pre-Training
NLP	-	Natural Language Processing
CV	-	Computer Vision
DCG	-	Discounted Cumulative Gain
NDCG	-	Normalized Discounted Cumulative Gain
WIT	-	Wikipedia-based Image Text Dataset
RNN	-	Recurrent Neural Network
CNN	-	Convolutional Neural Network
MLM	-	Mask Modelling
NSP	-	Next Sentence Prediction
MMT	-	Multilingual Machine Translation
LSTM	-	Long short-term memory

1 Einleitung

Die Beschreibung von Bildern dient online nicht nur als zusätzliche Information zur Verbesserung von Sucheigenschaften, sondern auch als Mittel zur Reduzierung von Barrieren für Menschen mit Sehbehinderungen. Die zusätzlichen Beschriftungen in Form von Alt-Text oder Captions sind essentiell für die Nutzung von Screenreader Software und dienen ebenso als unterstützende Informationsquelle. Allerdings haben McEwan und Weerts schon 2007 festgestellt, dass ein Großteil aller Webseiten trotz diverser Standards keinerlei oder nur unzureichende Beschreibungen und Metadaten im Web zu Bildern liefern.

Aber auch heute noch ist das Problem vorhanden, dass oft kontextuelle Informationen zu Bildern fehlen. So ist der derzeitige Stand auf Wikipedia, dass ein Großteil aller Bilder zu Artikeln mit keinem beschreibenden Kontext verknüpft sind (Wikimedia Foundation, 2021). Die derzeitige Lösung von Wikipedia ist es, direkte manuelle Übersetzungen zu übernehmen oder Page Interlinks zu verwenden. Allerdings kann dadurch nur ein geringer Teil aller Bildbeschreibungen abgedeckt werden. Um dieses Problem zu lösen und somit auch die Barrierefreiheit und die Suche nach Informationen zu verbessern, wurde deshalb von der Wikimedia Foundation die Wikipedia – Image/Caption Competition auf Kaggle (2022) vorgestellt (Wikimedia Foundation, 2021). Das Ziel dieser Competition ist es, ein Machine Learning Modell zu entwickeln, das einen beschreibenden, multilingualen Text und ein entsprechendes Bild zuweisen kann. Im Speziellen soll das Modell so trainiert werden, dass es übergebene Bilder mit multilingualen Artikeln oder auch mit komplexen multilingualen Captions assoziiert.

Für die Bearbeitung des Wettbewerbs ist es also notwendig, sowohl Natural Language Processing (NLP) Aspekte in Bezug auf die Beschreibungen als auch Bereiche aus der Computer Vision (CV) Forschung für die entsprechenden Bilder zu verbinden. Das vom Forschungs- und Entwicklungsunternehmen OpenAI 2021 erstellte Modell des Contrastive Language-Image Pre-Training (CLIP) ist ein multimodales Modell, dass die beiden Themenfelder verbindet (Radford et al., 2021).

Der Aufbau des Modells unterteilt sich in einen Text Encoder und in einen Image Encoder, die ihren jeweiligen Text- und Bildinput in Vektoren umwandeln, aus denen eine Softmax-Matrix erstellt wird. Hier wird während des Pre-Trainings durch Contrastive Loss dem Modell „beigebracht“, welche Bilder und Texte sich ähneln. Der extra für das Modell zusammengestellte Datensatz umfasst 400 Millionen Bild- und Textpaare (Radford et al., 2021).

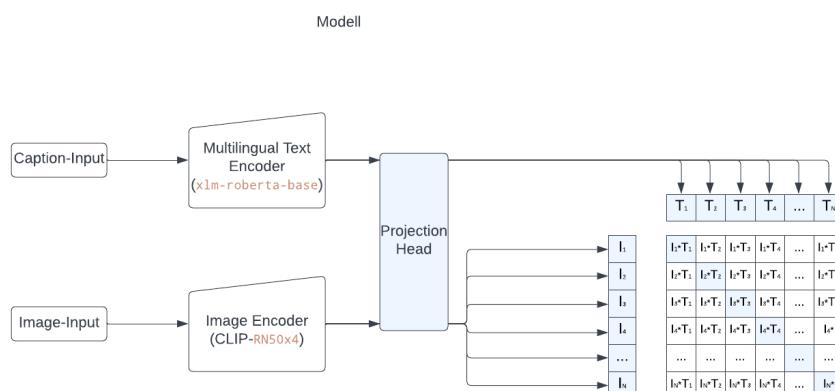
Durch das Pre-Training auf diesen Daten ist es möglich, dass das Modell bei vielen Aufgaben, die die Bereiche NLP und CV verbinden, per Zero Shot eine sehr gute Performanz erreicht (Radford et al., 2021). Allerdings ist das Modell auf die Sprache beschränkt, in der es trainiert wurde: “However, CLIP and the majority of related work focus exclusively on English. This has created a performance vacuum for other low-resource languages, where there is often a lack of high-quality data” (Carlsson et al., 2022, S. 1). Durch das Pre-Training von CLIP mit Hilfe des monolingualen Datensatzes in englischer Sprache ist die Bearbeitung der multilingualen Competition durch das CLIP-Modell nicht ohne Weiteres möglich. Um den Wettbewerb erfolgreich zu bearbeiten, ist es nötig, das CLIP Modell auf Multilingualität auszurichten.

1.1 Ziel der Arbeit

Das Ziel dieser Arbeit ist es, mit Hilfe von CLIP als Ausgangspunkt ein multilinguales, multimodales Machine Learning Modell zu entwickeln und so zu optimieren, dass die Wikipedia Image/Caption Competition damit erfolgreich bearbeitet werden kann. Es soll gezeigt werden, wie es möglich ist, Teile der Architektur eines Dual Encoder Systems wie CLIP zu nutzen, um damit ein Modell zu erstellen, das mit Hilfe des zur Verfügung gestellten Wikipedia basierten Image Text Datensatzes (WIT) (Srinivasan et al., 2021) trainiert werden kann.

Abbildung 1

Eigenes Modell basierend auf dem Modell von Radford et al. (2021)



Dazu soll, wie in Abbildung 1 zu sehen, der vortrainierte ResNet-50 “RN50x4” Image Encoder des CLIP-Modells bestehen bleiben (Radford et al., 2021), der monolinguale Text Encoder jedoch ausgetauscht werden. Als Ersatz wird eine Variante des auf BERT (Devlin et al., 2019) basierenden Sprachmodells XLM-RoBERTa (XLM-R) (Conneau et al., 2020) verwendet.

Zur Vergleichbarkeit und als Benchmark wird jedoch vor der Erstellung des multilingualen Modells die Performanz der Zuweisung der Bild- und Textdaten durch Zero Shot mit dem originalen CLIP-Modell durchgeführt. Im Anschluss wird mit dem erstellten Modell und zwei Optimierungsstrategien versucht, die aufgestellte Benchmark zu verbessern. Diese Strategien beinhalten die Verwendung eines weiteren multilingualen, multimodalen Modells, das auf der Arbeit von Reimers und Gurevych (2020) basiert und die Untersuchung von Prompt Engineering auf die Modelle. Für einen besseren Überblick der Abfolge wird im Folgenden der Aufbau der Arbeit genauer beschrieben.

1.2 Aufbau der Arbeit

Die Arbeit ist in drei größere Abschnitte unterteilt. Im Anschluss an diese Einleitung folgt eine Darstellung der Grundlagen sowie hintergründiger Theorien, daraufhin wird die Konzeption des Modells und der Optimierungsstrategien dargestellt, um dann im letzten Abschnitt die Implementierung und Evaluation zu erläutern.

Im Abschnitt der Grundlagen wird zunächst auf die Rahmenbedingungen und Ziele des Wettbewerbs genauer eingegangen. Es wird beschrieben, welche Ziele die Wikimedia Foundation mit der Organisation der Competition explizit beabsichtigt zu erreichen. Daraufhin wird der als Basis genutzte WIT Datensatz untersucht. Als wichtigster Ausgangspunkt für das Training des multilingualen Modells wird hier die Zusammensetzung des Datensatzes beschrieben. Für die Erstellung des multilingualen, multimodalen Modells sind viele neue Methoden der NLP-Forschung vonnöten. Deswegen wird der Fortschritt in der NLP-Forschung, der die Konzipierung des Modells möglich macht, dargestellt. Hierzu wird die Geschichte des NLP bis hin zur Entwicklung der Transformer Architektur und den daraus resultierenden neuesten Sprachmodellen der letzten Jahre, die dem Text Encoder als Basis dienen, erläutert. Im Anschluss wird die Konstruktion des CLIP-Modells untersucht und als abschließender Punkt wird auf andere multilingualen, multimodalen Modelle der letzten Jahre Bezug genommen.

Nach der Darstellung dieser Grundlagen kann die Konzipierung der Benchmark durch Zero Shot CLIP und des multilingualen Modells erfolgen. Hierfür wird das in der Industrie für den Prozess der Modellderstellung verwendete Ablaufmodell CRISP-DM herangezogen (Wirth & Hipp, 2000). Ausgehend von dieser Vorlage wird die Planung des Data Understanding, der Data Preparation und der Data Visualization beschrieben, um daraufhin die Konzeption der Benchmark durch Zero Shot CLIP vorzunehmen. Im Anschluss wird die Modellierung des multilingualen Text- und Bildabgleichmodells geplant und das Vorgehen für die zwei Optimierungsstrategien festgelegt. Die Festlegung der Evaluationsmethode wird in Abschnitt 3.7 erläutert. Hier wird die Top k Accuracy als Prüfung der Retrievalqualität hervorgehoben und mit der Evaluationsart des Wettbewerbs verglichen.

Die Beschreibung der Implementation in Abschnitt 4 folgt dem Ablauf der Konzeption und beschreibt die praktische Umsetzung der Arbeit. Es wird auf den Ausgangspunkt des Data Understanding des WIT Datensatzes mit einhergehender Visualisierung eingegangen. Ebenfalls wird die sich daraus ergebende Präparationen der Daten untersucht. Daraufhin wird die Erstellung des multilingualen Modells und dessen Training erläutert. Die Umsetzung der Optimierungsstrategien durch das multilinguale Sentence Transformer Modell (Reimers o.D.) und das Prompt Engineering schließt den Abschnitt der Implementation.

Im letzten Abschnitt werden abschließend die Ergebnisse zusammengefasst, auf die möglichen Limitationen und Bias Bezug genommen und ein Ausblick auf weitere mögliche Vorgehen gegeben. Doch um zunächst die Arbeit besser in den wissenschaftlichen Kontext einordnen zu können, wird im Folgenden auf den derzeitigen Stand der Forschung eingegangen.

1.3 Stand der Forschung

In den letzten Jahren hat sich die Forschung rund um Sprachmodelle im NLP mit Hilfe von großen, vortrainierten Transformer Modellen wie BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) und der multilingualen Version XLM-RoBERTa (Conneau et al., 2020) stark beschleunigt (Min et al., 2021). Diese Modelle wurden auf viele Arten erweitert. Zum Beispiel durch die Möglichkeit, das BERT Modell multimodal zu machen und die Bearbeitung von Bildern zu ermöglichen. Modelle wie ViLBERT (Lu et al., 2019) und Pixel-BERT (Huang et al., 2020) haben sich aus solchen Erweiterungen ergeben. Weitere Modelle wie ImageBERT (Qi et al., 2020) und VL-BERT (Su et al., 2020) folgten. Eine weitere Art der Ergänzung der Sprachmodelle ist die Inklusion von Multilingualität.

So wurde das Modell mBERT (Devlin et al., 2019) auf 104 der verbreitetsten Sprachen aus Wikipedia vorgenommen. Allerdings schneidet XLM-RoBERTa, das auf dem Common Crawls Datensatz (Wenzek et al., 2019) trainiert wurde, bei der cross-lingualen Klassifikation von unterrepräsentierten Sprachen um 23% besser ab als das mBERT Modell (Conneau et al., 2020).

Neue Sprachmodelle wie T5 (Raffel et al., 2020) und BART (Lewis et al., 2019) haben sich mittlerweile in die Reihe der sich schnell übertreffenden Modelle der letzten Jahre eingereiht und wurden ebenfalls mit z.B. mT5 (Xue et al., 2021) auf Multilingualität ausgeweitet. Möglich gemacht wurde die umfangreiche Ausbreitung der NLP-Forschung unter anderem durch die immer größer werdenden Datensätze der letzten Jahre.

GPT (Radford & Narasimhan, 2018) sowie dessen Nachfolger GPT-2 (Radford et al., 2019) und GPT-3 (Brown et al., 2020) veranschaulichen dies durch ihre großen Pre-Training Datensätze sehr gut. Ähnlich wie bei den NLP-Modellen verhält es sich auch bei den CV-Modellen. Moderne Datensätze wie beispielsweise ImageNet-21k (Ridnik et al., 2021) wachsen stetig an.

Auch im kombinierten Bereich der multimodalen Modelle hat die Entwicklung von umfangreichen Datensätzen mit CC12M (Changpinyo et al., 2021) und auch YFCC100M (Thomee et al., 2016) zugenommen. Der von Radford und Kollegen (2021) erstellte Datensatz, auf dem CLIP vorgenommen wurde, setzt diese Reihe an umfangreichen Pre-Training Datensätzen fort und ist mit mehr als 400 Millionen Datenpaaren einer der größten der letzten Jahre. Wegbereiter waren kleinere Datensätze, die sich auf visuell-sprachliche Daten spezialisierten. Flickr30k (Young et al., 2014) und MS-COCO (Lin et al., 2014) gehörten zu den ersten Datensätzen, die es ermöglichten, solche Pre-Training Modelle zu erstellen.

Ebenfalls im Bereich der multilingualen Datensätze gibt es den Trend des Wachstums. Multi30k-DE (Elliott, et al., 2016) und DeCOCO (Hitschler et al., 2016) erweiterten die englischsprachigen Datensätze um deutsche Anteile. Ebenso vergrößerte Multi30K-FR (Elliott et al., 2017) das Feld um französische, Multi30K-CS (Barrault, et al., 2018) um tschechische, YJCaptions26k (Miyazaki & Shimizu, 2016) um japanische und MS-COCO-CN (Li et al., 2019) auf chinesische Sprachanteile. Mit Hilfe dieser Datensätze ist es möglich geworden, NLP-Modelle in großem Umfang vorzutrainieren. Einhergehend mit den immer größer werdenden multimodalen Datensätzen werden auch mehr multimodale Modelle entworfen.

Vor allem transformerbasierte Herangehensweisen werden für den Bild-Text-Abgleich verwendet. So haben Messina und Kollegen (2021) ein Transformer-Netzwerk entwickelt für die Aufgabe des multimodalen Information Retrieval in großem Maßstab. Ebenso gehören CLIP genauso wie ALIGN (Jia et al., 2021) und auch das Modell UNIMO (Li et al., 2020) zu diesen. Weitere Modelle, die sich auf die multimodale Bild- und Text-Fokussierung in den letzten Jahren spezialisiert haben, sind UNITER (Chen et al., 2020), M6 (Lin et al., 2021) und auch DALL-E (Ramesh et al., 2021).

Alle diese Modelle haben gemein, dass sie sich auf den englischen Sprachgebrauch beschränken. Die Entwicklung von multilingualen, multimodalen Modellen ist auf Grund der geringeren Datenlage überschaubarer. Modelle wie das von Jain und Kollegen (2021) erstellte MURAL zeigen jedoch, dass auch in dieser Forschungsrichtung Fortschritte erzielt werden.

Ein weiteres multimodales Modell, das in dieser Arbeit im Abschnitt der Optimierungsstrategien noch genauer untersucht wird, ist das von Reimers (o.D.) erstellte clip-ViT-B-32-multilingual-v1 Sentence Transformer Modell, welches durch eine Teacher-Student-Architektur erstellt wurde (Reimers & Gurevych, 2020).

Damit schließt der Überblick über den sich schnell entwickelnden Forschungsbereich des NLP über multimodale Modelle und deren wachsenden Datensätze. Da nun der wissenschaftliche Kontext, in dem sich die Arbeit durch die Erstellung des multilingualen Modells bewegt, abgesteckt ist, folgt im nächsten Abschnitt die Erläuterung der Grundlagen.

2. Grundlagen

In diesem Abschnitt steht die Darstellung der grundlegenden Methoden im Fokus, die das Fundament bilden, auf dem CLIP und das in dieser Arbeit erstellte Modell fußt. Dies ist wichtig, um später die Konzeption und Implementation des Modells nachvollziehen zu können. Als Einstieg in die Grundlagen wird im Folgenden jedoch zunächst auf die Intention, Struktur und Inhalte des Wettbewerbs eingegangen, um daraufhin den WIT Datensatz genauer in Augenschein zu nehmen. Daraufhin werden die grundlegenden Eckpunkte der NLP-Forschung wiedergegeben, die die Erstellung eines multimodalen, multilingualen Modells möglich machen.

Vor allem wird auf die Erstellung der Transformer Architektur durch Vaswani und Kollegen (2017) eingegangen, die viele Neuerungen im Deep Learning Bereich nach sich zog. Aufbauend darauf werden die Sprachmodelle BERT, RoBERTa und schließlich das später als Text Encoder verwendete XLM-RoBERTa (XLM-R) Modell untersucht. Im Anschluss wird der Aufbau des CLIP Modells, dessen Pre-Training und die Struktur von anderen multilingualen, multimodalen Modellen dargestellt.

Wie schon in der Einleitung beschrieben ist das übergeordnete Ziel des Wettbewerbs, die Suche nach Informationen zu verbessern und die Barrierefreiheit zu erhöhen. Das Problem, dass ein Großteil aller Bilder zu Artikeln keinen beschreibenden Kontext hat, veranlasste das Forschungsteam der Wikimedia Foundation dazu, nach Lösungen zu suchen und die Wikipedia – Image/Caption Competition zu organisieren.

Im Wettbewerbsaufruf formulierten sie die Intention der Competition wie folgt: “The majority of images on Wikipedia articles, for example, don't have any written context connected to the image. Open models could help anyone improve accessibility and learning for all” (Wikimedia Foundation, 2021, Abs. 1).

Des Weiteren unterstreichen sie, dass der derzeitige Status von Wikipedia nur unzureichend Abhilfe schafft. Derzeit würden nur einfache Methoden basierend auf Übersetzungen und Seitenverknüpfungen verwendet werden und diese hätten dazu geführt, dass ein Großteil der Bilder ohne semantischen Kontext bleibt (Wikimedia Foundation, 2021). Deswegen ist das Ziel des Wettbewerbs, über diese Methoden hinaus zu gehen: “In this competition, you'll build a model that automatically retrieves the text closest to an image. Specifically, you'll train your model to associate given images with article titles or complex captions, in multiple languages” (Wikimedia Foundation, 2021, Abs. 3). Im Genauen ist das Ziel der Competition, ein Target zu bestimmen, das durch gegebene Informationen über ein Bild identifiziert werden soll. Diese Targets sind in mehreren Sprachen im schon erwähnten WIT Datensatz vorhanden. Die Modelle werden auf Kaggle durch Normalized Discounted Cumulative Gain (NDCG) evaluiert. Im Fall des Wettbewerbs bemisst der NDCG die Ranking-Qualität der Ausgaben an Hand der top fünf zurückgegebenen Captions je Bild, die von den Modellen ausgegeben werden und wird im Abschnitt 3.7 noch genauer untersucht und dem in der Arbeit verwendeten Top k Accuracy gegenübergestellt.

Als Ziel des Wettbewerbs und für die Evaluation der Modelle soll eine Datei erstellt werden, aus der hervorgeht, welche Zuweisungen das Modell vorgenommen hat. Diese Submission File soll aus einer Liste aus IDs der Bilder und deren wahrscheinlichsten Beschreibungen in Form der *caption_title_and_reference_description* bestehen.

Diese ID- und Caption-Paare sollen in ihrer Reihenfolge von oben nach unten entsprechend ihrer Relevanz geordnet werden und haben dabei maximal fünf Einträge. Zum Beispiel wäre das Gegenstück zur top ID die am meisten relevanten *caption_title_and_reference_description*. Auch soll die Datei über einen Header verfügen und somit wie folgt aufgebaut sein:

Abbildung 2

Header der beispielhaften sample_submisson.csv Datei

id	caption_title_and_reference_description
0	diam in ve ligula a sapien mattis a a
1	nisi ut eu
2	urna ad a duis a ligula dis ve
3	eros eu a vel congue ac et justo sapien
4	diam mi a mi nec a leo sed ornare magna mus se...
5	eros ad a magnis elit ad a in a
6	amet ut a a
7	pede ad
8	nisi eu ut est sapien felis id felis
9	nisl ac a eu a primis integer nih a

Anmerkung. Zu erkennen ist, dass es für eine Bild-ID bis zu fünf Captioneinträge gibt.

Was die *caption_title_and_reference_description* genau beschreibt, wie es in den Aufbau des Datensatzes passt und aus welchen anderen Bausteinen der Datensatz noch besteht, wird im Anschluss genauer erläutert.

2.1 WIT Datensatz

Die Zusammenstellung des Datensatzes durch das Forschungsteam der Wikipedia Foundation begann mit dem Sammeln der originalen Daten. Die Umsetzung des multilingualen Modells, die Art und Weise des Preprocessing der Daten in den Abschnitten 3 und 4 ergibt sich aus der Struktur des WIT Datensatzes.

Zu Beginn wurde der Inhalt von bis zu 124 Millionen Wikipediaseiten in 279 verschiedene Sprachen zusammengeführt. Um die Wikipedia-Daten zu filtern und zu reinigen, wurden sie durch eine Flume Pipeline bearbeitet und gespeichert (Srinivasan et al., 2021), da das Open

Source System Flume das Sammeln, Aggregieren und Verschieben großer Mengen unstrukturierter Daten aus verschiedenen Datenquellen ermöglicht (Chambers et al., 2010). Aus diesen unstrukturierten Daten wurden die Bilder und deren zugehörige Textdaten entnommen. Daraus ergaben sich ~150 Millionen Tupel an Daten aus Bildern, Beschreibungen und Kontextdaten, die alle wiederum durch verschiedene Filter bearbeitet wurden. Die unterschiedlichen Textinformationen über die vorhandenen Bilder stammen aus einer Vielzahl an Quellen. Die drei Textinformationen, die mit den Bildern am meisten in Verbindung stehen, sind die Referenzbeschreibung, die Zuordnungsbeschreibung und die Beschreibung des Alt-Texts (Srinivasan et al., 2021).

Die Referenzbeschreibung ist die direkte Caption unter einem Bild auf Wikipedia und wird von Srinivasan und Kollegen (2021) als die relevanteste und thematisch genaueste der Beschreibungen betitelt. Allerdings sind diese Beschreibungen mit ungefähr 25 Millionen die am wenigsten verbreitete Beschreibungsart im Datensatz.

Die meisten Textinformationen zu einem Bild werden durch die Zuordnungsbeschreibung oder auch Attribution-Beschreibung gegeben. Dies sind die Textbeschreibungen der Bilder, die auf der Wikimediawebseite der Bilder gefunden werden können. Viele dieser Textinformationen sind multilingual mit denselben Beschreibungen für das gleiche Bild in unterschiedlichen Sprachen. In 138 Millionen aller Tupeldaten ist dieses Feld vorhanden, allerdings ist ein Großteil dieser Beschreibungen noisy und somit nicht informativ.

Die letzte gesammelte deskriptive Informationsart der Bilder ist der Alternative Text. Genutzt wird dieser auf der Seite nicht sichtbare Text zu den Bildern von Screen Readern zur Verbesserung der Barrierefreiheit. Allerdings fanden Srinivasan und Kollegen (2021) heraus, dass diese Textart die am wenigsten informativste Art der Beschreibungen war, obwohl sie genau das Gegenteil erreichen sollte: "Despite this (surprisingly) we discovered that this was the least informative of the three texts and in most cases was simply the image file name (We found that of the 121M+ tuples containing this text, only a small fraction to be meaningful descriptions of the image)" (S. 3). Neben den drei Beschreibungsarten wurden vom Wikimediateam innerhalb des Kontextbereichs der Tupel zusätzliche Texte identifiziert, die indirekt mit den Bildern in Verbindung stehen. Dazu gehören zum Beispiel der *section_text* oder auch der *page_title*.

Um die Datentupel zu filtern, wurden mehrere Schritte angesetzt. Zuerst wurde festgelegt, dass nur Beschreibungen berücksichtigt werden sollten, die länger waren als drei Zeichen. Als zweiten Schritt implementierte das Team eine Lösung, die alle generischen Alt-text Phrasen wie ".jpg", "icon", "refer to" usw. entfernte.

Für die letzten Schritte der Filterung wurden alle Bilder in die Formate JPG oder PNG überführt, um so zu garantieren, dass möglichst viele der Bildinformationen bestehen bleiben. Ebenso wurden Bilder im GIF Format beibehalten, wenn sie eine Referenzbeschreibung besaßen. Weiterhin wurde für Tupel ohne Referenzbeschreibung festgelegt, dass das dazugehörige Bild nicht im letzten Abschnitt der Seite auftaucht (Srinivasan et al., 2021). Zur Filterung der Bilder wurden folgende Maßstäbe gesetzt. Zum Einen wurde die Höhe und Breite der Bilder auf ein Minimum von 100 Pixel angesetzt, um so zu ermöglichen, dass die Bilder ein Mindestmaß an Aussagekraft besitzen. Ebenso wurden Bilder entfernt, die generische Merkmale oder nicht sinnvolle Textverknüpfungen aufwiesen. Dazu gehören zum Beispiel Bilder von Landkarten, die generisch auf Wikipedia verwendet werden, jedoch nicht spezifisch für einen genauen geographischen Ort eingefügt werden. Daraus ergaben sich inkorrekte Bild-Text Assoziationen in den Daten, die deshalb entfernt wurden. Weitere dieser generischen Daten waren Icons, Flaggen, Logos oder auch Platzhalterbilder.

Um Bias zu vermeiden, wurden alle Bilder dieser Art stark unterbewertet. Ein weiteres Filtermerkmal war das Lizenzattribut der Bilder. Laut Srinivasan und Kollegen (2021) kamen nur Bilder in Betracht, die unter der Creative Commons Lizenz eingestellt wurden. Auch wurde die Anzahl der Sprachen auf 100 begrenzt. Es wurden somit nur Sprachen berücksichtigt, die mindestens zwölftausend oder mehr Tupel besitzen. Das Ergebnis dieser Filterungen war ein Datensatz, der bis dato mit 37 Millionen (Bild, Text, Kontext) Tupeln der größte multilinguale Datensatz war. Er umfasst 108 Sprachen und beinhaltet 11,5 Millionen einzigartige Bilder. Viele der Bilder besitzen mehrere unterschiedliche Textbeschreibungen und die Hälfte aller Tupel hat mindestens zwei Referenzbeschreibungen, Zuordnungsbeschreibung und Alt-text-Beschreibungen. Ebenso besitzen die Hälfte der 108 Sprachen mehr als hunderttausend einzigartige Bild-Text-Tupel und hunderttausend einzigartige Bilder. Durch diese Zusammensetzung ihres Datensatzes gehen Srinivasan und Kollegen (2021) davon aus, dass es möglich ist, die multilinguale Performanz von multimodalen Modellen zu verbessern.

2.1.1 Datensatz des Wettbewerbs

Der Datensatz des Wettbewerbs und somit die Basis für die Erstellung des multilingualen Modells setzt sich aus den zusammengestellten Daten des Forscherteams von Wikimedia zusammen. Ebenso wie im WIT Datensatz befinden sich im Wettbewerb-Datensatz 108 unterschiedliche Sprachen und es befinden sich 37.6 Millionen Bild-Text Paaren mit 11.5

Millionen einzigartigen Bildern darin. Es gibt für jede Sprache mindestens 12.000 Beispiele. Der Datensatz wurde schon im Vorhinein in ein Test- und ein Trainingsset aufgeteilt, die jeweils aus 92.367 Eintragungen bestehen. Jeder Datenpunkt im Trainingsset enthält eine Bild-URL, von der das Bild abgefragt werden kann und mehrere Textinformationen, darunter auch die Target-Beschreibung. Im Gegensatz dazu besteht das Testset aus zwei unterschiedlichen Listen aus Bild-URL Eintragungen und den durch das Modell zu zuweisenden Beschreibungen. Zusätzlich zu diesen Sets werden die Bilddaten noch einmal gesondert als codierte base64 Versionen, als *resnet_embeddings* oder auch zum Herunterladen bereitgestellt. Für eine bessere Übersicht wird nachfolgend ein Überblick über den Datensatz in Stichpunkten gegeben.

- **train-{0000x}-of-00005.tsv** - Die Trainingsdaten sind in sechs .tsv Dateien aufgeteilt. Die Struktur des Datensatzes wird in Abbildung 3 dargestellt. Neben den dort gezeigten Attributen gibt es noch die *caption_reference_description*, *caption_attribution_description*, *caption_alt_text_description*, *mime_type*, *original_height*, *original_width*, *is_main_image*, *attribution_passes_lang_id* und die *page_changed_recently*. Die für das Training wichtigsten Spalten sind die *image_url* und die *caption_title_and_reference_description*.

Abbildung 3

Beispielhafte, verkürzte Darstellung des Headers der Trainingsdaten

	language	page_url	image_url	page_title	...	context_page_description	context_section_description	caption_title_and_reference_description
0	vi	https://vi.wikipedia.org/wiki/Gi%C3%A0y_cao_g%C3%AD%	https://upload.wikimedia.org/wikipedia/commons...	Giày cao gót	...	Giày cao gót là một loại giày trong đó gót châ...	Giày cao gót có một lịch sử lâu đời, có niên đ...	Giày cao gót [SEP] Giày cao gót châu Âu, khoán...
1	fr	https://fr.wikipedia.org/wiki/Grand_Prix_automobile_de_France_1957	https://upload.wikimedia.org/wikipedia/commons...	Grand Prix automobile de France 1957	...	Le Grand Prix de France 1957, disputé sur le c...	La saison 1957 est la quatrième se disputant s...	Grand Prix automobile de France 1957 [SEP] Ju...
2	zh-TW	https://zh.wikipedia.org/zh-tw/%E9%81%93%E8%80...	https://upload.wikimedia.org/wikipedia/commons...	道谷站	...	道谷站是一個位於首爾特別市江南區道谷洞，屬於首爾地鐵3號線與盆唐線的轉乘站。	2面2線對向式月台，設有月台幕門。本線軌道中間分隔得很開，看起來就像設有留置線，實際上是大時...	道谷站 [SEP] 車站月台
3	zh-TW	https://zh.wikipedia.org/wiki/%E5%8C%97%E4%BB%...	https://upload.wikimedia.org/wikipedia/commons...	北仙台站	...	北仙台站是一個位於日本宮城縣仙台市青葉區昭和町，屬於仙山線、仙台市地下鐵南北線的鐵路車站。仙...	島式月台1面2線的地面車站，2號月台與本線是1線貫通構造。月台、大堂位於1樓是地面車站，軌道...	北仙台站 [SEP] 月台
4	en	https://en.wikipedia.org/wiki/Silver_spoon	https://upload.wikimedia.org/wikipedia/commons...	Silver spoon	...	"The English language expression silver spoon ...	Before the place setting became popular around...	Silver spoon [SEP] Two silver-gilt strainer sp...

Anmerkung. Für die bessere Übersichtlichkeit wurden die Spalten *caption_reference_description*, *caption_attribution_description*, *caption_alt_text_description*, *mime_type*, *original_height*, *original_width*, *is_main_image*, *attribution_passes_lang_id* und *page_changed_recently* nicht aufgeführt.

- **test.tsv** - Bei den Testdaten ist das Ziel die target *caption_title_and_reference_description* für jede ID-Zeile zu bestimmen.
- **test_captions_list.csv** - Diese csv Datei enthält eine Liste an Captions, die für Testzwecke retrieved werden können.
- **sample_submission.csv** - Die schon im vorherigen Abschnitt beschriebene Submission File gibt das korrekte Format der Lösung vor. Hier ist zu sehen, dass es möglich ist, bis zu fünf Vorhersagen der *caption_title_and_reference_description* je ID aus der test_captions_list.csv zu geben.
- **image_data_test/** - Der image_data_test Ordner enthält zusätzlich die Bilder des Datensatzes in unterschiedlichen Formaten.
 - **image_pixels/test_image_pixels_part-{0000x}.csv.gz**
 - image_url: Die URL zum originalen Bild ähnlich der Verwendung in den train-{0000x}-of-00005.tsv Dateien.
 - b64_bytes: byte64 codierte Bytes der Bilder in 300px Auflösung.
 - metadata_url: URL, die zur “commons” Seite der Bilder zeigt.
 - **resnet_embeddings/test_resnet_embeddings_part-{0000x}.csv.gz**
 - image_url: Auch hier, ähnlich der Verwendung in den train-{0000x}-of-00005.tsv wird per URL zum originalen Bild verwiesen.
 - embedding: Die Embeddings der Bilder durch eine mit Kommata getrennte Liste von 2048 Float Werten.

Im Gegensatz zu **image_data_test** werden die Daten in **image_data_train** gesondert auf einem Server gehostet, da sie mit ihrer Größe von ~275 GB den Umfang sprengen würden.

2.2 Natural Language Processing

Nachdem nun die Zusammensetzung des Datensatzes erläutert wurde, können die Grundlagen der hinter dem Modell stehenden fundamentalen Theorien untersucht werden. Dieser Abschnitt dient zur besseren Veranschaulichung, wie durch die Nutzung von NLP ein multimodales Modell auf Basis der CLIP Architektur erstellt werden kann, dass es möglich macht, Bild- und Textabgleiche herzustellen. Wie in der Einleitung beschrieben, hat der Aufbau des CLIP Modells zwei grundlegende Ansatzpunkte. Beide Encoder fußen auf den umfangreichen Erkenntnissen der jeweiligen Forschungsfelder, die in den letzten Jahren rasante Fortschritte gemacht haben.

Deswegen folgt ein kurzer Überblick über die Entwicklung der NLP-Forschung, um das Thema dieser Arbeit besser in die Beziehung der Forschungsfelder einordnen zu können. Torfi und Kollegen (2021) definieren NLP als “[...] a sub-discipline of computer science providing a bridge between natural languages and computers. It helps empower machines to understand, process, and analyze human language” (S. 1).

NLP ist somit ein Forschungs- und Anwendungsbereich, in dem durch Algorithmen natürliche Sprache aus Text- oder Audioformaten verstanden und bearbeitet werden. Die Ursprünge von NLP liegen in den 1940er Jahren und begannen mit ersten Überschneidungen aus künstlicher Intelligenz und Linguistik (Khyani & Siddhartha, 2020). Hier lag das Augenmerk besonders auf maschineller Übersetzung. Weitere Entwicklungen im NLP wurden vor allem durch Noam Chomsky berühmt, der durch die theoretische Analyse von Grammatiken die Erstellung der Backus-Naur-Form beeinflusste (Nadkarni et al., 2011). Ein weiterer Meilenstein war das am MIT konzipierte NLP-Programm ELIZA in den 1960er Jahren. Dieses Dialogprogramm simulierte durch Mustervergleiche und Substitutionsmethoden einen Dialog mit einem Benutzer (Khyani & Siddhartha, 2020).

All diese Errungenschaften machten es möglich, dass zwischen Ende der 1980er und Anfang der 1990er Jahre ein Paradigmenwechsel in der NLP-Forschung stattfand. In diesen Jahren wurde die Verwendung von statistischen NLP-Methoden zum neuen Standard (Nadkarni et al., 2011). Klein (2005) fasste vier Anhaltspunkte dieses Wechsels zusammen. Statt ausführlicher, tiefer Analyse wurden nun einfache Approximationen verwendet. Auch wurde die Evaluation immer wichtiger und umfangreicher. Einer der ausschlaggebendsten Punkte war, dass Machine Learning Methoden prominent und große Textkörper zum Trainieren von Algorithmen für maschinelles Lernen verwendet wurden. Torfi und Kollegen (2020) fasst diese Entwicklung wie folgt zusammen: “These developments led to a paradigm shift from traditional to novel data-driven approaches aimed at advancing NLP” (S. 1). Auch wurden während dieser Zeit die ersten Recurrent Neural Networks (RNN) erstellt und trainiert, die ab der Jahrtausendwende immer häufiger benutzt wurden. Mit Google Translate wurde 2006 das erste große kommerziell erfolgreiche statistische NLP-System in Betrieb genommen (Turovsky, 2016). Im letzten Jahrzehnt hat sich die Entwicklung innerhalb des NLP rasant beschleunigt. Anfang der 2010er wurde durch Word2Vec die Effizienz von Embeddings drastisch erhöht (Mikolov et al., 2013). Die am häufigsten verwendeten neuronalen Netze innerhalb des NLP in dieser Zeit waren Long Short-term Memory (LSTM) Netzwerke, RNNs und Convolutional Neural Networks (CNN). Diese Entwicklungen haben sich in den letzten Jahren noch einmal ausgedehnt und wurden durch die Verwendung von Deep Learning immer leistungsfähiger (Torfi et al., 2020).

Auch wurden durch neue Strukturen wie die Encoder-Decoder-Architektur neue Herangehensweisen erschlossen. Insbesondere durch die Erfindung der Transformer-Architektur 2017 (Vaswani et al., 2017) gab es einen stetigen Anstieg an NLP-Modellen und es wurden neue Standards im NLP erreicht (Sharma et al., 2022).

Als Basismethoden innerhalb der NLP-Forschung, die vielen Bearbeitung vor allem durch Machine Learning zugrunde liegen, kann man Tokenisierung, Stemming und Lemmatisierung ansehen (Khyan et al., 2020).

Tokenisierung

Bei der Tokenisierung wird der Input Satz in einzelne Teilabschnitte unterteilt. Die gebräuchlichste Methode zur Bildung von Token ist die Verwendung von Leerzeichen. Der Satz wird somit auf die einzelnen Wörter aufgeteilt. Ebenso können Token jedoch als Zeichen oder Teilwörter eines Satzes festgelegt werden (Manning, 2008).

Stemming

Khyani und Kollegen (2020) beschreiben das Stemming als einen wichtigen Teil des Pipelining Prozesses innerhalb des NLP. Aber auch in Bereichen des Information Retrieval ist Stemming eine Technik, die oft benutzt wird (Balakrishnan & Lloyd-Yemoh, 2014). Der Vorgang des Stemming beschreibt das Entfernen von Ableitungssuffixen und Beugungen, so dass der Stamm oder die Wurzel eines Wortes zurückbleibt. Durch diesen Stamm kann die Verbindung von einander ähnlichen Wörtern veranschaulicht werden. Der Input des Stemmers ist meist eine Liste von tokenisierten Wörtern. Es gibt verschiedene Stemming-Algorithmen, die entwickelt wurden, um sicherzustellen, dass Wörter auf ihre Stammformen reduziert werden. Der Porter-Stemmer ist zum Beispiel einer der am häufigsten genutzten Stemmer im Information Retrieval (Balakrishnan and Lloyd-Yemoh, 2014).

Lemmatisierung

Die Ziele, beim Prozess des Stemming und der Lemmatisierung, die Stammform eines Wortes zu erhalten, sind dieselben. Bei der Lemmatisierung werden die unterschiedlichen flektierten Arten eines Wortes gruppiert, so dass sie als einziges Element erkannt werden können, das als Lemma oder Grundform des Wortes bezeichnet wird. Hierin ähnelt die Lemmatisierung dem Stemming, zusätzlich wird ein Wort jedoch in den kontextuellen Rahmen eingebettet. Es werden Wörter mit ähnlicher Bedeutung zu einem Wort in Verbindung gebracht.

Die Unterschiede zur Stemming-Methode liegen darin, dass bei der Stemming-Methode die Präfixe und Suffixe der Worte betrachtet werden und bei der Lemmatisierung die Fokussierung auf die kontextuelle Bedeutung der Worte deutlich wird (Khyan et al., 2020).

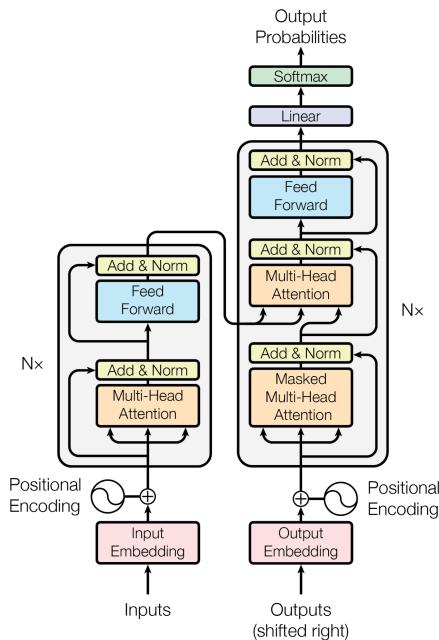
Die beschriebenen Fortschritte der NLP-Forschung und auch die Nutzung von Basismethoden, wie der Tokenisierung, dem Stemming und Lemmatization, machen es möglich, dass es in den letzten Jahren eine enorme Fülle an neuen Errungenschaften in der NLP-Forschung gibt. So wurde zum Beispiel die Transformer-Architektur 2017 entworfen, die im nächsten Abschnitt genauer betrachtet werden soll. Diese Architektur stellt die Basis für die Encoder des CLIP-Modells dar und somit auch für die des zu entwickelnden multilingualen Modells. Nachdem in diesem Abschnitt ein kurzer geschichtlicher und theoretischer Werdegang der NLP-Forschung dargelegt wurde, kann im Folgenden auf diese Architektur der Transformer genauer eingegangen werden, um daraufhin den Text Encoder des multilingualen Modells vorzustellen und diesen besser nachvollziehen zu können.

2.2.1 Transformer

Die wichtigsten Aspekte der Transformer-Entwicklung, die Vaswani und Kollegen (2017) in ihrer Arbeit verfolgten, sind Attention, Positional Encoding und Self-Attention. Das Ziel war es, ein Modell zu entwerfen, dass Sequenz-Transduktion, also die Umwandlung einer Symbolfolge in eine andere verbessert (Vaswani et al., 2017). Dieses Ziel haben sie umgesetzt, indem sie in der neuen Architektur des Transformers, die in Abbildung 4 dargestellt ist, vollständig auf Attention-Mechanismen abzielen, die globale Abhängigkeiten zwischen Input und Output herstellen. Dadurch wurde es mit dem Transformer möglich, erheblich mehr Parallelisierung zu erreichen (Vaswani et al., 2017).

Abbildung 4

Darstellung der Transformer Architektur



Anmerkung. Aus "Attention is all you need" von A. Vaswani, et al., 2017, *Advances in neural information processing systems*, 30, S. 3.

Wie in Abbildung 4 zu sehen ist, ist die neuronale Transformer-Architektur in einen Encoder-Decoder-Stack aufgeteilt. Der Ablauf des Arbeitsprozesses des Transformers beginnt mit dem Input in den Encoder. Durch die vorgelagerte Embedding-Schicht wird der Input in eine Vektorendarstellung umgewandelt. Im Anschluss wird diese Sequenz aus Symboldarstellungen vom Encoder auf eine interne Repräsentation dargestellt. Danach wird vom Decoder eine Ausgangssequenz aus einzelnen Symbolen erstellt. Während jedes einzelnen Bearbeitungspunktes ist das Modell automatisch regressiv und verwendet die zuvor generierten Symbole als zusätzlichen Input für die Generierung des nächsten. Sowohl für den Encoder, als auch für den Decoder gilt, dass sie Self-Attention und vollständig verbundene Schichten untereinander verwenden (Vaswani et al., 2017).

Encoder

Der Encoder besteht dabei aus $N = 6$ identischen Schichten, die wiederum jeweils aus zwei Unterschichten bestehen. Die erste Unterschicht verwendet einen Multi-Head Self-Attention Mechanismus, und die zweite besteht aus einem vollständig verbundenen Feed Forward Netzwerk.

Zwischen den beiden Unterschichten besteht eine Verbindung, die durch Layer Normalization (Ba et al., 2016) abgeschlossen wird. Der Output jeder Unterschicht kann mit der Formel $SchichtNorm(x + Unterschicht(x))$ beschrieben werden.

Hierbei steht $SchichtNorm$ für den normalisierten Output und $Unterschicht(x)$ für die Funktion, die von der Unterschicht wiederum selbst angewandt wird (Vaswani et al., 2017). Alle Unterschichten im Modell und auch alle Embedding-Schichten geben ihren Output in der Dimension 512 aus. Ähnlich dem XLM-RoBERTa (XLM-R) Text Encoder, der in dieser Arbeit benutzt wird und im Anschluss in Abschnitt 2.2.2 beschrieben wird.

Decoder

Genau wie der Encoder besteht der Decoder aus $N = 6$ identischen Schichten. Wie in der Abbildung zu sehen, gibt es zusätzlich zu den zwei Unterschichten wie bei dem Encoder noch eine weitere Unterschicht. Diese wendet ebenso Multi-Head Attention auf den Output des Encoderabschnitts und Layer Normalization auf die Verknüpfungen der Unterschichten an. Zusätzlich wird durch Masking und durch die Verschiebung der Output Embeddings um eine Position abgesichert, dass die Voraussage der Position i nur abhängig sein kann von den bekannten Outputs, die kleiner als i sind. Dies geschieht durch die Modifizierung der Self-Attention der Unterschicht. Alleine die Entwicklung dieses Decoder Stacks ist schon sehr nützlich. So hat OpenAI auf Basis dieses Decoder Stacks die GPT Modelle entwickelt (Radford & Narasimhan, 2018). In ihrer Arbeit haben sie ein 12 Schichtiges Transformer Model mit Hilfe dieses Decoder Stacks entworfen.

Attention

Der Attention-Mechanismus kann als Funktion beschrieben werden, die einen Abfragevektor (Query), einen Satz an Schlüssel und Wertpaaren (Key, Value) als Vektoren und das Embedding aufeinander abbildet. Während des Trainings des Modells werden per Matrixmultiplikation aus den Embeddings die anderen drei Vektoren berechnet. Ziel dieses Mechanismus ist es, die Korrelation eines Embeddings zu anderen Input-Sequenzen zu berechnen und somit zum Beispiel die grammatische Beziehung von Wörtern zueinander in einem Satz zu speichern. Schon vor der Erstellung des Transformer Modells durch Vaswani und Kollegen (2017) war der Attention-Mechanismus im Sequenzmodeling ein integraler Bestandteil. Allerdings war dieser Mechanismus auf die Verwendung in RNN beschränkt.

Self-attention

Der Self-Attention oder auch Intra-Attention genannte Mechanismus unterscheidet sich vom Attention-Mechanismus in der Hinsicht, dass die Inputwerte andere sind. Bei Self-Attention werden ausschließlich die Inputwerte des vorgelagerten Encoders oder Decoders verwendet.

Hierbei werden verschiedene Positionen einer einzelnen Sequenz miteinander in Beziehung gesetzt, um eine Repräsentation der Sequenz zu berechnen. Diese Vorgehensweise wurde schon vor der Erstellung des Transformers erfolgreich in Aufgaben wie der abstrakten Zusammenfassung (Paulus et al., 2017) oder dem Lernen aufgabenunabhängiger Satzrepräsentationen (Lin et al., 2017) verwendet. Allerdings war das Transformer Modell das erste, dass dabei auf RNN oder CNN Herangehensweisen verzichtete: “[...] the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution” (Vaswani et al., 2017, S. 2). Der Self-Attention Mechanismus ist optimal dafür, Abhängigkeiten von Token in einer Sequenz darzustellen und begünstigt, wie schon vorher erwähnt, die parallele Verarbeitung und ist somit gut geeignet für lange Sequenzeingaben.

Attention, Matrixmultiplikation und Multi-Head Attention

Wie im Abschnitt Attention beschrieben, wird die genaue Umsetzung des Attention-Mechanismus im Transformer Modell durch das Skalarprodukt der Eingaben erreicht. Ein Satz an Abfragevektoren, Schlüsselvektoren und Wertvektoren wird in Matrizen Q, K und V mit der Dimension dk zusammengefasst. Aus diesen wird dann per Softmax-Funktion die Gewichtung des Outputs generiert.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

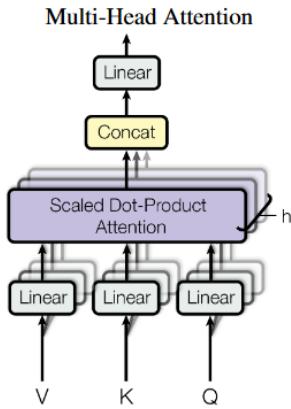
Multi-Head Attention bedeutet in diesem Zusammenhang, dass jeder Head eine Matrix widerspiegelt und dem Modell erlaubt, Informationen von verschiedenen Positionen gleichzeitig zu erreichen.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \\ \text{where } \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned}$$

Wie in der Funktion und in der Abbildung X zu sehen, wird die Multi-Head Attention hier durch die Multi-Head Matrizen Q , K und V als auch die Parametermatrizen W^o und die Zusammenführung der Heads errechnet. Für den Transformer wurden acht solcher Head Schichten mit der Dimension 64 aneinander gereiht.

Abbildung 5

Darstellung der Multi-Head Attention



Anmerkung. Aus “Attention is all you need” von A. Vaswani, et al., 2017, *Advances in neural information processing systems*, 30, S. 4.

Positional Encodings

Da der Transformer keinerlei Wiederholungen wie in RNNs und CNNs besitzt, musste eine andere Art der Informationsgewinnung zu den vorherigen Sequenzen erfolgen. Dies wurde gelöst, indem man Positions kodierungen der Tokens zu den Input-Embeddings am Anfang des Encoder-Decoder Stacks hinzugefügt hat. So ist es möglich, anhand dieser Positions kodierungen festzustellen, welche Position ein Token in der Sequenz einnimmt.

Embeddings

Als Abschluss der Beschreibung der Transformer-Architektur folgt nun noch ein kurzer Abschnitt zu den Embeddings. Die Umwandlung von Input und Output Token zu Vektoren wird im Transformer durch das sogenannte “learned embedding” (Vaswani et al., 2017, S. 5) vorgenommen. Zusätzlich wird eine “learned linear transformation” (Vaswani et al., 2017, S. 5) und Softmax-Funktion benutzt, um die Wahrscheinlichkeit des nächsten Tokens aus dem Decoder Output zu berechnen.

Damit wurden die wichtigsten Aspekte des Transformer-Modells genannt. Wie umfangreich diese Errungenschaft für die Forschung innerhalb des NLP war und wie grundlegend diese für die Erstellung eines multilingualen Modells in dieser Arbeit ist, wird durch die vielen

Sprachmodelle ersichtlich, die in den letzten Jahren mit dieser Technik erstellt worden sind. Die Möglichkeit, mehrere Schichten von Multi-Head Self-Attention zu verwenden, ermöglicht immer effektivere Repräsentationen, die für eine Reihe von NLP-Problemen nützlich sind. Infolgedessen basieren fast alle modernen Sprachmodelle, einschließlich der in Abschnitt 1.3 erwähnten GPT, BERT, BART und T5 auf der Transformer-Architektur (Min et al., 2021).

Auch in vielen übergreifenden Gebieten innerhalb des NLP wurde der Transformer zu einer weit verbreiteten Standardimplementation: “Since the publication of Vaswani et al.’s (2017) paper, a plethora of transformer implementations have been released with various modifications.” (Hommel et al., 2022 S. 754).

2.2.2 Sprachmodelle, Pre-Training und Finetuning

Jurafsky und Martin (2021) beschrieben Sprachmodelle wie folgt: “Models that assign probabilities to sequences of words are called language models or LMs” (S. 30). Viele dieser neuen Sprachmodelle wurden im Abschnitt 1.3 schon erwähnt. Allerdings gab es in den letzten Jahren einen sehr starken Anstieg und die Fülle an neuen Modellen ist schwer zu überblicken. Der Grund dafür ist die schon erwähnte übergreifende Verwendung der neuen Transformer-Architektur. Für eine genauere Beschreibung bezeichnet Manning (2008) Sprachmodelle als eine Funktion, die einen Wahrscheinlichkeitswert einem String zuordnet, der aus einem Vokabular entnommen wurde.

Sprachmodelle oder Sprachmodellierung ist also die Verwendung von probabilistischen Techniken, um die Wahrscheinlichkeit zu berechnen, mit der das Auftreten einer Wortfolge in einem Satz bestimmt werden kann. Dazu werden Textdaten als Grundlage analysiert, um dadurch eine wahrscheinliche Wortvorhersage zu berechnen.

Ab 2018 wurde durch Sprachmodelle wie ELMo (Peters et al., 2018), BERT und den ersten Vertreter der GPT Familie GPT (Radford & Narasimhan, 2018) ebenfalls deutlich, welche Bedeutung Pre-Training und Finetuning nun im NLP-Bereich einnehmen. Denn obwohl Pre-Training schon mindestens ab 2006 (Bengio et al., 2006) untersucht wurde, hat es erst mit diesen großen Sprachmodellen an mehr Bedeutung gewonnen: “[...], the technique did not gain traction in NLP until later in the decade, with the publication of Vaswani et al. (2017)” (Min et al., 2021, S. 2). Diese vortrainierten Modelle werden auch Large Pre-trained Transformer-based Language Models (PLMs) (Min et al., 2021) oder auch Large Language Models (LLMs; Hoffmann et al., 2022) genannt.

Pre-Training bezeichnet den Vorgang, dass PLMs auf einer sehr großen Menge an nicht annotierten Textdaten vorgenommen werden. Dadurch haben diese Modelle bei unterschiedlichen NLP-Aufgaben den Stand der Technik auf ein neues Niveau gehoben (See et al., 2019). Dies wurde möglich, da Pre-Training als Technik zur Verringerung der Varianz beim Training dient. Erhan und Kollegen (2010) ziehen aus dieser Verringerung der Varianz und aus der Tatsache, dass ein vorgenommenes Modell einen geringeren Trainingsfehler aufweist, den Schluss, dass das Pre-Training einen Optimierungseffekt nach sich zieht. Und vor allem in großen Datensätzen hat das einen Vorteil: "We have observed that, surprisingly, the pre-training advantage is present even in the case of really large training sets, [...]" (Erhan et al., 2010, S. 653).

In den letzten Jahren ging der Trend in die Richtung, dass die Datenmengen, die für das Pre-Training genutzt wurden, nicht so schnell anwachsen wie die immer größer strukturierten PMLs. So hat das neueste Modell der GPT-Familie GPT-3 bis zu 175 Milliarden Parameter (Brown et al., 2020) und wurde auf dem Common Crawl Datensatz trainiert, das aus bis zu einer Trillionen Wörtern zusammengesetzt ist (Raffel et al., 2020). Allerdings haben Hoffmann und Kollegen (2022) Mitte dieses Jahres gezeigt, dass dieser Trend nicht unbedingt optimal ist. Durch ihr Modell Chinchilla (Hoffmann et al., 2022), das im Vergleich zu GPT-3 nur 75 Milliarden Parameter besitzt, in den Benchmark Tests jedoch besser bis genau so gut abschneidet, zeigten sie, dass eine Optimierung der Modellgröße einhergehen muss mit der Größe des Datensatzes.

Die übliche Herangehensweise des Finetunings von BERT und auf BERT basierenden Sprachmodellen ist es, die originale Output-Schicht "einzufrieren" und durch eine neue, der Aufgabe entsprechend spezifizierten Schicht zu ersetzen. Um dann mit dem neuen Datensatz das Modell auf die Feinabstimmung zu trainieren und durch eine Verlustfunktion zu optimieren. Diese Feinabstimmung ergibt das Lernen der neuen Parameter der Output-Schicht und die Veränderung aller ursprünglichen Gewichte (Zhang, Wu, et al., 2021).

Der in dieser Arbeit verwendete Text Encoder, der aus dem XML-RoBERTa Modell besteht, das sich durch Pre-Training und daraus resultierende sehr gute Performanz in Finetuning Aufgaben auszeichnet, wird im nächsten Abschnitt genauer untersucht. Im Folgenden wird zunächst der Weg über BERT, RoBERTa hin zum letztendlich verwendeten XML-RoBERTa Modell aufgezeigt.

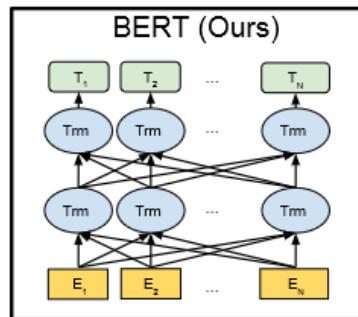
BERT

Eines der bekanntesten PLMs ist das Modell Bidirectional Encoder Representations from Transformers (BERT). Es ist ein auf dem Transformer basierendes Sprachmodell, das 2019 von Devlin und Kollegen entworfen wurde. Die Neuerung, die BERT so erfolgreich macht, ist die Implementierung von Bidirektionalität in ihrem Encoder Stack (Devlin et al., 2019). Dieser Ansatz steht im Gegensatz zur vorher geläufigen sequenziellen Vorgehensweise. Wie am Anfang dieses Abschnitts beschrieben, ging die Verwendung der Transformer mit sequenziellem Training der Modelle einher.

Devlin und Kollegen zeigten, dass es für das Modell von Vorteil ist, wenn es bidirektional trainiert wird, im Gegensatz zu unidirektional, da es so eine bessere Abbildungsmöglichkeit des Kontextes eines Wortes durch dessen Umfeld hat. Wie in Abbildung 6 zu sehen ist, haben Devlin und Kollegen (2019) für die Architektur des BERT Modells nur einen Encoder Block des Transformers verwendet.

Abbildung 6

Darstellung der bidirektionalen Struktur von BERT



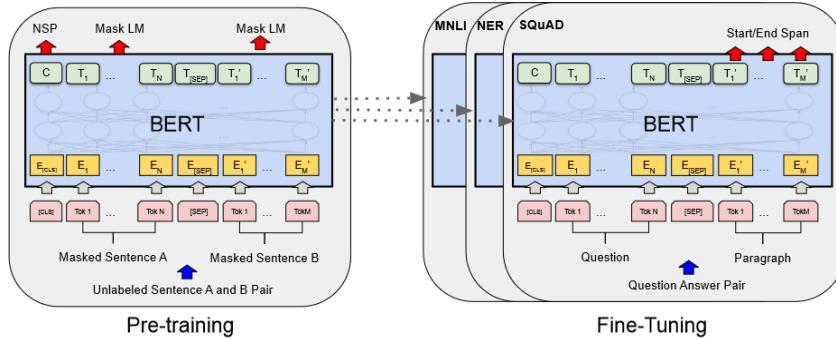
Anmerkung. Aus “Bert: Pre-training of deep bidirectional transformers for language understanding.” von J. Devlin, et al., arXiv, 2019, S. 13.

Für den Aufbau des Modells fällt folglich der Decoder Stack weg. Die sonstige Zusammensetzung aus mehreren Schichten ist identisch mit der Transformer-Architektur mit der Ausnahme, dass der Self-Attention Vorgang bidirektional ist. Das Modell wurde auf dem BookCorpus (Zhu et al., 2015) Datensatz und einem manuellen Datensatz aus englischen Wikipediaartikeln vorgenommen.

Für das Pre-Training wurden zwei Herangehensweisen verfolgt. Beide werden in der Abbildung 7 aus der Arbeit von Devlin und Kollegen (2019) zur besseren Veranschaulichung dargestellt.

Abbildung 7

Darstellung des Pre-Training und Fine Tuning Prozesses von BERT (Devlin et al., 2019, S. 3)



Anmerkung. Aus “Bert: Pre-training of deep bidirectional transformers for language understanding.” von J. Devlin, M.-W. Chang,, K. Lee, und K. Toutanova, 2019, arXiv, S. 3.

Der erste Ansatz ist das Masked Modelling (Mask LM/MLM). Um dem Problem auszuweichen, dass jedes Wort bei einem bidirektionalen Training sich indirekt selbst referenzieren könnte (Devlin et al., 2019), wird eine Prozentzahl der Input-Tokens randomisiert maskiert. Daraufhin werden diese maskierten [MASK] Tokens vom Modell durch die den Token umgebenden Kontextinformationen vorhergesagt. Dazu wird eine Klassifikationsschicht auf den Encoder-Output hinzugefügt und der Output Vektor wird mit der Embedding Matrix multipliziert, um so eine Dimensionsmatrix eines Vokabulars zu erhalten. Am Ende wird die Wahrscheinlichkeit eines jeden Wortes im Vokabular mit einer Softmax Funktion berechnet (Devlin et al., 2019).

Im Gegensatz dazu gibt es beim Ansatz des Next Sentence Prediction (NSP) keine Maskierung. Um diesen Ansatz zu verfolgen wird der Input aus dem Trainingsdatensätzen verändert. Während des Trainings erhält das Modell zwei unterschiedliche Satzpaare. Die Hälfte aller Satzpaare sind zueinander gehörende Sätze. Die andere Hälfte besteht aus randomisierten Satzpaaren. Das Ziel des Trainings ist es, dass das Modell lernt, vorherzusagen, welche Sätze zueinander gehören. Um die Abgrenzung der einzelnen Sätze zu verdeutlichen, wird ein [CLS] Token am Anfang eines Satzes platziert und ein [SEP] Token am Ende. Um vorauszusagen, ob der erste Satz tatsächlich zum zweiten gehört, wird das Ergebnis des ersten Satzes in einen Vektor der Größe 2x1 transformiert. Daraufhin wird die Wahrscheinlichkeit wieder mit einer Softmax-Matrix wie beim Masked Modelling berechnet.

Beim Pre-Training des BERT Modells wurden Masked Modelling und NSP gleichzeitig verwendet, mit dem Ziel, das kombinierte Ergebnis der Verlustfunktion so gering wie möglich zu halten.

Wie schon erwähnt kann das BERT Modell durch das Einfrieren der letzten Schicht und das Hinzufügen einer zusätzlich auf die Aufgabe abgestimmten Schicht sehr gut durch Finetuning spezialisiert werden. “[...]BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications” (Devlin et al., 2019, S. 1). Für das Finetuning werden die meisten Hyperparameter wie im Pre-Training beibehalten. Es wird meist nur eine Anpassung an der Batchgröße, der Learning Rate und an die Anzahl an Trainingsepochen vorgenommen.

RoBERTa

Wie am Beispiel von Chinchilla (Hoffmann et al., 2022) zu sehen, ist die Optimierung und Nutzung von Parametern ein wichtiger Bestandteil, um die Leistung eines Modells zu beeinflussen. Auch im Fall der Arbeit von Liu und Kollegen (2019) ist dies ein wichtiger Anhaltspunkt.

In ihrer Arbeit stellten sie eine Replikation der Studie von Devlin und Kollegen (2019) vor, jedoch mit dem Unterschied, dass sie die Hyperparameter und Trainingsdatengröße veränderten. Wie sie feststellen, hat die Veränderung der Parameter ausschlaggebende Auswirkungen auf die Performanz des BERT Modells: “[...] hyperparameter choices have significant impact on the final results” (Liu et al., 2019, S. 1). Sie haben herausgefunden, dass BERT stark untertrainiert ist und schlagen in der Arbeit einen besseren Methode vor, wie man BERT Modelle trainieren kann und nennen dieses RoBERTa. Die Modifikationen, die sie vornehmen, sind:

1. Das Modell länger trainieren mit einem noch größeren Datensatz;
2. Das Ziel des NSP, den nächsten Satz vorherzusagen, wurde entfernt;
3. Das Modell auf längeren Input Sequenzen trainieren und größere Batches benutzen;
4. Das Masking wird nun dynamisch auf die Trainingsdaten angewandt;

Außerdem wurde ein neuer Datensatz (CC-NEWS) (Liu et al., 2019) hinzugefügt, um die Auswirkung auf die Größe der Trainingsmenge besser zu kontrollieren. Diese Veränderungen haben dazu geführt, dass die Ergebnisse von BERT verbessert werden konnten.

XLM-RoBERTa (XLM-R)

Das in dieser Arbeit für den Text Encoder verwendete Sprachmodell XLM-RoBERTa (XLM-R) basiert auf den Erkenntnissen aus der Studie über RoBERTa von Liu und Kollegen (2019). In der Studie von Conneau und Kollegen aus dem Jahr 2020 werden die Arbeiten von BERT und RoBERTa als Grundlagen verwendet.

Das Ziel ihrer Arbeit war es zu zeigen, dass das Pre-Training multilingualer Sprachmodelle in großem Maßstab zu erheblichen Leistungssteigerungen führt. Und ihr bestes Modell übertraf mBERT um bis zu 23 Prozent (Conneau et al., 2020).

Um dies zu erreichen, trainierten sie das transformerbasierte Modell auf zwei Terabyte ungefilterten Daten des CommonCrawl Datensatzes mit über 100 unterschiedlichen Sprachen. Im Abschnitt 3.5 wird über die Konzeption des Text Encoders für das multilinguale CLIP Modell genauer auf die Verwendung des XLM-RoBERTa (XLM-R) Modell als Text Encoder eingegangen. Im Abschnitt 4.4, der sich um die Implementierung dreht, wird die praktische Einbettung des multilingualen Sprachmodells als Text Encoder genauer beschrieben.

2.3 CLIP

Das CLIP-Modell vereint all die bis zu dieser Stelle erwähnten Grundlagen und Fortschritte der letzten Jahre aus der NLP-Forschung und verbindet diese mit denen aus der CV-Forschung. Das CLIP-Modell wird in dieser Arbeit als Basis verwendet, da für den Wettbewerb multimodale Daten als Grundlage dienen und somit die Nutzung eines multimodalen Modells vornötig ist.

Da der Entwurf von multimodalen Modellen in den letzten Jahrzehnten stetig zugenommen hat (Xu et al., 2022) und das Kernstück dieser Arbeit ein solches Modell ist, sollte kurz auf die Eigenschaften von multimodalen Modellen eingegangen werden. Das multimodale Lernen beschreibt die unterschiedliche Art der Eingabedaten wie zum Beispiel Bild und Text (Ngiam & Lee, 2011) und die Möglichkeit eines einzelnen Modells, aus diesen unterschiedlichen Eingabedaten ein Ergebnis zu erzielen. Vor allem bei multimodalen Transformer Modellen ist die Verwendung von unterschiedlichen Daten laut Xu und Kollegen (2022) in zwei Schritten umsetzbar. Als Erstes muss der Input tokenisiert werden und als Zweites muss ein Embedding Space ausgewählt werden, der die Token repräsentiert, bevor die Daten an den Transformer übergeben werden.

CLIP ist ein multimodales Modell, das auf einer sehr umfangreichen Menge an Bild- und Textdaten vtrainiert wurde. Wie schon in der Einleitung beschrieben ist der Aufbau in einen Image Encoder und einen Text Encoder unterteilt, welche die visuellen und textuellen Inputs als Repräsentation in Vektoren umwandeln. Aus diesen Vektoren wird eine Matrix erstellt und die Cosine Similarity der Bilder und Texte generiert. Zusätzlich wird durch Contrastive Loss dem Modell die Ähnlichkeit der Bilder und Texte vermittelt. Im Kern besteht das CLIP-Modell aus einer vereinfachten Form des ConVIRT Modells (Zhang et al., 2022), das von Grund auf trainiert wurde. Contrastive Language-Image Pre-Training beschreiben Radford und Kollegen (2021) als effiziente Methode, um von der Überwachung der natürlichen Sprache zu lernen.

Natural Language Supervision

Diese Methode steht im Mittelpunkt der Entwicklung von CLIP. Die Idee des Natural Language Supervision beschreiben sie als die Möglichkeit des Modells, durch Überwachung von natürlicher Sprache während des Trainings einen quasi-semantischen Kontext aufzubauen. Dies ist möglich geworden, weil mittlerweile viele Verbesserungen im Bereich des *deep contextual representation learning* (Radford et al., 2021) gemacht worden sind. Sie argumentieren, dass das Lernen durch natürliche Sprache einige Vorteile gegenüber anderen Trainingsmethoden hat. So ist es um ein Vielfaches einfacher Daten für das Natural Language Supervision zu skalieren, im Gegensatz zu Crowdsourced Labeling für die Bildklassifizierung. Gegenüber supervised oder unsupervised Lernmethoden ist es bei dieser Methode möglich, dass passiv durch die riesigen Textmengen Informationen übernommen werden und nicht nur eine Repräsentation aus den Daten übernommen wird, sondern Repräsentationen mit der natürlichen Sprache verknüpft werden, was einen flexiblen Zero-Shot-Transfer ermöglicht (Radford et al., 2021).

Die Grenzen der Skalierbarkeit des CLIP Modells wurden an 8 unterschiedlich großen CLIP Modellen untersucht und die Übertragungsleistung zwischen diesen stellte eine gleichmäßig vorhersagbare Funktion der Rechenleistung dar. Ähnlich wie die Modelle der GPT-Familie lernt CLIP durch das Training ein breites Spektrum an Aufgaben zu erfüllen. Radford und Kollegen (2021) zählen dazu zum Beispiel OCR, Geolokalisierung, Handlungserkennung und einige Weitere.

Möglich sind diese Errungenschaften durch die Zusammenstellung eines umfangreichen Datensatzes. Vorherige Arbeiten im multimodalen Bereich nutzen vor allem drei bekannte Datensätze, die auch im Stand der Forschung bereits erwähnt wurden. Diese drei sind Visual Genome (Krishna et al., 2017), MS-COCO und YFCC100M (Thomee et al., 2016).

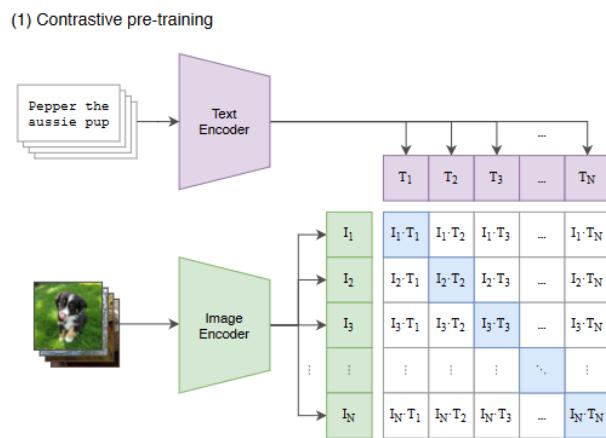
Für Radford und Kollegen (2021) stellte sich allerdings heraus, dass diese Datensätze für ihre Ambitionen zu klein sind. YFCC100M umfasst zwar 100 Millionen Bilder, allerdings stellten sich die Metadaten zu diesen als zu wenig und von variierender Qualität dar. Deswegen wurde ein neuer Datensatz aus Quellen aus dem Internet zusammengetragen, der aus 400 Millionen Bild- und Text-Paaren besteht. Diese sind jedoch, wie Anfangs dieser Arbeit bereits erwähnt, ausschließlich auf Englisch.

Architektur

Die Architektur des CLIP Modells wird in Abbildung 8 dargestellt. Zu erkennen ist der Vorgang des Pre-Trainings durch die Einspeisung der Bild- und Textinputs auf der linken Seite.

Abbildung 8

Darstellung der CLIP-Architektur (Radford et al., 2021, S. 2)



Anmerkung. Aus “Learning transferable visual models from natural language supervision.” von A. Radford et al., 2021, Proceedings of Machine Learning Research, 139, S. 2.

Text Encoder

Der originale, monolinguale Text Encoder ist ein Transformer mit Modifikationen, die von Radford et al. (2019) übernommen worden sind. Trotz des Austauschs des Text Encoders soll hier der Vollständigkeit halber und zur besseren Vergleichbarkeit die originale Gestaltung des Text Encoders kurz beschrieben werden. Die Basis des Encoders besteht aus einem 12-schichtigen Modell mit 63 Millionen Parametern. Zusätzlich hat das Modell 8 Attention Heads und die Breite des Modells umfasst eine Vektorgröße von 512. Für eine

optimale Recheneffizienz wurde die Länge der Input Sätze auf 76 Zeichen beschränkt und werden eingegrenzt mit [SOS] und [EOS] Token.

Das Encoding der Input Sätze wird durch Byte Pair Encoding mit einer Vokabulargröße von 49.152 Zeichen vorgenommen. Ähnlich der Herangehensweise zu BERT werden die Sätze zusätzlich mit Masked Self-Attention maskiert und auf eine festgesetzte Länge auf 77 Zeichen begrenzt.

Image Encoder

Der Image Encoder, der auch im multilingualen Modell verwendet wird, hat ResNet-50 (He et al., 2016) als Basisarchitektur. Allerdings wurden durch die Arbeit von Radford und Kollegen (2021) einige Modifikationen an der Basisstruktur von ResNet-50 vorgenommen. Zum einen wurden die Verbesserungen von ResNet-D (He et al., 2019) übernommen und zum anderen wurde das vorherige Average Pooling System zu einem Attention basierten System umgewandelt, das sie von Zhang (2019) verwenden. Das CNN des Image Encoders besteht aus 48 Schichten mit einer Schicht, die mit Multi-Head Attention Pooling das übergebene Bild bearbeitet.

Noch eine zweite Architektur wurde von Radford und Kollegen (2021) entworfen. Dazu wurde der neuere Vision Transformer (ViT) (Dosovitskiy et al., 2021) herangezogen und es wurden nur kleinere Modifikationen, wie das Hinzufügen einer Normalisationsschicht für die Position Embeddings, vorgenommen. Der ViT ist ein bildorientiertes Netzwerk, das auf dem Encoder System der Transformer Architektur aufbaut.

Zusätzlich zu den Modifikationen an beiden Encodern werden die Outputdimensionen aneinander angepasst. Für den Image Encoder wird die Breite, die Tiefe und die Auflösung gleichmäßig vergrößert, und im Fall des Text Encoders wird die Breite des Modells dem des Image Encoders angepasst.

Pre-Training

Der Vorgang des Pre-Training, der in Abbildung 8 dargestellt ist, beginnt mit der Einspeisung der Bild- und Textdaten in die entsprechenden Encoder. Durch die Größe des Datensatzes ist Overfitting kein zu beachtendes Problem (Radford et al., 2021). Das CLIP Modell wird von Grund auf ohne pre-trained Gewichtungen der zugrunde liegenden Modelle trainiert. Zusätzlich wird die Text- und Bildtransformation vereinfacht und der Temperaturparameter, welcher die Größe der Logits der Softmax-Matrix kontrolliert, wird direkt während des Trainings optimiert. Insgesamt wurden fünf unterschiedlich große ResNet Netze und drei unterschiedlich große VIT Modelle als Image Encoder trainiert. Für die ResNet Netze wurde

ein ResNet-50, ein ResNet-101 und drei weitere ResNet-50 Netzte, die dem EfficientNet (Tan & Le, 2019) Ansatz folgen, in der Größe 4x, 16x und 64x trainiert.

Diese werden als RN50x4, RN50x16, und RN50x64 bezeichnet. Für den Image Encoder des multilingualen Modells wird das RN50x4 Modell verwendet. Parallel werden für die ViTs die Bezeichnungen ViT-B/32, ViT-B/16, und ViT-L/14 gemäß ihrer Größe verwendet. Alle Modelle wurden für 32 Epochen trainiert. Als Optimierer wurde, wie auch später während des Trainings des multilingualen Modells in dieser Arbeit, der Adam Optimizer (Kingma & Ba, 2017) benutzt. Der Temperaturparameter wurde zu Anfang auf 0.07 festgelegt und über die Trainingsdauer abgeschnitten, um eine Skalierung der Logits über 100 zu verhindern.

Aufgrund der umfassenden Trainingsdaten benötigte das größte ResNet Modell, RN50x64, 18 Tage auf 592 GPUs, während das größte ViT Modell, ViT-B/32, 12 Tage auf 256 GPUs benötigte. Dieses umfassende Pre-Training macht es möglich, dass das CLIP Modell als neuer Stand der Technik angesehen wird und durch Zero Shot Klassifikation ohne Finetuning eine sehr gute Performance erreicht.

2.4 Multilinguale, multimodale Modelle

Der große Nachteil des CLIP Modells ist, dass es durch den monolingualen Datensatz und Text Encoder nicht auf multilinguale Aufgaben ausgelegt ist. Allerdings gibt es, wie im Abschnitt 1.1 in der Einleitung erwähnt, schon mehrere Ansätze für multimodale, multilinguale Modelle. Auch mit der Basis der CLIP Architektur gibt es schon Herangehensweisen, die hier ebenfalls dargestellt werden müssen. Die meisten der früheren Modelle beschränken sich auf nur wenige Sprachen. So wurde 2016 von Rajendran und Kollegen ein Modell entwickelt, das sich auf Englisch, Französisch und Deutsch spezialisiert hat, mit Englisch als Pivot Sprache. Ähnlich zu diesem Modell gab es eine Arbeit im Jahr 2017 von Gella und Kollegen, die in ihrem Modell das Bild als Pivotpunkt nahmen und sich auf die Sprachen Deutsch und Englisch festlegten. Ebenfalls 2017 erschien die Arbeit von Calixto und Kollegen, in der wie in dieser Arbeit die visuelle Komponente durch ein vortrainiertes Modell gehandhabt wurde. Für die textuelle Umsetzung wurde ein RNN Sentence Encoder erstellt und es wurde sich wieder auf die Sprachen Englisch und Deutsch fokussiert. Nach der Erstellung des Transformers wurde 2019 von Huang und Kollegen das erste Mal Attention für ein multilinguales, multimodales Modell verwendet. Auch in diesem Fall wurde sich auf die Sprachen Englisch und Deutsch beschränkt.

Das erste übergreifend multilinguale, multimodale Modell, das auch in dieser Arbeit später als Optimierungsstrategie herangezogen wird, wurde 2020 von Reimers und Kollegen in ihrer Arbeit erstellt. Das Ziel ihrer Arbeit war es, eine neue Methode vorzustellen, die es ermöglichte, schon existierende Satz Embeddings auf andere Sprachen auszuweiten.

Durch eine Teacher-Student-Architektur (Hinton et al., 2015) trainierten sie ein Testmodell auf über 50 Sprachen. Sie benutzen für das Teacher Modell ein SBERT Sprachmodell und für das Student Modell das schon beschriebene XLM-RoBERTa (XLM-R) Modell.

Das erste multilinguale, multimodale Modell, das sich mit der CLIP Architektur auseinandersetzt, haben Carlsson und Kollegen in 2022 entworfen. Im Gegensatz zu dieser Arbeit wurde auch bei ihnen auf eine Teacher-Student Architektur zurückgegriffen, um den Text Encoder auf Multilingualität “umzulernen”: “we propose to use cross-lingual teacher learning to re-train the textual encoder for various non-English languages” (Carlsson et al., 2022, S. 1). Der originale Text Encoder von CLIP wird von ihnen durch das vortrainierte mBERT (Devlin et al., 2019) Modell ausgetauscht und auf 68 Sprachen trainiert. Als Ergebnis stellen sie fest, dass ihr Modell auf multilingualen Aufgaben zwar gut, im Vergleich zur Performance des originale CLIPs allerdings eher schlecht abschneidet: “Starting by comparing the M-CLIP against the original CLIP in English, we find that M-CLIP performs noticeably worse, [...]”(Carlsson et al., 2022, S. 4).

Zusammenfassend kann man festhalten, dass die Erstellung eines multilingualen, multimodalen Modells mit vielen grundlegenden Methoden im Bereich NLP und CV zusammenhängt. In diesem Abschnitt der Arbeit wurden die Grundlagen und hintergründigen Theorien dargestellt, die es möglich machen, ein multilinguales, multimodales Modell auf der Basis von CLIP zu erstellen. Angefangen mit der Darstellung des Aufbaus und der Größe des WIT-Datensatzes wurde deutlich, wie umfangreich die Bearbeitung des Wettbewerbs ist. Mit der Beschreibung der Geschichte der NLP-Forschung wurde die Basis dargestellt, die die Beschleunigung im letzten Jahrzehnt möglich macht.

Es wurde gezeigt, wie der Entwurf der Transformer Architektur die NLP Landschaft verändert hat und die Erstellung von neuen Sprachmodellen auf ein neues Niveau gehoben wurde. Seitdem bauen viele Sprachmodelle auf dem Transformer System und auch aufeinander auf, wie das in dieser Arbeit als Text Encoder verwendete Modell XLM-RoBERTa (XLM-R). Es wurde deutlich, dass all die in diesem Abschnitt besprochenen Fortschritte das CLIP Modell möglich machten, dessen Aufbau im oberen Abschnitt beschrieben wurde. Weiterhin wurden die Bestrebungen der letzten Jahre, multilinguale, multimodale Modelle auch mit CLIP als Basis zu erstellen dargelegt, die, anders als in dieser Arbeit, auf die Teacher-Student-Struktur setzen.

Mit dieser umfangreichen Darlegung der Grundlagen als Basis, ist es nun möglich, ein Modell zu konzipieren, das die CLIP Architektur verwendet, um den Wettbewerb von Wikipedia zu bearbeiten.

3. Konzeption

Um bei der Konzeption des Arbeitsablaufs einen besseren Überblick zu haben, wird sich bei der Struktur der Konzeption und Implementation am Ablaufmodell des Cross-Industry Standard Process For Data Mining (CRISP-DM) (Wirth & Hipp, 2000) orientiert. In diesem Abschnitt der Arbeit wird das Design der Datenbearbeitung, der Benchmark, des multilingualen Modells und der Evaluation dargestellt, um im nächsten Kapitel 5 die konkrete Implementation zu erläutern. Im nächsten Schritt dieses Abschnitts wird ein Überblick über das CRISP-DM gegeben. Danach folgen die weiteren Schritte, die sich aus dem Ablaufmodell ableiten. Angefangen mit dem Data Understanding und der darauffolgenden Data Preparation, in der gezeigt wird, wie die Daten möglichst optimal durch die gezogenen Schlüsse aus dem Data Understanding, für die Bearbeitung präpariert werden können.

Im Anschluss wird die Planung für die Erstellung der Benchmark durch Zero Shot CLIP erläutert. Und in den beiden letzten Schritten werden die Kernstücke der Arbeit konzipiert. Zuerst wird auf die Modellierung des multimodalen, multilingualen Modells eingegangen. Es folgt dann die Ausarbeitung der zwei Optimierungsstrategien. Die erste Strategie behandelt das multilinguale Prompt Engineering integriert in den Ablauf des multilingualen Modells. Die zweite Strategie beschäftigt sich mit der Ausweitung auf das in der Einleitung erwähnte Sentence Transformer Modell, das auf der Arbeit von Reimers und Gurevych (2020) beruht.

3.1 CRISP-DM

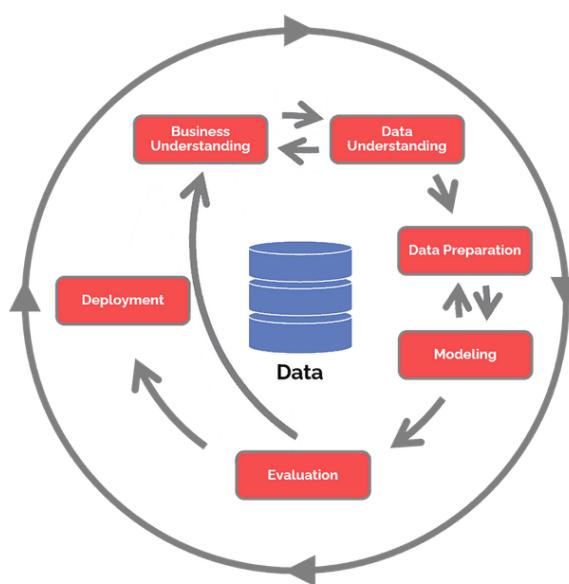
Das CRISP-DM entstammt, wie man an dem Namen erkennen kann, eigentlich dem Data Mining Bereich und wurde 1996 vom CRISP-DM Konsortium, bestehend aus der DaimlerChrysler AG, SPSS, NCR und OHRA erstellt. Teilweise gesponsert wurde die Erstellung des Prozessmodells von der Europäischen Kommission unter dem ESPRIT Programm (Wirth & Hipp, 2000). Trotz des Alters des Prozessmodells wird es auch heute

noch in den Bereichen des Data Mining und auch im Machine Learning als Industriestandard verwendet (Studer et al., 2021).

Das Ziel des Modells ist es, den Prozess der Erstellung von datengetriebenen Anwendungen reproduzierbar und überschaubar zu machen: “CRISP-DM process model aims to make large data mining projects less costly, more reliable, more repeatable, more manageable, and faster” (Wirth & Hipp, 2000, S. 2). Deswegen wird sich in dieser Arbeit an einigen der Grundabläufe des CRISP-DM orientiert.

Abbildung 9

Darstellung des CRISP-DM Prozessmodells basierend auf (Chapman et al., 2000, S. 13)



Anmerkung. Aus “What is CRISP DM?” von N. Holz, 2022, Data Science Process Alliance.

Wie Abbildung 9 zeigt, besteht CRISP-DM aus 6 dieser Abläufe, die miteinander verbunden sind und in einem iterativen Fluss zueinander stehen. Die Basis des Prozessmodells sind die Phasen des (1) Business Understanding, (2) Data Understanding, (3) Data Preparation, (4) Modeling, (5) Evaluation und (6) Deployment. Da die Phase des Deployment für diese Arbeit nicht relevant ist, wird sie in der späteren Bearbeitung nicht weiter beachtet.

(1) Als Einleitung in den Prozessablauf wird die Business Understanding Phase angeführt. Hier geht es darum, die Anforderungen und Ziele zu verstehen und einen vorläufigen Lösungsprozess zu entwerfen (Wirth & Hipp, 2000). In Bezug zu dieser Arbeit wurde dieser Schritt schon in der Einleitung zu Kapitel 2 mit der genaueren Auseinandersetzung mit den Zielen und der Struktur des Wettbewerbs erledigt und

wird im weiteren Verlauf ähnlich zu der (6) Deployment Phase nicht weiter in Betracht gezogen.

- (2) Die zweite Phase wird als Data Understanding bezeichnet. Diese Phase startet im Standardablauf mit dem Sammeln und einer vorläufigen Analyse der Daten. Dieser Schritt soll einen ersten oberflächlichen Überblick geben, um vorläufige Einblicke in die Daten zu gewinnen. So können zum Beispiel Probleme mit der Datenqualität früh erkannt werden oder interessante, beziehungsweise verborgene Informationen entdeckt werden (Chapman et al., 2000). Im Allgemeinen gibt es eine enge Verbindung zwischen der Business Understanding Phase und dem Data Understanding. Es gilt, dass zumindest ein rudimentäres Verständnis der Daten erlangt werden soll, um es möglich zu machen, einen Projektplan zu gestalten (Chapman et al., 2000). Auch die Visualisierung des Datensatzes hilft in dieser Phase, ein Verständnis für die Daten zu erhalten.
- (3) In der Phase der Data Preparation werden alle Aktivitäten untergebracht, die sicherstellen, dass am Ende dieser Phase ein nutzbarer Datensatz erstellt wird. Mit nutzbarer Datensatz ist ein Datensatz gemeint, der aus den Rohdaten des Data Understanding erstellt wurde und mit dem in der Modelling Phase das Modell erfolgreich trainiert werden kann. Diese Aufbereitungsaufgaben der Daten können an einem späteren Zeitpunkt des Prozessmodells immer wieder nötig sein und müssen wahrscheinlich mehrfach und nicht in einer bestimmten Reihenfolge durchgeführt werden (Wirth & Hipp, 2000). Zu diesen Aufgaben gehören zum Beispiel Datenbereinigung, Erstellung neuer Attribute oder das Entfernen von diesen, sowie die Umwandlung von Daten.
- (4) Der Vorgang des Modelling beschreibt in Hinblick auf den Machine Learning Bereich die Auswahl oder Erstellung eines Modells, das mit den präparierten Daten arbeiten kann. Im Fall dieser Arbeit besteht diese Phase aus der Planung und Umsetzung des neuen multilingualen Modells anhand der CLIP Architektur. Es gibt einen engen Zusammenhang zwischen der Data Preparation und dem Modelling (Chapman et al., 2000), denn manchmal ist es nötig, den ursprünglichen Datensatz gegebenenfalls anzupassen. Auch die Vorgänge des Trainings und Optimierung fallen in diese Phase.
- (5) In der Evaluationsphase wird das erstellte Modell überprüft und dessen Qualität evaluiert. Dies geschieht durch die Anwendung des Modells auf einem Evaluierungsset des Datensatzes, das vor dem Training optimalerweise in der Data Preparation Phase von diesem getrennt wurde. Für die Konzeption der Arbeit gilt es nun, diese Phasen des Prozessmodells angewandt auf das Thema dieser Arbeit im Einzelnen zu erläutern.

3.2 Data Understanding

“First step in building an application of artificial intelligence is to understand the data.[...]Some data can be missing, some data can be mal-formatted, etc. It is the first task to get familiar with the data. Clean up the data as necessary” (Joshi, 2020, S. 21).

Wie in der obigen Kurzbeschreibung zu CRISP-DM und in der Empfehlung von Joshi dargestellt, sollen in der Phase des Data Understanding als erster Schritt die zur Verfügung stehenden Daten verstanden werden.

Die erste rudimentäre Darstellung des Datensatzes wurde vorher schon unter Abschnitt 2.1 durchgeführt. In diesem Abschnitt geht es um das tiefergehende Verständnis der Daten und die daraus resultierenden Anwendungsrichtlinien für die Implementation. Joshi (2020) stellt für das erfolgreiche Data Understanding in seiner Arbeit drei Schritte fest:

1. Das Verstehen von Entitäten
2. Das Verstehen von Attributen
3. Das Verstehen von Datentypen

Entitäten innerhalb des Machine Learning sind nach Joshis (2020) Beschreibung Gruppen an Daten, die nach konzeptionellen Themen getrennt voneinander sind. Außerdem schreibt er: “An entity typically represents a table in a database, or a flat file, e.g., comma separated variable (csv) file, or tab separated variable (tsv) file” (Joshi, 2020, S. 22). Im Fall dieser Arbeit ist die Entität zusammengesetzt aus allen Trainingsdaten des WIT Datensatzes. Dazu gehören alle sechs Datensätze der **train-{0000x}-of-00005.tsv** Dateien, sowie alle Daten in dem Ordner **image_data_test**. Nicht Teil der Entität in Bezug auf die Relevanz des Data Understanding und der weiteren Abläufe sind die Dateien **test.tsv**, da ein eigener Testsatz erstellt wird, **test_captions_list.csv** und die **sample_submission.csv** Datei. Im Großen und Ganzen besteht die Entität, die für diese Arbeit wichtig ist, also aus den Trainingsdaten.

Eine Entität enthält mehrere Attribute, die im Machine Learning oft Features genannt werden. Die Attribute beschreiben die einzelnen Spalten eines Datensatzes. “Each attribute can be thought of as a column in the file or table” (Joshi, 2020, S. 22). Die Attribute der Entität bestehen aus den Spalten language, page_url, image_url, page_title, section_title, caption _reference_description, caption_attribution_description, caption_ alt_text_description, mime_type, original_height, original_width, is_main_image, attribution

`_passes_lang_id`, `page_changed_recently`, `context_page_description`, `context_section_description`, und `caption_title_and_reference_description`.

Es gilt die für die Aufgabe wichtigsten Attribute zu identifizieren und auszuwählen. Durch den Vorgang der Dimensionality Reduction (Desai, 2022), also der Reduzierung der Dimensionalität der Entität, kann ein neuer Datensatz mit den wichtigsten Spalten generiert werden. In diesem Fall bezeichnet Dimensionalität die Anzahl an Spalten in der Entität. Bei diesem Vorgang werden während der Data Preparation redundante oder nicht notwendige Features entfernt und so die Dimensionalität der Entität reduziert. Dafür müssen zunächst die wichtigen Spalten identifiziert werden. Da in der Untersuchung des Datensatzes in Kapitel 2.1 und 2.1.1 herausgestellt wurde, dass das Ergebnis eine Datei mit Bild-ID und der zugehörigen `caption_title_and_reference_description` ist und das Ziel des Wettbewerbs die Erstellung eines Modells mit Bild- und Textabgleichs ist, sind die zwei wichtigsten Attribute, die beachtet werden müssen die `image_url` Spalte und die `caption_title_and_reference_description`. Die `image_url` enthält alle URLs zu den originalen Bilddateien. Die `caption_title_and_reference_description` enthält die entsprechenden zuweisbaren Bildbeschreibungen. Neben der `image_url` gibt es jedoch noch andere Wege, an die Bilddateien zu gelangen, die in der Implementation untersucht werden müssen. Über die Option der bereitgestellten `b64_bytes` können die Bilddateien in base64 kodiertem Format in der Auflösung von 300 Pixeln benutzt werden. Ebenso ist es möglich, über den Weg der `image_data_train` die Bilder im Originalzustand herunterzuladen. Aufgrund der Größe des Datensatzes von 250 GB wird diese letzte Option für die Arbeit jedoch nicht in Betracht gezogen.

Das Verstehen der Datentypen bezieht sich auf die Daten innerhalb der zu betrachteten Attribute. Neben der grundlegenden Frage nach der Art des Datentyps wie String, Integer, Float usw. hängt das Verstehen des Datentyps auch mit der Auswirkung auf den Prozess und des Endergebnisses zusammen. In Bezug zu der Arbeit sind hier Fragen zu klären wie die Frage nach der durchschnittlichen Sequenzlänge der Caption-Strings. Da der Input des Text Encoders auf 77 Zeichen beschränkt ist, müssen die Längen der Captions überprüft werden. Ansonsten kann es hier zu einem Informationsverlust kommen. Auch ist die Frage nach der Verteilung der Sprachen im Datensatz interessant. Gibt es vielleicht Sprachen, die viel häufiger im Datensatz vorkommen und so das Ergebnis zugunsten einer Sprache verbessern? Diese und ähnliche Fragen müssen im Data Understanding der Implementierung geklärt werden. Um dies zu gewährleisten, ist ein wichtiger Punkt während des Data Understanding, die Daten zu visualisieren. Dafür werden in dieser Arbeit Werkzeuge verwendet, die im Folgenden näher beschrieben werden.

Matplotlib

Matplotlib ist eine Python Bibliothek für die Erstellung von Diagrammen aus Arrays. Sie wurde von Hunter (2007) entwickelt, um in erster Linie die Visualisierung von wissenschaftlichen, technischen und finanziellen Daten zu ermöglichen. Die meisten der Matplotlib-Werkzeuge befinden sich in dem Submodul Pyplot und werden normalerweise wie auch in dieser Arbeit unter dem Alias plt importiert.

Seaborn

Eine weitere neue Python Bibliothek für die Visualisierung von statistischen Daten, die in dieser Arbeit verwendet wird, ist Seaborn. Für die Beantwortung der oben gestellten Fragen hinsichtlich der Eigenschaft der Attribute bietet Seaborn eine deklarative, datenorientierte API durch seine Funktionen, um diese Fragen in Grafiken übersetzen zu können und somit einfacher beantwortbar zu machen (Waskom, 2021).

3.3 Data Preparation

Aus den Ergebnissen des Data Understanding und der Visualisierung der Daten können Handlungsfolgen für die Phase des Data Preparation gezogen werden. Das Ziel dieser Phase ist es, ein fertiges Data Frame zu haben, mit dem man das im Modelling zu erstellende Modell erfolgreich trainieren kann.

Dazu sind mehrere Schritte notwendig. Zuerst sollte die angesprochene Reduzierung der Dimensionalität erfolgen. Die beiden Attribute *caption_title_and_reference_description* und die Bilder, sei es aus der *image_url* oder durch andere Quellen, werden als einzige Attribute aus dem originalen Datensatz übernommen.

Im Anschluss folgt der Schritt der Datenbereinigung. Hier wird der neue, aus den beiden Attributen bestehende Datensatz auf fehlende oder defekte Daten untersucht und diese daraufhin entfernt. Als Zwischenschritt schließt sich diesem Prozess eine eventuelle Erweiterung der Attribute an. Dies kann zum Beispiel aus dem Hinzufügen von Token, die dem XLM-RoBERTa (XLM-R) Modell zu eigen sind, zu den Captions bestehen.

Als letzter Schritt wird der präparierte Datensatz in ein Trainings-, ein Test- und einen Evaluationsdatensatz aufgeteilt, um so das Modell trainieren zu können. Damit diese Vorgaben umsetzbar sind, wird auch in dieser Phase der Implementation auf Python Bibliotheken zurückgegriffen.

Pandas

Dazu gehört als Erste die Pandas Bibliothek. Dies ist ein Open Source Paket, das eine leistungsstarke Datenmanipulation in Python ermöglicht (McKinney, 2010). Außerdem stellt es schnelle und flexible Datenstrukturen bereit, mit denen die Arbeit mit "relationalen" oder "gelabelten" Daten einfacher gestaltet werden kann. Als mittlerweile wichtige Basis im Machine Learning wird durch Pandas die Bearbeitung von Datensätzen vorgenommen (McKinney, 2013).

Mit den beiden primären Datenstrukturen von Pandas, Series und Data Frame kann man den Großteil aller typischen Anwendungsfälle der Datenverarbeitung bewältigen (The pandas development team, 2022). So werden diese Datenstrukturen auch für die Datenbearbeitung während der Implementation in dieser Arbeit verwendet. Zu den häufigsten Arten der Datenbearbeitung, die sehr gut mit Pandas vorzunehmen sind, gehört die Handhabung von fehlenden Daten (dargestellt als NaN) in Fließkomma- und Nicht-Fließkommadaten. Ebenso wie die Size Mutability, also das Einfügen oder Löschen von Spalten aus Data Frames und höherdimensionalen Objekten. Das Pandas-Paket ist aufgebaut auf dem NumPy Python Paket (The pandas development team, 2022).

Numpy

"NumPy is the foundation upon which the scientific Python ecosystem is constructed" (Harris et al., 2020, S. 357). Wie das Zitat von NumPy erahnen lässt, steht NumPy als Fundament für die Bearbeitung von statistischen Daten und Anwendungen, die mit statistischen Daten in Python arbeiten. Die Bibliothek ermöglicht durch die Datenstruktur des NumPy-Arrays das effiziente Arbeiten mit Vektoren, Matrizen und allgemein mehrdimensionalen Arrays. Das NumPy-Array ist eine Datenstruktur, die die mehrdimensionalen Arrays, die auch Tensoren genannt werden, effizient speichern und auf diese zugreifen kann, um so eine Berechnung mit diesen zu ermöglichen.

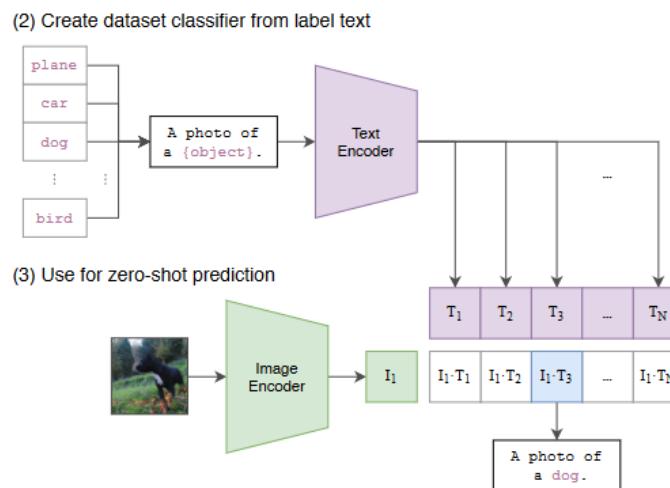
3.4 Benchmark Zero Shot CLIP

Vor dem eigentlichen Designen des multilingualen, multimodalen Modells wird hier der Ablaufplan für die Benchmark durch Zero Shot CLIP dargestellt. Zero Shot Learning wird in der Arbeit von Radford und Kollegen (2021) breiter interpretiert als üblich. Sie untersuchen im Zusammenhang mit CLIP und Zero Shot die Generalisierung des Modells auf ungesehene Datensätze. Im Gegensatz zu den meisten Forschungen auf dem Gebiet des Unsupervised Learning, die sich auf die Fähigkeiten zum Erlernen von Repräsentation

fokussieren, setzen sie auf die Untersuchung von Zero Shot Transfer als Weg Machine Learning Modellen die Fähigkeit zum Erlernen von Aufgaben zu ermöglichen (Radford et al., 2021). Als unerwarteter Nebeneffekt wurde das Erlernen von Aufgaben erstmals von Liu und Kollegen 2018 beobachtet, als sie ein Sprachmodell trainierten, welches Wikipediaartikel generieren sollte und dabei von sich aus die richtige Zuweisung von Namen erlernte. Auch GPT (Radford & Narasimhan, 2018) zeigte, dass die Performanz von vier heuristischen Zero Shot Transfer Methoden über das Pre-Training hinweg ohne Supervision verbessert wurde. Diese Ergebnisse führten zur Erstellung von GPT-2 (Radford et al., 2019), das ausschließlich die Möglichkeiten der Erlernung von Aufgaben von Sprachmodellen durch Zero Shot Transfer untersuchte.

Abbildung 10

Darstellung des Zero Shot (Radford et al., 2021, S. 2)



Anmerkung. Aus “Learning transferable visual models from natural language supervision.” von A. Radford et al., 2021, Proceedings of Machine Learning Research, 139, S. 2.

Im Fall von CLIP ist der Ablauf der Zero Shot Klassifikation in Abbildung 10 abgebildet. CLIP wird vortrainiert, um vorauszusagen, ob ein Bild- und Textinput im Datensatz gegenübergestellt wurde (Radford et al., 2021). Bei der Zero Shot Klassifikation wird diese Eigenschaft abgerufen. Als erster Schritt werden von den beiden Encodern die Feature Embeddings von Bild und Text kodiert. Daraufhin wird die Kosinus-Ähnlichkeit der Embeddings berechnet, skaliert durch einen Temperaturparameter τ , und über eine Softmax-Matrix in eine Wahrscheinlichkeitsverteilung normalisiert (Radford et al., 2021). Die Klasse mit der höchsten Wahrscheinlichkeit wird dann ausgewählt.

Für die Umsetzung in der Implementation müssen die Bilder zunächst aus den URLs der `image_url` Spalte abgefragt werden und zusammen mit den Beschreibungen aus der `caption_title_and_reference_description` Spalte den Encodern übergeben werden. Diese kodieren daraufhin die Embeddings und die Performanz der Zero Shot Klassifikation kann durch eine Evaluation, die im späteren Verlauf noch beschrieben wird, erfolgen. Für einen Vergleich des originalen Zero Shot wäre es von Interesse, wenn zunächst nur der englische Anteil des Datensatzes überprüft wird, um so einen Vergleich zu haben für die Performanz auf dem multilingualen Datensatz. Nachdem eine Benchmark mit dieser Zuweisung der Bilder und Texte erfolgt ist, kann die Modellierung des multilingualen, multimodalen Modells erfolgen.

3.5 Modelling des multilingualen Modells

In der Modelling Phase wird die Zusammenstellung des Modells erläutert. Die Architektur fußt auf derselben Struktur wie die des CLIP Modells. Ziel ist es, dass beide Encoder wie oben beschrieben eine Embedding erstellen, aus der die Zugehörigkeit von Bild und Text übertragen werden kann. Die übergreifende geplante Architektur des Modells wird in Abbildung 1 in Abschnitt 1.2 dargestellt.

Text Encoder

Der monolinguale Text Encoder von CLIP wird hier durch eine Variante des XLM-RoBERTa Modells (Conneau et al., 2020) ausgetauscht. Das originale XLM-RoBERTa Modell wurde in den Grundlagen schon vorgestellt. Das spezielle Modell, das hier verwendet wird, ist das XLM-RoBERTa-base Modell (Conneau et al., 2020). Dieses Sprachmodell verwendet Byte-Pair-Encoding (Sennrich et al., 2016) mit einer Vokabulargröße von 50.000 Token und besteht aus insgesamt 12 kodierten Schichten. Das Modell besitzt 8 Attention Heads mit insgesamt 125 Millionen Parametern. Das vortrainierte XLM-RoBERTa-base Modell ist das Ergebnis des Trainings auf einem speziellen Datensatz, der aus dem 2,5 Terabyte großen CommonCrawl Datensatz zusammengesetzt wurde, welcher aus 100 unterschiedlichen Sprachen zusammengesetzt ist (Conneau et al., 2020). Die Implementation des Modells ist über die Python Transformers Bibliothek möglich (Debut & Bourdois, 2022).

Um den Input aus `caption_title_and_reference_description` für das Modell verarbeitbar zu machen, muss der Klartext durch einen Tokenizer umgewandelt werden. Hierfür wird aus dem Repertoire von Hugging Face zunächst die Autotokenizer Klasse verwendet (Hugging Face, o.D.). Diese Klasse ist eine generische Tokenizer Klasse, die als eine der Tokenizer aus der Hugging Face Bibliothek mit der `AutoTokenizer.from_pretrained()` Methode

instanziert wird. Der Tokenizer, der instanziert werden soll, wird durch den *model_type* des Konfigurationsobjekts ausgewählt. Dieses wird entweder per Argument übergeben, oder direkt durch den *pretrained_model_name_or_path* geladen. Im Fall dieser Arbeit wird der xlm-roberta-base Tokenizer übergeben. Dieser Tokenizer bereitet den Input für den Text Encoder durch Maskierung und Tokenisierung vor. So gibt es für den Anfang der Sequenz einen *bos_token* “*<|s|*” String-Token, der auch beim Pre-Training des Modells verwendet wurde. Für das Ende der Sequenz wird der *eos_token* Token “*</s>*” gesetzt. Ein separierender Token *sep_token* der gleichen Struktur “*</s>*” wird eingefügt, wenn eine Sequenz aus mehreren anderen Sequenzen gebildet wird. Unabhängig von der Tokenisierung wird zusätzlich die Länge der Sequenzen auf die von CLIP festgelegte Maximallänge von 77 Zeichen begrenzt. Diese Arbeit stützt sich durch die Implementierung des Text Encoders mit Hilfe des XLM-RoBERTa-base Modells auf die Hugging Face Bibliothek. Aus diesem Grund wird diese in einer kurzen Übersicht vorgestellt.

Hugging Face

Zentraler Inhalt der Hugging Face Bibliothek sind die sorgfältig geprüften Implementierungen von Varianten der Transformer-Architektur, die im NLP Feld weit verbreitet sind. Das Framework ist dazu konzipiert, dass es möglichst standardisierte Pipelines für NLP Aufgaben bereitstellt. Damit trägt Hugging Face dazu bei, Modelle direkt anwendbar zu machen. Jedes Modell in der Bibliothek wird durch drei spezielle Module zusammengesetzt. Das erste ist einem Tokenizer, welcher wie schon oben beschrieben den Klartext in Indexkodierungen umwandelt, zweitens aus einem Transformer, welcher diese Kodierungen in kontextuelle Embeddings transformiert und drittens ein Head, der kontextuelle Embeddings verwendet, um Aufgabenspezifische Voraussagen zu machen.

Image Encoder

Für den Image Encoder wird, wie in den Grundlagen beschrieben, eines der ursprünglichen Modelle beibehalten. Die Auswahl aus den 5 ResNet Modell, die Radford und Kollegen (2021) in ihrer Arbeit für 32 Epochen trainiert haben, fiel auf das RN50x4 Modell. Das Modell wurde ausgesucht, weil es am ressourcenschonendsten ist und trotzdem eine gute Performanz erreicht.

Die ResNet Modelle haben viele Varianten, die alle auf dem gleichen Konzept basieren, allerdings aus unterschiedlich vielen Schichten zusammengesetzt sind. ResNet50 wird als Begriff verwendet für das Modell, das aus einem neuronalen Netz mit 50 Schichten besteht. Die Art des speziellen neuronalen Netzes wird als Residual Neural Network bezeichnet (He et al., 2016). Der grundsätzliche Aufbau des Residual Network ist der eines sehr tiefen

Feedforward Neural Networks (Bebis & Georgopoulos, 1994). Die Besonderheit des Residual Networks ist, dass es zwischen den Schichten untereinander Abkürzungen gibt. Typisch für das ResNet Modell ist, dass zwischen jedem 3x3 Filter eine Abkürzung zwischen den Schichten eingefügt ist (He et al., 2016).

Radford und Kollegen (2021) haben sich für dieses Modell als Basis ihres Image Encoders festgelegt, da es umfangreiche Adaption gibt und die Performanz des Modells sehr gut ist.

Projection Head

Damit der Output der Encoder die gleiche Dimension für die spätere Berechnung aufweist, wird den Encodern der Projection Head angegliedert. Dessen Zweck ist es, die Bild und Textembeddings in die gleiche Dimension zu bringen. Nach der Kodierung weist der Bildvektor die Größe von 640 und der Textvektor die Größe von 768 auf. Ähnlich der Umsetzung von Salama (2021) mit TensorFlow (Abadi et al., 2016) transformiert das Projection Modul die Embeddings auf die Dimension von 512 und macht diese vergleichbar.

3.5.1 Training

Mit diesen Ergebnissen des Projection Head ist es möglich, das Modell zu trainieren. Hierfür müssen zunächst die Hyperparameter gesetzt werden. Im Gegensatz zu den festen Parametern des Modells sind Hyperparameter verstellbar und mit diesen Parametern als Variablen ist es möglich zu regulieren, wie der Lernprozess und die Performanz des Modells beeinflusst werden. Zu diesen Hyperparametern gehören zum Beispiel die Learning Rate, die Batch Size, die Temperatur und die Anzahl an Epochen.

Die Batch Size bestimmt die Anzahl an Proben aus dem Datensatz, die in einem Durchlauf durch das Modell geleitet werden. Die Anzahl an Epochen hingegen beschreibt, wie oft der gesamte Trainingsdatensatz während des Trainings durchlaufen wird. Eine Epoche kann also eine oder mehrere Batches beinhalten.

Temperature

Die Temperatur ist ein Hyperparameter, der dafür sorgt, dass die Zufälligkeit der Vorhersagen durch die Skalierung der Logits gesteuert wird, bevor die Softmax-Funktion angewandt wird. Hinton und Kollegen (2015) weisen darauf hin, dass neuronale Netze typischerweise Klassenwahrscheinlichkeiten durch die Nutzung einer Softmax-Output-Schicht erzeugen. Diese konvertiert die Logit (in der Funktion als z_i

gekennzeichnet), berechnet diese für jede Klasse in die Wahrscheinlichkeit q_i , indem z_i mit den anderen Logits verglichen wird.

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

T steht für den Temperaturwert, der normalerweise auf den Wert 1 festgelegt wird. Die Verwendung eines höheren Wertes für T führt zu einer weicheren Wahrscheinlichkeitsverteilung über die Klassen hinweg (Hinton et al., 2015). Wie groß oder klein die Temperatur gewählt wird, hat also Auswirkungen auf das Ergebnis des Modells. Aus einem größeren Temperaturparameter ergibt sich ein mehr “zuversichtliches” Modell (Hinton et al., 2015). Im Fall des originalen CLIP-Modells wurde die Temperatur von Radford und Kollegen (2021) auf 0,07 gesetzt: “The learnable temperature parameter was initialized to the equivalent of 0.07 [...] and clipped to prevent scaling the logits by more than 100 which we found necessary to prevent training instability” (S. 5) und wurde auch in dieser Arbeit beibehalten.

Learning Rate

Die Lernrate wird als Einstellung für einen Optimierungsalgorithmus verwendet, in dem die Schrittgröße bzw. die Anpassung der Parameter im Modell in jeder Epoche für die Erreichung des Minimums in der Verlustfunktion (Loss Function) kontrolliert wird (Kingma & Ba, 2017). Diese Lernrate wird für den visuellen und textuellen Teil des Modells unterschiedlich gewählt. Je geringer der Wert gewählt Wert ist, desto langsamer wird das Modell trainiert. Die Bedeutung der Lernrate ist laut (Smith, 2017) in Bezug auf das Training besonders wichtig. Für die Implementation müssen also im nächsten Kapitel die Hyperparameter der Learning Rate, der Batch Size und der Anzahl an Epochen festgelegt werden.

Contrastive Loss

Eine Verlustfunktion dient im Ablauf des Trainings für die iterative Verbesserung der Ergebnisse. Im Allgemeinen ergibt sich aus der Funktion, wie sehr die Vorhersage des Modells von den tatsächlichen Werten abweicht. Daraufhin werden die Parameter im nächsten Schritt optimiert.

Contrastive Loss (Hadsell et al., 2006) ist eine solche Funktion. Bei dieser werden paarweise Inputs aus den Embeddings übergeben und die Distanz zwischen den

Wertepaaren kalkuliert. Falls die Paare Ähnlichkeit besitzen wird ihr Wert erhöht, andernfalls wird er niedriger (Hadsell et al., 2006).

Adam Optimizer

Der Adam Optimierungsalgorithmus wurde von Kingma und Ba (2017) entworfen. Er ist eine Erweiterung des Gradient Descent und kann anstatt diesem verwendet werden, um Gewichte von neuronalen Netzen in den iterativen Durchläufen anzupassen. Die Autoren stellen heraus, dass der Adam Optimizer zwei wichtige Vorteile des Gradient Descent vereint. Einerseits den Adaptive Gradient Algorithm (AdaGrad) (Duchi et al., 2011), der die Performanz mit Problemen mit geringen Gradienten erhöht und dem Root Mean Square Propagation (RMSProp; Tieleman & Hinton, 2012), der gut in online und nicht stationären Situationen funktioniert. Ein weiterer Vorteil ist, dass durch die Implementierung des Hyperparameters der Learning Rate, die auch Schrittgröße (Step Size) genannt wird, die Adaptierung der Skalierung der Parameter ermöglicht wird. Der Konfigurationsparameter der Learning Rate des Adam Optimizers wird, wie in der Implementation zu sehen ist, mit der Variablen α bezeichnet.

PyTorch/PyTorch Lightning

Trotz der Übernahme des Projection Head aus dem Beispiel von Salama (2021) wird in dieser Arbeit die Umsetzung des Modells nicht mit TensorFlow, sondern mit PyTorch durchgeführt. Im Speziellen durch die PyTorch Lightning Bibliothek. Pytorch ist eine optimierte Bibliothek für Tensoren-Berechnung im Umfeld von Deep Learning Implementationen, die GPUs und CPUs verwenden (Paszke et al., 2019). Sie basiert auf der Open Source Torch Bibliothek und ist ebenfalls unter der BSD Lizenz ein Open Source Framework (Paszke et al., 2019). PyTorch ist nach vier Grundprinzipien zusammengestellt worden.

- “**Be Pythonic**” (Paszke et al., 2019, S. 3) Durch die weite Verbreitung der Programmiersprache Python im Umfeld der Data Science wurde vom PyTorch-Team der Entschluss gefasst, dass PyTorch Teil dieses Ökosystems sein soll. Bei der Umsetzung folgt das Framework den allgemein anerkannten Designzielen, die Schnittstellen einfach und konsistent zu halten. Zusätzlich lassen sich Standardaufgaben wie Datenverarbeitung, Debugging und das Erstellen von Diagrammen leicht integrieren (Paszke et al., 2019).
- “**Put researchers first**” (Paszke et al., 2019, S. 3) Das Ziel von PyTorch ist es, die Erstellung von Modellen, Data Loadern und Optimieren innerhalb des Deep Learning so einfach und produktiv wie möglich zu machen. Deswegen wird ein Großteil der

Komplexität dieser Erstellungen durch PyTorch gehandhabt, die durch APIs angesteuert werden kann.

- “**Provide pragmatic performance**” (Paszke et al., 2019, S. 3) Um einen guten Kompromiss zwischen Benutzerfreundlichkeit und überzeugender Leistung anzubieten, wird einerseits bei der Implementierung zusätzliche Komplexität in Kauf genommen. Andererseits wird durch die Bereitstellung von Werkzeugen, die es ermöglichen, die Ausführung des Codes manuell zu kontrollieren, die Benutzerfreundlichkeit erhöht. Außerdem ist es möglich, eigene Leistungsverbesserungen zu implementieren, unabhängig von denen, die die Bibliothek automatisch liefert (Paszke et al., 2019).
- “**Worse is better**” (Paszke et al., 2019, S. 3) Als letzter Punkt wird von den Erstellern von PyTorch hervorgehoben, dass mit einer Umsetzung durch PyTorch Zeit gespart werden kann und somit am schnellen Fortschritt im Bereich des Deep Learning mitgehalten werden kann. Sie argumentieren, dass es besser ist, eine einfache, aber etwas unvollständige Lösung zu haben als ein umfassendes, aber komplexes und schwer zu pflegendes Design (Paszke et al., 2019).

PyTorch Lightning ist ein Framework, das auf PyTorch aufbaut und somit diese vier Prinzipien übernimmt. Es ist ein leichtes, aber sehr performatives Framework, das den PyTorch Code anders organisiert und den Fokus auf das Engineering legt, um so das Erstellen und Trainieren von Modellen leichter umsetzbar und reproduzierbar zu machen. Man kann PyTorch Lightning als vereinfachte Strukturierung von PyTorch beschreiben (Lawaniya, 2020).

Das Framework wurde für die kommende Implementation ausgewählt, weil sich die Erstellung des Modells, des Data Loaders und der Verwendung des Optimizers effektiver und schneller umsetzen lässt als mit dem PyTorch Framework. Besonders von Vorteil ist die Möglichkeit, mit Unterbrechungen zu trainieren, da der sehr rechenintensive Wettbewerb so in mehreren Etappen bearbeitet werden kann.

3.6 Optimierungsstrategien

Zur Verbesserung der Benchmark und der Optimierung des Ergebnisses des erstellten Modells sollen im Folgenden zwei Optimierungsstrategien vorgestellt werden. Die erste Strategie befasst sich mit dem clip-ViT-B-32-multilingual-v1 Sentence Transformer Modell von Reimers (o.D.). Die zweite Strategie setzt sich mit der Untersuchung der Auswirkungen des Prompt Engineering auf die Modelle auseinander.

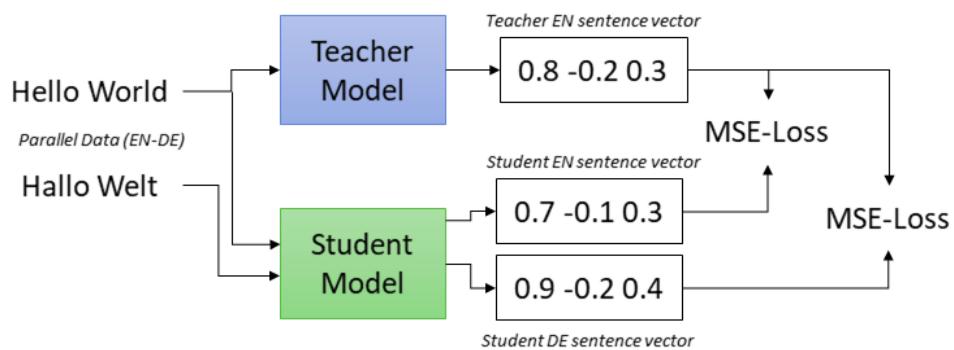
3.6.1 Sentence Transformer Modell

Das Modell wurde 2021 durch die Vorarbeit aus der Arbeit von Reimers und Gurevych (2020) erstellt. Der vordergründige Nutzen ist die multilinguale Zero Shot Bildklassifikation. Durch eine Methode, die sie Multilingual Knowledge Distillation (Reimers & Gurevych, 2020) nennen, haben sie bestehende Embeddings auf andere Sprachen erweitert.

Dazu haben Reimers und Gurevych (2020) wie in Abbildung 11 zu sehen eine Teacher-Student-Architektur entworfen, bei der das Teacher Modell auf einer Anker-Sprache trainiert ist und das Student Modell durch verschiedene Übersetzungen der Anker-Sprache und Mean Squared Loss auf Multilingualität trainiert wird. So lernt das Student Modell multilinguale Satz-Embedding mit zwei wichtigen Eigenschaften. Erstens sind die Vektorräume über alle Sprachen hinweg gleich groß und zweitens werden die Eigenschaften des Vektorraums des Teacher Modells zu allen anderen Sprachen übernommen (Reimers & Gurevych, 2020). Als Teacher Modell wurde das originale clip-ViT-B-32 (Radford et al., 2021) verwendet und als Student Modell wurde DistilBERT (Sanh et al., 2020) herangezogen. Das Ergebnis ist ein Modell, das auf bis zu 50+ Sprachen gute Performanz erreicht. Der Image Encoder von CLIP wurde unverändert übernommen.

Abbildung 11

Darstellung der Teacher-Student Architektur (Reimers & Gurevych, 2020, S. 2)



Anmerkung. Aus "Making monolingual sentence embeddings multilingual using knowledge distillation" von N. Reimers und I. Gurevych, 2021, arXiv, S. 2.

In dieser Optimierungsstrategie soll überprüft werden, ob die Herangehensweise des Multilingual Knowledge Distillation mithilfe einer Student-Teacher Architektur das Ergebnis des in dieser Arbeit erstellten multilingualen Modells übertrifft und somit als Optimierung in

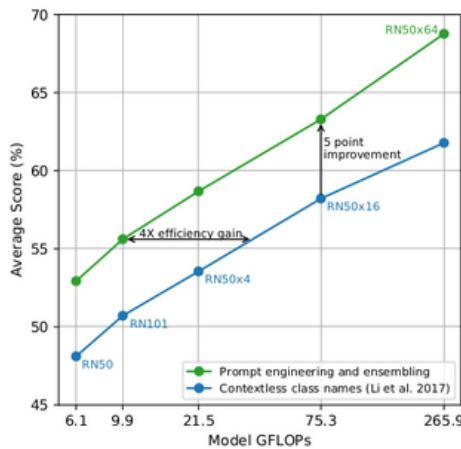
Betracht gezogen werden sollte. Dazu wird ähnlich wie in der Benchmark Erzielung per Zero Shot die Performanz des Modells getestet und den anderen Ergebnissen gegenübergestellt.

3.6.2 Multilinguales Prompt Engineering

Die zweite Strategie zur Optimierung ist das Prompt Engineering. Dieser Vorgang beschreibt das Vorgehen, eine Eingabeaufforderung (Prompt) der Input-Embedding hinzuzufügen, um so die nachgelagerte Aufgabe effektiver zu bearbeiten (Radford et al., 2021).

Abbildung 12

Darstellung der Auswirkungen von Prompt Engineering (Radford et al., 2021, S. 7)



Anmerkung. Aus “Learning transferable visual models from natural language supervision.” von A. Radford et al., 2021, Proceedings of Machine Learning Research, 139, S. 7.

Wie in Abbildung 12 zu sehen, konnten Radford und Kollegen (2021) durch Prompt Engineering die Performanz von CLIP verbessern. Im Vergleich zu der Baseline bei der Verwendung von kontextlosen Klassennamen konnte durch Prompt Engineering und durch die Einfügung von Boost Zero Shot die Klassifikationsperformanz um fast durchschnittlich fünf Punkte über 36 Datensätze erhöht werden (Radford et al., 2021).

Das Problem, das bei der Erstellung von CLIP Prompt Engineering offensichtlich wurde, ist, dass Standard-Datensätze für die Bildklassifikation die Namen oder Beschreibungen der Bildklassen als nachträgliche Priorität einstufen. Die große Mehrheit der Datensätze annotiert Bilder laut den CLIP Autoren nur mit einer numerischen ID des Labels und enthält eine Datei, die diese IDs auf ihre englischen Namen zurückführt (Radford et al., 2021). Einige Datensätze scheinen diese Zuordnung in ihren veröffentlichten Versionen nicht zu

enthalten, wodurch Zero Shot Transfer vollständig verhindert wird. Bei vielen Datensätzen haben Radford und Kollegen (2021) festgestellt, dass diese Bezeichnungen eher willkürlich gewählt werden (Radford et al., 2021). Deswegen kann der CLIP Text Encoder, wenn der Name einer Klasse die einzige Information ist, die ihm zur Verfügung gestellt wird, aufgrund des fehlenden Kontextes nicht unterscheiden, welcher Wortsinn gemeint ist. Radford und Kollegen (2021) beobachteten, dass in einigen Fällen mehrere Bedeutungen desselben Wortes als unterschiedliche Klassen in ein und demselben Datensatz enthalten sein können. So zum Beispiel in dem schon erwähnten ImageNet Datensatz, in dem sowohl Baukräne als auch Vögel der Art Kräne vorhanden sind.

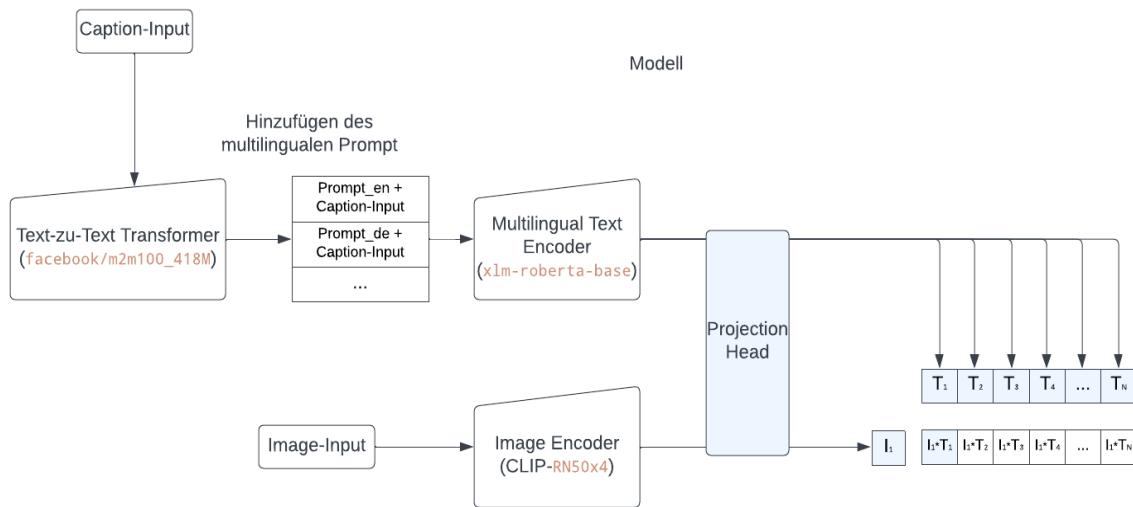
Ein weiteres Problem, das erkannt wurde, ist, dass selten ein Bild mit einem Text gepaart ist, das aus einem einzigen Wort besteht. In den meisten Fällen besteht der Text aus einem ganzen Satz, der das Bild beschreibt. Um dieses Problem zu umgehen, haben Radford und Kollegen (2021) in ihrer Arbeit festgestellt, dass die Eingabeaufforderung "*A photo of a {label}*" dabei hilft, die semantische Beziehung zwischen dem Text und dem Bild zu verstärken. Dies soll häufig die Leistung im Vergleich zur bloßen Standardklassenbeschriftung erhöhen. So haben Radford und Kollegen (2021) zum Beispiel dadurch die Genauigkeit im ImageNet-Test um 1,3 % gesteigert. Sie gehen davon aus, dass dadurch auch bei anderen spezifischen Aufgaben die Performanz erhöht werden kann (Radford et al., 2021). In diesem Sinn haben sie herausgefunden, dass bei einigen Bildklassifizierungsdatensätzen es hilfreich ist, die Kategorie zu spezifizieren. Bei der Bearbeitung des Oxford-IIIT Pets Datensatzes (Parkhi et al., 2012) beispielsweise funktionierte der Prompt "*A photo of a {label}, a type of pet*" gut, um den Kontext zu vermitteln. Bei dem OCR Datensatz haben Radford et al. (2021) herausgefunden, dass Anführungszeichen, die den Prompt einschließen, helfen, die Performanz zu erhöhen. Eine weitere Eigenschaft, die herausgestellt werden konnte, ist, dass es bei Daten zur Klassifizierung von Satellitenbildern hilfreich ist, den Prompt in der Form "*a satellite photo of a {label}*." zu formulieren.

Die Auswirkungen der Anwendung von Prompt Engineering sind jedoch keinesfalls immer nachvollziehbar. So ist es kein Garant, dass das Hinzufügen von Prompts die Performanz des Modells positiv beeinflusst: "Finally, we've observed that CLIP's zero-shot classifiers can be sensitive to wording or phrasing and sometimes require trial and error "prompt engineering" to perform well" (OpenAI, 2021, Abs. 18).

Um diese Optimierungsstrategie zu testen, wird vor dem Text Encoder des eigentlichen Modells ein Zwischenschritt eingefügt. In Abbildung 13 wird der Prozess des multilingualen Prompt Engineering dargestellt.

Abbildung 13

Eigene Darstellung basierend auf (Radford et al., 2021, S.2)



Anmerkung. Aus "Learning transferable visual models from natural language supervision." von A. Radford et al., 2021, Proceedings of Machine Learning Research, 139, S. 2.

Der Caption Input wird durch das multilinguale Übersetzungsmodell "facebook/m2m100_418M" (Fan et al., 2021) um das jeweilig übersetzte Prompt erweitert. Das auf Multilingual Machine Translation (MMT) (Fan et al., 2021) spezialisierte Modell M2M100 ist ein multilingualer Encoder-Decoder, vorgenommen für Many-to-Many multilinguale Übersetzung. Das Modell wird dazu verwendet, die beiden Prompts "[a photo of](#)" und "[From Wikipedia:](#)" in die jeweiligen Sprachen zu übersetzen. Der zusammengesetzte Input wird daraufhin an den Text Encoder weitergegeben.

Um diese Strategie übergreifend vergleichbar zu machen, wird auch die Veränderung der Performanz von Zero Shot CLIP und die Performanz des Sentence Transformer Modells von Reimers (o.D.) mit dem multilingualen Prompt Engineering getestet.

3.7 Evaluation

Als letzter Schritt, der aus dem CRISP-DM Modell übernommen wird (Wirth & Hipp, 2000), wird hier die Evaluation beschrieben, mit der die Modelle getestet werden. Die Evaluation des Wettbewerbs unterscheidet sich von der Implementation der Evaluation, die in dieser Arbeit ausgewählt wurde.

Die Auswahl des Wikimedia Teams für die Evaluation der Einreichungen fiel auf den NDCG@5 (Normalized Discounted Cumulative Gain).

Wie schon im Abschnitt 2 beschrieben, soll die Einreichung eine Liste bestehend aus Bild-ID und *caption_title_and_reference_description* sein, die nach Wahrscheinlichkeit von oben nach unten absteigend sortiert geordnet ist. Dabei können bis zu fünf Captions per Bild-ID angegeben werden.

Der NDCG ist eine Erweiterung des klassischen Discounted Cumulative Gain (DCG). Das Ziel des DCG ist es, relevante Elemente, die in dem zurückgegebenen Ergebnis unter nicht relevanten Ergebnissen angeordnet sind, zu bestrafen. Der Wert des relevanten Objekts wird logarithmisch proportional zur Position des Ergebnisses reduziert. Der DCG an einer bestimmten Rangposition p wird wie folgt definiert:

$$DCG_p = \sum_{i=1}^p \frac{rel_i}{\log_2(i+1)}.$$

Hier steht rel_i für die bestimmte Relevanz des Elementes an der Position i in den zurückgegebenen Ergebnissen. Der NDCG normalisiert den DCG auf den maximalen theoretischen Wert und liefert somit Werte im Bereich $[0, 1]$.

Im Gegensatz dazu wird für die schnellere Vergleichbarkeit und Übersichtlichkeit in dieser Arbeit inspiriert durch die Evaluation von Salama (2021) durch Top k Accuracy evaluiert. Diese Metrik, ähnlich zum Recall@K, berechnet die Anzahl der Fälle, in denen das richtige Label unter den k zurückgegebenen Elementen ist. Das Ergebnis des Top k Accuracy wird also für jedes Element als korrekt angesehen, solange es in den k zurückgegebenen Ergebnissen liegt. Für die Implementation der Evaluation im nächsten Abschnitt werden die Caption als Abfragen benutzt. Es werden aus dem vorher aufgeteilten Testdatensatz Bilder und Captions ausgewählt, um die Retrieval Qualität der Modelle durch den Top k Accuracy zu vergleichen. Eine korrekte Zuweisung für ein Bild wird gezählt, solange die entsprechenden Captions unter den Top $k = 20$ Treffern sind.

4. Implementation

In diesem Abschnitt wird die Planung und Konzeption aus dem letzten Abschnitt übernommen und die praktische Umsetzung dargestellt. Hierfür werden im Folgenden zunächst die übergeordneten Rahmenbedingungen wie Hardware und

Entwicklungsumgebung dargestellt. Im Laufe dieses Kapitels wird entsprechend der Konzeption die Implementation der Data Understanding, der Data Preparation, die Umsetzung des Modelling und der Optimierungsstrategien erläutert.

Beim Data Understanding liegt der Fokus neben der Visualisierung des Datensatzes, um dessen Eigenschaften herauszustellen, auch auf den Zusammenhängen innerhalb des Datensatzes. Deshalb wird die Herangehensweise der praktischen Untersuchung der Daten genau beschrieben. Die Data Preparation steht in enger Beziehung zum Data Understanding. Hier wird der Datensatz bereinigt und bearbeitet gemäß den Schlüssen, die aus der Untersuchung gezogen werden. Im nächsten Schritt wird die Umsetzung des Zero Shot CLIP auf dem bereinigten Datensatz erläutert. Im Abschnitt Modelling wird die Erstellung des multimodalen, multilingualen Modells mit Hilfe von PyTorch Lightning beschrieben und das anschließende Training auf dem Datensatz umgesetzt. Im letzten Abschnitt wird die Umsetzung der zwei Optimierungsstrategien genauer beleuchtet.

Die Implementation der Arbeit wurde durch die Nutzung von verschiedenen Entwicklungsumgebungen möglich. Ein Großteil wurde durch Google Colaboratory Notebooks umgesetzt. Colaboratory oder auch in der bekannteren Kurzform Colab wurde von Google Research entwickelt. Es ermöglicht das Schreiben und Ausführen von beliebigem Python Code und ist besonders gut geeignet für Machine Learning und Data Analysis Anwendungen (Google, o.D.). Colab wird auf dem Jupyter Notebook Service gehostet und stellt die Nutzung von Rechenressourcen wie CPUs und GPUs bereit.

Eine weitere Entwicklungsumgebung, die für die Erstellung der Arbeit genutzt wurde, sind die Kaggle eigenen Notebooks. Ähnlich zu Colab basieren diese ebenfalls auf der Jupyter Notebook Anwendung. In diesen wurde vor allem das Preprocessing des Datensatzes ausgelagert, aber auch teilweise das Training des Modells. Die gesamte Arbeit wurde durch die Programmiersprache Python umgesetzt.

Die Hardware der verschiedenen Notebooks unterscheidet sich leicht. Im Fall der Colab Umgebung wurden während der Implementation Nvidia Tesla T4 GPUs verwendet und im Fall der Kaggle Notebooks Nvidia K80 GPUs. Die Implementation wurde in mehreren Notebooks umgesetzt, die im digitalen Anhang dieser Arbeit beigefügt sind. Neben den genannten Bibliotheken in der Konzeption wurden für die Verwirklichung der Arbeit einige weitere benötigt.

Urllib

Urllib ist ein Pythonbibliothek, das es ermöglicht URLs mit Python abzufragen. Dafür wird die *urlopen* Funktion verwendet, wodurch es möglich ist, URLs mit diversen unterschiedlichen Protokollen abzurufen. Für den Umgang mit den Bild-URLs im Datensatz ist dies unumgänglich. Bei der Handhabung der URLs verwendet das Urllib Paket verschiedene Module, die auch in der Arbeit genutzt werden, wie zum Beispiel `urllib.request` für das Öffnen und Lesen von Seiten oder auch `urllib.error` für die Handhabung mit Fehlern und das Melden von Exceptions.

Pillow

Pillow ist ein PIL Fork von Clark (2015). PIL ist die ursprüngliche Python Bibliothek von Umesh und Kollegen (2012), die Funktionen für die Bildbearbeitung dem Python Interpreter hinzufügt. Dabei bietet die Bibliothek eine umfangreiche Unterstützung von unterschiedlichen Dateiformaten, eine effiziente interne Darstellung und daraus resultierend leistungsfähige Bildverarbeitungsfunktionen. Die Basis der Bibliothek ist für einen schnellen Zugriff auf Daten ausgelegt, die in Pixelformaten gespeichert werden (Clark, 2015).

Dask

Für das Preprocessing der Daten wurde aufgrund der Größe die von Rocklin (2015) erstellte Dask Bibliothek herangezogen. Dies ist eine flexible Open Source Python Bibliothek für paralleles Computing.

Dask setzt sich im Grund aus zwei Komponenten zusammen:

1. Die Erste ist die dynamische Aufgabenplanung (Task Schedulers), die für die Optimierung von Berechnungen erstellt wurde. Die Autoren von Dask ziehen hier Vergleiche zu Airflow, Luigi, Celery oder Make (Dask, o.D.). Jedoch ist die dynamische Aufgabenplanung bei Dask optimiert für interaktive Berechnungsaufgaben.
2. Die Zweite sind “Big Data”-Sammlungen wie parallele Arrays, Data Frames und Listen, die Schnittstellen wie Numpy oder Pandas durch Umgebungen mit mehr als einem Arbeitsspeicher oder für verteilte Umgebungen erweitern. Diese parallelen Sammlungen arbeiten basierend auf dynamischen Aufgabenplanern.

Die Bearbeitung von Datensätzen wird durch Dask von multi-core lokalen Maschinen auf große, verteilte Systeme in der Cloud umgelenkt.

Dabei wird ein ähnliches Interface bereitgestellt wie bei anderen Bibliotheken, zum Beispiel Pandas, Scikit-learn oder auch NumPy. Vor allem die Nutzung von dask.dataframe hat für das Preprocessing effiziente Ergebnisse erzielt. Hierbei handelt es sich um ein großes Parallel Data Frame, das aus vielen kleinen Pandas Data Frames, die entlang des Index aufgeteilt werden, zusammengestellt ist. Durch die Nutzung von Dask Data Frame konnte die Bearbeitung der Daten während des Preprocessing enorm beschleunigt werden.

Scikit-learn

Eine weitere Open Source Python Bibliothek, spezialisiert auf Machine Learning, ist Scikit-learn. Neben den umfassenden integrierten Werkzeugen für Machine Learning wie Klassifikations-, Regressions- und Clusteringalgorithmen, beinhaltet diese Bibliothek auch Werkzeuge für das automatisierte Aufteilen eines Datensatzes in Trainings- und Testdatensatz. Hiermit wurde der Datensatz dieser Arbeit während der Data Preparation aufgeteilt, sodass auf diesem später trainiert und evaluiert werden konnte (Pedregosa et al., 2011).

Hyperparameter Tuning

Wie im letzten Kapitel beschrieben, sind die Hyperparameter einer der wichtigsten Faktoren, wenn es darum geht, ein Modell effektiv und reproduzierbar zu trainieren. Deswegen sollen hier, bevor es mit der Darstellung des praktischen Ablaufs des Data Understanding weitergeht, die für die Arbeit festgelegten Hyperparameter und andere wichtige globale Variablen dargestellt und erläutert werden.

Zunächst wurde der Seed auf den Wert 42 gesetzt, um so für die Reproduzierbarkeit zu garantieren, indem immer derselbe Trainings- und Validationsdatensatz erstellt wird. Die Batch Size wurde auf 64 festgelegt, da gezeigt wurde, dass ein zu hoher Wert kontraproduktiv für die Performanz des Modells ist: "It has been observed in practice that when using a larger batch there is a significant degradation in the quality of the model, as measured by its ability to generalize"(Keskar et al., 2017).

Aufgrund der Größe des Datensatzes und der beschränkten Verfügung an Rechenressourcen muss für die absehbare Dauer des Trainings ein Kompromiss gezogen werden. Trainingstests mit dem zu erstellenden Modell haben gezeigt, dass die Dauer von 5 Epochen optimal für die zur Verfügung stehenden Ressourcen ist. Das Modell wird so iterativ in Abschnitten von je 5 Epochen insgesamt bis zu 45 Epochen auf jeweils einer GPU trainiert.

Für die Lernrate gibt es zwei unterschiedliche Werte. Die Lernrate für den Image Encoder beträgt $1e-4$ und die Lernrate für den Text Encoder $1e-5$. Die Lernraten sind so niedrig angesetzt, da, wie Nakamura und Harada (2019) festgestellt haben, dass eine kleine Lernrate beim Retraining von Vorteil ist. “[...] a low learning rate stabilizes the retraining process,[...]"(Nakamura and Harada, 2019, S. 1).

Ebenfalls bietet sich hier das Setzen der übergreifenden Variable der Content Length für den Text Encoder an. Wie in den vorherigen Abschnitten gezeigt, muss die Länge auf 77 Zeichen begrenzt werden. Beim Training auf den Daten zeigte sich allerdings ein Problem mit der Länge des Inputs. Es stellte sich heraus, dass die Länge der im Datensatz enthaltenen Unicode Sequenzen noch einmal auf 60 Zeichen verringert werden musste.

Beim Festlegen der Hyperparameter für den Projection Head gab es drei maßgebliche Variablen. Zuerst wurde die Anzahl der Schichten des Heads auf drei festgelegt. Weiterhin, um Overfitting zu verhindern, wurde der Dropout auf 0.5 gesetzt. Als letzter Punkt wurde die Größe für die Vektoren, die der Projektion Head als Ergebnis ausgeben soll, auf 256 festgelegt.

4.1 Data Understanding

Die Umsetzung des Data Understanding beginnt mit der Untersuchung der Trainingsdaten. Im weiteren Verlauf folgt die eng mit der Data Understanding verknüpfte Data Preparation und danach die Data Visualization. Ziel ist es, sich in diesem Abschnitt mit den Trainingsdaten vertraut zu machen und einen Datensatz zu erstellen, der bereinigt und fertig für das Training des Modells ist.

In der Konzeption wurde die Frage aufgeworfen, in welcher Form die Quelle der Bilddateien ausgewählt wird. Entweder über die bereitgestellten base64 kodierten Dateien oder über die Bild-URL direkt aus dem Datensatz. Mehrere Tests zeigten, dass aufgrund der begrenzten Kapazität der Entwicklungsumgebungen der ressourcenintensive Vorgang über die base64 Kodierung nur sehr umständlich umsetzbar ist. Deswegen wurde der Weg über die im Datensatz vorhandenen Bild-URLs gewählt.

Der am Ende fertige Datensatz für das Training muss aus den Trainingsdaten **train-00000-of-00005.tsv**, **train-00001-of-00005.tsv**, **train-00002-of-00005.tsv**, **train-00003-of-00005.tsv** und **train-00004-of-00005.tsv** zusammengestellt werden, die alle rund 14 GB je Trainingsdatei umfassen.

In Abbildung 3 in Abschnitt 2.1.1 wird zur besseren Veranschaulichung der Head mit 7 der 18 Spalten der Trainingsdatei **train-00000-of-00005.tsv** dargestellt. Zu erkennen ist, dass in den *caption_title_and_reference_description* schon [SEP] Token vorhanden sind. Diese müssen während der Data Preparation in die XLM-RoBERTa (XLM-R) spezifischen Tokens umgewandelt werden. Eine weitere Aufgabe ist die Reduzierung der Dimension auf die zwei relevanten Spalten *image_url* und *caption_title_and_reference_description* und die allgemeine Bereinigung der Daten. Der letzte Schritt ist die Umformung der Daten in einen für das Training und die Evaluation gerechten Datensatz.

4.1.1 Data Preparation

Die Data Preparation beginnt mit dem Schritt der Reinigung der Daten. In der Preprocessing Phase wird der Datensatz das erste Mal genauer in Augenschein genommen. Die Umsetzung ist nicht ohne Weiteres durch Pandas in den Entwicklungsumgebungen möglich, da die Größe der Datensätze die Kernel-Memory übersteigt. Deswegen wird der erste Bearbeitungsschritt durch Dask ausgeführt. Wie im oberen Abschnitt zu Dask beschrieben, können damit Datensätze in aufgeteilten Partitionen bearbeitet und somit einfacher an den Speicher der Kernels angepasst werden. Da Dask ähnliche APIs anbietet wie Pandas, können mit ähnlichen Befehlen die gewünschte Datenbearbeitung erfolgen.

Zunächst wird die Reduzierung der Dimension des Datensatzes vorgenommen und es bleiben die relevanten Spalten der *image_url* und *caption_title_and_reference_description* zurück. Als nächster Schritt werden alle NULL bzw. NaN Werte aus dem Datensatz entfernt. Somit sind alle fehlenden Werte aus dem Datensatz getilgt. Das Ergebnis dieser Bearbeitung wird daraufhin in ein Pandas Data Frame umgewandelt und in einem Feather Format gespeichert. Das Benutzen von Feather beschleunigt signifikant I/O Operationen, das Lesen des Datensatzes und verbraucht weniger Speicherplatz.

Denn Feather ist ein kompaktes, binäres Dateiformat für Data Frames, das im Kern das Arrow IPC Format beinhaltet. Dies wurde durch das Apache Arrow Projekt ermöglicht, das als Proof of Concept schnelle Speicherung von Data Frames für Python zum Ziel hatte (Apache Arrow, o.D.). Als Nächstes werden die zwei Spalten des neuen Datensatzes untersucht. Bei der Bearbeitung der *image_urls* wird sich bei der Abfrage auf die in der Einleitung von Abschnitt 4 dargestellte Urllib Bibliothek gestützt. Um die *image_urls* zu bereinigen, wird der Datensatz durchgegangen und jedes einzelne Bild testweise abgerufen. Dabei gab es bei mehreren Bildern einen 404 Fehler und es war zum Stand der letzten Überprüfung nicht möglich, 146 Bilder abzurufen. Allerdings ist im Verlauf der Arbeit deutlich

geworden, dass es immer weitere Ausfälle von Bildern über die Zeit hinweg gibt. Der Stand der Bearbeitung bis zu diesem Punkt wird in der Darstellung des Heads des Datensatzes in folgender Abbildung 14 deutlich.

Abbildung 14

Darstellung des Headers der vorläufigen Trainingsdaten...

	image_url	caption_title_and_reference_description
0	https://upload.wikimedia.org/wikipedia/commons...	Giày cao gót [SEP] Giày cao gót châу Âu, khoả...
1	https://upload.wikimedia.org/wikipedia/commons...	Grand Prix automobile de France 1957 [SEP] Juan...
2	https://upload.wikimedia.org/wikipedia/commons...	道谷站 [SEP] 車站月台
3	https://upload.wikimedia.org/wikipedia/commons...	北仙台站 [SEP] 月台
4	https://upload.wikimedia.org/wikipedia/commons...	Silver spoon [SEP] Two silver-gilt strainer sp...
...

Die fehlenden Werte bzw. teilweise leere Zeilen sind entfernt und die image_urls sind bereinigt. Der Datensatz besteht jetzt nur noch aus den image_url und caption_title_and_reference_description Spalten. Bei der Bearbeitung der zweiten Spalte müssen die vorhandenen [SEP] Token durch die für den Text Encoder erforderlichen XLM-RoBERTa (XLM-R) Token <s> und </s> ersetzt werden. Hierfür wird wieder die komplette Spalte durchlaufen und jeder Token ausgetauscht. Das Ergebnis ist in Abbildung 15 zu sehen.

Abbildung 15

Darstellung der letztendlichen Trainingsdaten...

	image_url	caption_title_and_reference_description
0	https://upload.wikimedia.org/wikipedia/commons...	Giày cao gót </s> Giày cao gót châу Âu, khoảng...
1	https://upload.wikimedia.org/wikipedia/commons...	Grand Prix automobile de France 1957 </s> Juan...
2	https://upload.wikimedia.org/wikipedia/commons...	道谷站 </s> 車站月台
3	https://upload.wikimedia.org/wikipedia/commons...	北仙台站 </s> 月台
4	https://upload.wikimedia.org/wikipedia/commons...	Silver spoon </s> Two silver-gilt strainer spo...
...
11771	https://upload.wikimedia.org/wikipedia/commons...	Renault R23 </s> La Renault R23 de Jarno Trull...
11772	https://upload.wikimedia.org/wikipedia/commons...	Chanson des quatre fils Aymon </s> Vue des rui...
11773	https://upload.wikimedia.org/wikipedia/commons...	Дунъхунская карта </s> Китайский Дунъхунский...
11774	http://upload.wikimedia.org/wikipedia/commons/...	Géorgiques </s> Le jardin du vieillard corycie...
11775	http://upload.wikimedia.org/wikipedia/commons/...	Black Point </s> Black Point i Sydgeorgien och...
11776 rows × 2 columns		

Der präparierte und bereinigte Datensatz umfasst nun wie zu erkennen 11776 Elemente und zwei Spalten. Der letzte Schritt für die Bearbeitung des Datensatzes ist die Aufteilung in Trainingsset und Testset mit den Anteilen 0,8 und 0,2. Vor dem Training, das im Abschnitt 4.4 Multilinguales Modell beschrieben wird, wird durch den Pytorch Data Loader des PyTorch Lightning Moduls eine weitere Aufteilung vorgenommen.

Hier wird das Trainingsset noch einmal in ein weiteres Trainingsset und ein Evaluationsset aufgeteilt. Die Performanz der Modelle wird am Ende wie schon beschrieben durch die Top k Accuracy Retrieval Qualität auf dem Testset evaluiert. Zusätzlich wird vorher jedoch die Abspaltung eines kleinen Stichprobenevaluationsset aus den ersten Exemplaren der häufigsten fünf Sprachen vorgenommen. Auf diesen soll die Performanz der Modelle durch die Darstellung in einer Cosine Similarity Matrix visualisiert werden. Dieses Set wird am Ende der Data Visualization vorgestellt.

4.1.2 Data Visualization

Nach der Beschreibung der Data Preparation Umsetzung folgt nun die Visualisierung des Datensatzes zur besseren Veranschaulichung. Hier wird mit den Bibliotheken Matplotlib und Seaborn, die in der Konzeption beschrieben wurden, die Zusammensetzung des Datensatzes dargestellt, um diesen besser zu verstehen und eine geeignete Herangehensweise an das Training zu finden. Die Beziehung der nun zwei Attribute des Datensatzes und die Charakteristik der enthaltenen Daten wird im Folgenden in mehreren Diagrammen dargestellt. So soll vor allem Aufschluss über die verschiedenen Eigenschaften der für den Text Encoder wichtigen Captions gegeben werden. Als Erstes stellt sich die Frage nach der Verteilung der Sprachen innerhalb der Captions.

Abbildung 16
Darstellung der Sprachverteilung im Datensatz

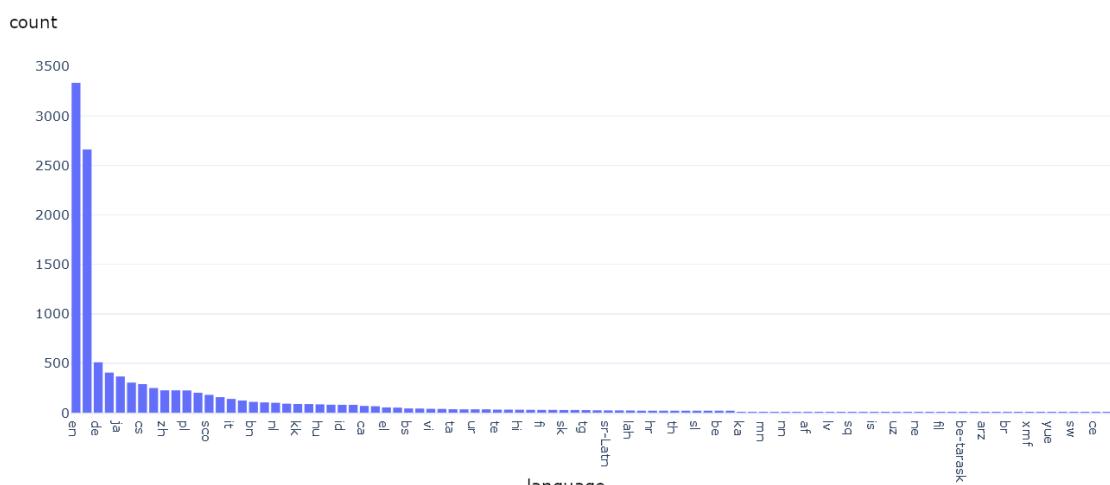
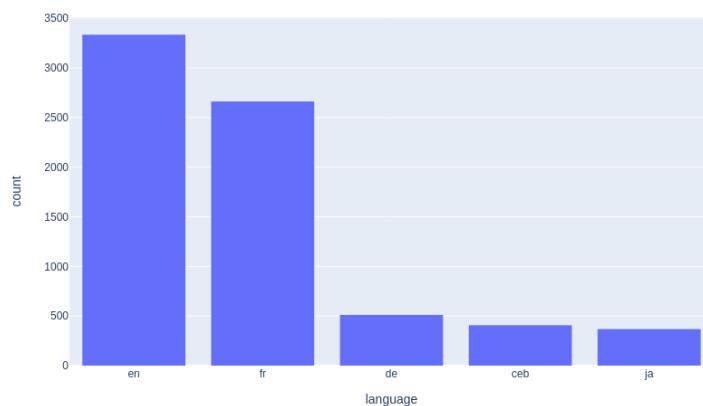


Abbildung 16 zeigt die Sprachverteilung der 94 unterschiedlichen Sprachen. Zu sehen ist, dass einige Sprachen deutlich mehr vertreten sind als andere. Diese unausgewogene, als Long Tail benannte Datenverteilung kann als Resultat nach sich ziehen, dass Modelle voreingenommen gegenüber den häufig vorhandenen Daten sind und eher schlechtere Leistung bei der Klassifikation der niedrigen Daten erreichen (Zhang, Kang, et al., 2021). Andersherum ist es schwierig, durch das Fehlen genügender Tail-Daten das Modell für diese zu trainieren.

In ihrer Arbeit benennen Zhang, Kang und Kollegen (2021) das Problem wie folgt: “In [...] applications, training samples typically exhibit a long-tailed class distribution, where a small portion of classes have massive sample points but the others are associated with only a few samples. Such class imbalance of training sample numbers, however, makes the training of deep network based recognition models very challenging” (Zhang, Kang, et al., 2021, S. 1). In ihrer Arbeit konzentrieren sie sich zwar vorsätzlich mit Bildklassifikation, jedoch ist dieses Problem der Long Tail Datenverteilung auch in den anderen Bereichen von Machine Learning vorhanden. Bei näherer Betrachtung der fünf häufigsten Sprachen wird die Verteilung noch deutlicher.

Abbildung 17
Darstellung der Verteilung der fünf häufigsten Sprachen



In Abbildung 17 ist zu sehen, dass vor allem die englisch- und französischsprachigen Captions dominieren. Um mit dieser Datenverteilung umzugehen, gibt es mehrere Herangehensweisen. Eine Möglichkeit wäre es, die Long Tail Daten abzuscheiden und sich auf die häufigsten Sprachen Englisch und Französisch zu konzentrieren. Dies würde eventuell die Performanz des Modells in diesem speziellen Fall erhöhen. Allerdings würde

dieses Ergebnis entgegen dem Ziel der Competition stehen, ein möglichst umfassendes multilinguales Modell zu erstellen. Eine weitere Möglichkeit wäre es, zusätzliche sprachlich unterrepräsentierte Captions dem Datensatz hinzuzufügen und so die Trainingsbreite zu erhöhen. Die naheliegendste Methode ist, auf das Pre-Training des Text Encoder-Modells zu vertrauen und die Daten zunächst so zu belassen. Weitere Möglichkeiten, dieses Problem zu lösen, könnten Gegenstand zukünftiger Forschungsbemühungen sein. Eine weitere Eigenschaft, die bei der Untersuchung der Captiondaten von Bedeutung ist, ist die Verteilung der Sequenzlänge. Da es eine Begrenzung von 77 Zeichen für den Text Encoder aufgrund der Vorgaben des CLIP Modells gibt, muss sich mit der Sequenzlänge der einzelnen Captions auseinandersetzen werden.

Abbildung 18

Darstellung der Verteilung der Sequenzlänge

The figure is a scatter plot titled "caption_title_and_reference_description". The Y-axis is labeled "Sequence Length" and ranges from 0 to 800 with major ticks at 0, 200, 400, 600, and 800. The X-axis is labeled "Sample Nummer" and ranges from 0 to 12000 with major ticks every 2000 units. The data points are represented by small blue dots. A large majority of points are clustered tightly around a sequence length of approximately 150. There are several distinct vertical bands of points at higher sequence lengths, notably around 400, 600, and 800. The highest point visible is near a sample number of 9000 and a sequence length of about 900.

Über den gesamten Datensatz wurde für jede *caption_title_and_reference_description* die Sequenzlänge ermittelt und das Ergebnis in Abbildung 18 visualisiert. Wie im Diagramm zu erkennen ist, gibt es viele Captions mit einer Länge, die größer als 77 Zeichen beträgt. Ebenfalls gibt es Captions, die im Datensatz im Unicode vorhanden sind, welche auf 60 Zeichen begrenzt werden mussten.

Durch die Begrenzung der Captions auf eine geringere Länge geht einige Information, die für die Zuordnung im Modell hilfreich sein könnte, verloren. Dies könnte ähnlich wie die ungleiche Verteilung der Sprachen Auswirkung haben auf die Performanz des Modells. Dies ist ein Punkt, der im Ausblick der Zusammenfassung noch einmal aufgegriffen wird.

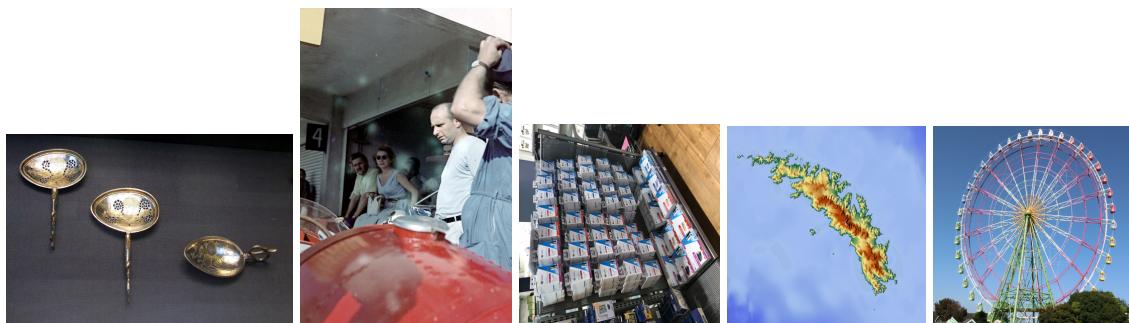
63

Neben der näheren Auseinandersetzung mit den Captions gehört zu der Visualisierung, um den Datensatz in Gänze zu verstehen, auch der Bildanteil dazu.

Deswegen wird in Abbildung 19 ein Stichprobenset des Datensatzes aus Caption und Bildern vorgestellt. Diese jeweils fünf Elemente sind die ersten Bilder und Captionpaare der fünf häufigsten Sprachen Englisch, Französisch, Deutsch, Cebuano und Japanisch. Sie werden aus dem originalen Datensatz entfernt und dienen als Visualisierung der Performanz der Modelle in den folgenden Abschnitten. Dazu werden sie später in einer Cosine Similarity Matrix angeordnet, um die Zuordnung der Modelle zu verdeutlichen.

Abbildung 19

Darstellung des Stichprobensets aus dem WIT-Datensatz



Anmerkung. Captions:

Englisch, erstes Bild: *Silver spoon </s> Two silver-gilt strainer spoons and a cignus spoon decorated with a mythical marine creature. (4th century AD Roman spoons from the Hoxne Hoard.)*

Französisch, zweites Bild: *Grand Prix automobile de France 1957 </s> Juan Manuel Fangio et sa Maserati 250F : la combinaison à battre en 1957.*

Deutsch, drittes Bild: *Ninestar </s> Ninestar G&G Druckerpatronen in einer Berliner Karstadt Filiale.*

Cebuano, viertes Bild: *Johnson Point </s> Mga dapit nga gitawag Johnson Point sa South Georgia and the South Sandwich Islands (Hiniusang Gingharian).*

Japanisch, fünftes Bild: *国営ひたち海浜公園 </s> 観覧車「プリンセスフラワー」*

Aus "Wit: Wikipedia-based image text dataset for multimodal multilingual machine learning" von K. Srinivasan et al., 2021, Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval.

Diese Bilder stellen typische Beispiele für die Mehrheit der Bilder im Datensatz dar. Stichproben während der Untersuchung der Daten haben gezeigt, dass vor allem in dem cebuanischen Datenanteil ein sehr großer Anteil an Landkarten vorhanden ist. Aber auch Bilder von historischen Objekten oder von zeitgenössischen Ereignissen wie der Grand Prix 1957 sind Teile des Datensatzes, genau wie Bilder von alltäglichen Dingen wie beispielsweise in Abbildung 20 dargestellt Druckerpatronen oder das Riesenrad 'Princess Flower' im japanischen Hitachi Seaside Park.

Die Captions in den späteren Cosine Similarity Matrizen werden aus Platzgründen nur mit dem Sprachkürzel identifiziert. Die erste Matrix der Bild- und Textvergleiche wird im Ergebnis des nächsten Abschnitts dargestellt, um die Zuordnung durch Zero Shot CLIP darzustellen.

Damit schließt die Visualisierung des Datensatzes. Es wurde ein tieferer Einblick der Eigenschaften der beiden Attribute gegeben und es kann nun mit der Umsetzung der Benchmark durch Zero Shot CLIP fortgefahren werden.

4.3 Benchmark Zero Shot CLIP

Für die Erstellung der Benchmark müssen die Daten für den Text- und Image Encoder von CLIP vorbereitet werden. Das Ziel ist es, wie in den Abschnitten zuvor beschrieben, Bild- und Textembeddings zu erstellen und durch die Cosine Similarity die Zuordnung der Bilder und Texte festzulegen. Dazu werden zuerst die Bilder für die weitere Verwendung vorverarbeitet und im Anschluss folgen die Captions. Auch wird zunächst am Beispiel der Stichproben, die im letzten Abschnitt vorgestellt wurden, der Ablauf dieses Vorgangs veranschaulicht, um daraufhin die restlichen Daten zu klassifizieren und durch die Evaluation im Abschnitt Ergebnis die Performanz des Modells wiederzugeben.

Die Vorverarbeitung der Bilder wird durch die CLIP-eigene Funktion *preprocess()* vorgenommen. Hier wird zuerst die Pixelintensität mithilfe des Mittelwerts und der Standardabweichung des Datensatzes normalisiert. Daraufhin wird durch die Funktion die Größe der Eingabebilder angepasst und sie werden zentriert, damit sie mit der vom Modell erwarteten Bildauflösung übereinstimmen (OpenAI, 2021). Das Ergebnis dieser Vorverarbeitung wird daraufhin in Tensoren umgewandelt. Aus diesen wird nun durch die Übergabe an den Image Encoder mit der *model.encode_image()* Methode die Embedding mit der Dimension 512 codiert.

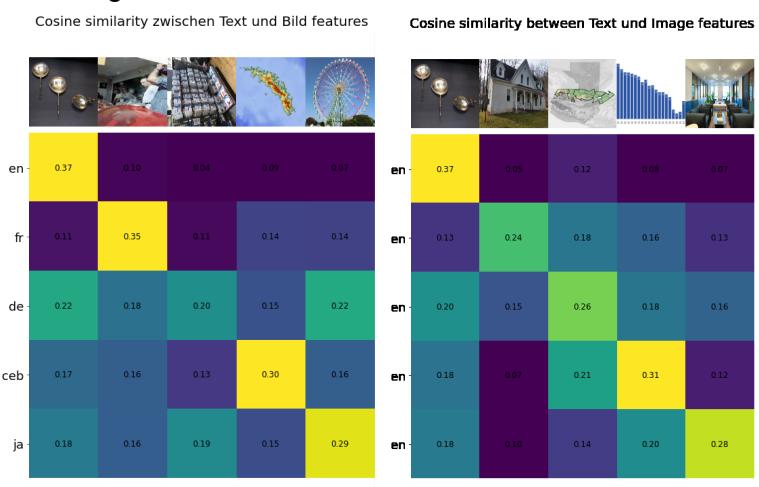
Ähnlich der Vorverarbeitung der Bilder wird bei der Vorverarbeitung der Captions vorgegangen. Der erste Schritt ist die Tokenisierung des Inputtextes durch die *clip.tokenize()* Methode basierend auf Byte-Pair-Encoding. Hier werden dem Text die CLIP eigenen Token <|startoftext|> und <|endoftext|> hinzugefügt und er wird auf 77 Zeichen begrenzt. Das Resultat der Funktion sind Tensoren, die per *model.encode_text()* an den Text Encoder übergeben werden. Dieser gibt die codierten Text-Features aus, die zusammen mit den Bild-Features nun durch das Berechnen der Cosine Similarity einander gegenübergestellt werden können.

Zur Berechnung dieser Cosine Similarity werden die beiden Feature normalisiert und daraufhin wird mit ihnen das Skalarprodukt eines jeden Paars berechnet. Für die Klassifikation kann man diese Cosine Similarity (multipliziert mit 100) als Logit der Softmax-Operation benutzen. Dieser Vorgang wird im nächsten Abschnitt als Erstes an Hand des Stichprobensets veranschaulicht. Es wird jedoch auch, um die Effektivität des CLIP Modells zu testen und mit dem multilingualen Ergebnis vergleichbar zu machen, die Klassifikation des rein englischen Anteils des Datensatzes vorgenommen. Das Ergebnis dieser Klassifikation wird daraufhin mit dem Ergebnis der Klassifikation des gesamten, multilingualen Datensatzes in Beziehung gesetzt.

4.3.1 Ergebnis

In Abbildung 20 wird die Cosine Similarity Matrix des Stichprobensets visualisiert und als Vergleich wird ebenfalls die Cosine Similarity Matrix der ersten fünf Bild- und Textpaare des englischen Anteils des Datensatzes dargestellt. In der Diagonalen sind die Werte der Gegenüberstellung der Bilder und Captions zu sehen. Wie in der linken Darstellung zu erkennen ist, wurden trotz der unterschiedlichen Sprachen die meisten der Zuordnungen durch CLIP mit dem höchsten Wert versehen. Nur im Fall der deutschen Zuordnung wurde durch CLIP ein falsches Ergebnis erreicht. Im Vergleich dazu wurden in der englischen Stichprobenuordnung alle Ergebnisse richtig ausgewiesen.

Abbildung 20
Darstellung der Zuweisung durch Zero Shot CLIP



Anmerkung. Links ist die Darstellung des multilingualen Stichprobensets. Rechts die des Englischen.

Diese beispielhafte Darstellung der Zuordnung durch CLIP steht für den Vorgang, der für alle restlichen Bild- und Textpaare vorgenommen wird. Zuerst wird anhand des englischen Anteils des Datensatzes ein vergleichbares Ergebnis erstellt. Dafür wird, wie in der Konzeption beschrieben, die Evaluationsart des Top k Accuracy herangezogen. Für die Evaluation wird eine richtige Zuweisung für ein Bild gezählt, falls die zugehörige Caption innerhalb der Top $k = 20$ Treffer zurückgegeben wird. Im Fall des englischen Datensatzanteils ist die Retrievalqualität des Modells mit 78,89% sehr gut. Im Vergleich dazu liegt die Genauigkeit des Modells bei dem multilingualen Datensatz bei nur 35,65%.

Dieser Wert beschreibt die Benchmark, die es durch die Implementation des multilingualen, multimodalen Modells und die Optimierungsstrategien zu verbessern gilt.

4.4 Multilinguale Modell

Die Implementation des multilingualen, multimodalen Modells wurde auf mehrere Klassen aufgeteilt. Dabei ist die Umsetzung des Modells in der Architektur teilweise der Vorgehensweise von Abeywardana (2021) nachempfunden, bei dem im Gegensatz zu dieser Arbeit das distilbert-multilingual Modell als Text Encoder verwendet wurde.

Die Umsetzung beginnt mit der Erstellung des Tokenizers. Wie in der Konzeption beschrieben und ähnlich der Tokenisierung des CLIP-Tokenizers, werden den Inputtexten die schon beschriebenen Abgrenzungs- und Positionstoken hinzugefügt. Ebenfalls wird die Länge auf 77 Zeichen begrenzt und der Input in einen Tensor umgewandelt. Das Ergebnis der Dekodierung ist die Rückgabe eines Dictionary aus input_ids und der attention_mask.

Darauf folgend wird der Text Encoder mit dem xlm-roberta-base Modell als Basis strukturiert. Diesem wird mit dem Parameter *text_encoder_model* das Modell übergeben. Die Schichten des Transformers werden bis auf die letzte für das Training eingefroren und die internen Gewichtungen werden nicht neu trainiert. Das Ergebnis dieser Encoderklasse ist die Ausgabe eines Vektors mit der Dimension 768.

Ähnlich dazu wird der Image Encoder umgesetzt. Die Basis bildet der schon beschriebene CLIP-Encoder mit dem neuronalen Netz RN50x4. Es wird wieder das Backbone des Modells eingefroren, sodass nur die letzte Schicht für das Training relevant ist. Die Ausgabe ist in diesem Fall die Umwandlung der Bilder in einen Vektor mit der Dimension 640.

Die nun erstellten ungleichen Embeddingdimensionen werden durch die Projektion-Head-Klasse in eine Dimension gebracht, um so zu ermöglichen, dass sie später

vergleichbar sind. Dafür werden der Klasse die Parameter `dim_in` und `dim_out` übergeben, welche die Größe der Inputvektoren und Outputvektoren bestimmen. Die Größe der Outputvektoren wird auf die Dimension 512 festgelegt.

Für die bessere Handhabbarkeit der Daten und deren Umwandlung wird zusätzlich eine Datensatzklasse erstellt. Hier wird die Vorverarbeitung der Daten ähnlich der Vorverarbeitung des CLIP-Modells im letzten Abschnitt gemanagt. Im Zusammenspiel mit der Umsetzung eines manuellen Dataloaders wird so der Fluss der Daten für das Training gesteuert. Der Dataloader regelt die Aufteilung der nun vorverarbeiteten Daten in ein Trainings- und Evaluationsset, das gemäß der übergebenen Batchgröße separiert wird.

Die Architektur schließt mit der Umsetzung der Modellklasse mithilfe der PyTorch Lightning Bibliothek. Der Vorteil von PyTorch Lightning ist die kompaktere Strukturierung des originalen PyTorch Code (Falcon, 2019). Durch das *LightningModule* wird der Code in fünf bis sechs Hauptblöcke unterteilt.

- Der erste Block besteht aus dem normalen Setzen der Instanzattribute (`init`). Im Fall dieser Implementation besteht er aus dem Setzen der Encodermodelle, dem Übergeben des Tokenizers und der Dimensionsgrößen für die weitere Verarbeitung in den Encodern.
- Der nächste Block ist der komplette Trainingsloop. Hier startet das Training des Modells mit der Überschreibung der `training_step()` Methode.
- Der Validation Loop kann durch die Überschreibung der `validation_step()` Methode aktiviert werden.
- Der Test Loop (`test_step`) wird in dieser Umsetzung nicht benötigt. Genauso wie der Prediction Loop (`predict_step`).
- Der letzte Block ist die Festlegung des Optimizers und der Learning-Rate-Planer über `configure_optimizers()`.

Die kompakte Erstellung dieses Trainingsablaufs in einer Klasse ermöglicht einen besseren Überblick und eine einfachere, effektivere Trainingssteuerung.

4.4.1 Training

Der weitere Trainingsablauf des Modells wird durch den PyTorch Lightning Trainer automatisiert. Im Detail wird durch den Trainer die automatische Aktivierung bzw. Deaktivierung der Gradientenberechnung, die Abwicklung des Trainings und Validation Data Loader sowie die automatische Einteilung der Batches vollzogen (Falcon, 2019).

Der Training Step wird in der Umsetzung noch um den Common Step erweitert, in den die meiste Logik des Training Step ausgegliedert wird. Hier werden die Bild- und Textbatches aus dem übergebenen Input erstellt. Ebenso wird hier die Ähnlichkeit der Bilder und Captions mit dem Skalarprodukt und das Ergebnis der Verlustfunktion berechnet. Im Training Step wird der Common Step aufgerufen und dann das Ergebnis der Verlustberechnung ausgegeben. Innerhalb des Validation Step wird am Ende jeder Epoche das Model auf dem Validationset evaluiert.

Insgesamt wurde das Modell in fünf Epochenschritten iterative über mehrere Wochen bis zu 45 Epochen trainiert. In Abbildung 21 wird beispielhaft der Trainingsverlauf über die ersten fünf Epochen dargestellt. In der rechten Abbildung ist der abnehmende Trainingsverlust über die Epochen hinweg zu sehen.

Abbildung 21

Darstellung des Trainingsverlaufs und des Trainingsverlusts über die ersten fünf Epochen



Der letzte Vorgang ist die Konfigurierung des Optimizers. Hier wird der in der Konzeption beschriebene Adam Optimizer festgelegt und die jeweiligen Learning Rate Parameter für die beiden Encoder.

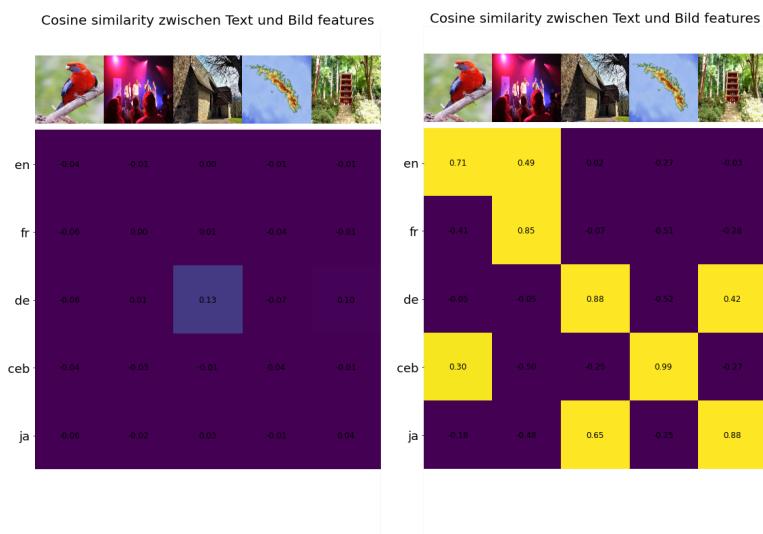
Verlustfunktion

Die Umsetzung der Berechnung der Verlustfunktion wird auf mehrere Funktionen aufgeteilt. Hier werden die beiden Vektoren aus den Batches aufeinander ausgerichtet und der zweite für die Berechnung transponiert. Die korrespondierenden Bild- und Textwerte müssen so

weit wie möglich an die Zahl 1 angenähert und alle anderen Richtung 0 bewegt werden. Für die jeweiligen Captions wird das Softmax-Skalarprodukt für alle Bilder berechnet und daraufhin die Kreuzentropie als Kostenfunktion herangezogen. In der praktischen Umsetzung wird innerhalb der PyTorch Bibliothek das Skalarprodukt mithilfe des @ Operators gebildet. Das Gleiche wird für jedes Bild des Datensatzes gemacht. Da Bilder und Captions als Paare an das Modell übergeben werden, entsteht so eine Matrix mit den Ähnlichkeitswerten in einer Diagonalen.

Abbildung 22

Darstellung der Zuweisung während des Trainings durch das multilinguale Modell...



In Abbildung 22 wird der Vorgang der Zuordnung anhand von Matrizen aus Bild- und Textpaaren aus dem Evaluationsset dargestellt. Die linke Matrix stellt die Zuweisung des Modells nach dem Training von fünf Epochen dar. Die rechte Matrix stellt das Ergebnis des Modells nach 45 Epochen dar. Zu erkennen ist, dass über die Epochen hinweg die Ähnlichkeitswerte in der Diagonalen immer ausgeprägter werden. Während nach fünf Epochen das Modell nur eine richtige Zuweisung für das deutsche Bild- und Textpaar vornahm, weist es nach 45 Epochen schon bei allen Sprachen eine korrekte Zuweisung vor.

4.4.2 Ergebnis

Um das Ergebnis des Trainings des multilingualen, multimodalen Modells vergleichbar zu machen, wird es wieder durch die Ermittlung der Retrievalqualität erstellt. Dazu wird, wie im Fall der Evaluation des CLIP-Modells, die Top k Accuracy mit $k = 20$ gewählt. Für eine

bessere Veranschaulichung, wie sich die Retrieval-Performance über die Epochen hinweg verbessert, wurde die Genauigkeit des Modells nach fünf, fünfundzwanzig und fünfundvierzig Epochen gemessen.

Die Bewertung der multilingualen Zuweisung wird auf dem vor dem Training ausgegliederten Testset durchgeführt. Nachdem das Modell für fünf Epochen trainiert wurde, ergab sich eine Retrievalqualität von nur 1,9%. Nach einem Training für 25 Epochen stieg die Genauigkeit des Modells auf 21,31%. Am Ende des Trainings nach 45 Epochen lag die letztendliche Genauigkeit des multilingualen Modells bei 35,27% und ist vergleichbar mit der Performanz der Benchmark des originalen CLIP-Modells. Aufgrund des umfassenden Datensatzes und den daraus resultierenden intensiven Rechenressource für das Training wird die Performanz bei diesem Ergebnis belassen und es werden noch weitere Optimierungsstrategien untersucht, die die Benchmark verbessern können.

4.5 Optimierungsstrategien

Die Implementation der Optimierungsstrategien folgt den Vorgaben der Konzeption. Die Umsetzung der beiden Strategien wird im Folgenden vorgestellt und beginnt mit dem auf der Arbeit von Reimers und Gurevych (2020) fußenden Sentence Transformer Modell durch Knowledge Distillation. Im Anschluss wird der Vorgang des multilingualen Prompt Engineering mit beiden Prompts erläutert.

Das Ziel ist es hier, die Benchmark des CLIP Modells entweder durch das trainierte multilinguale Modell mit Hilfe von Prompt Engineering oder durch die Nutzung des Sentence Transformer Modells zu verbessern.

4.5.1 Sentence Transformer Modell

Das multilinguale Sentence Transformer Modell clip-ViT-B-32-multilingual-v1 (Reimers, o.D.) wird über das Sentence Transformer Framework installiert. Dieses Framework ermöglicht es, Text Embeddings in über 100 Sprachen zu berechnen. Es basiert auf PyTorch und nutzt Quellen aus den Transformer Umsetzungen von Hugging Face. Für die Bearbeitung von NLP-Aufgaben wird vom Framework eine große Auswahl an verschiedenen vortrainierten Modellen zur Verfügung gestellt, die das Finetunen auf neuen Daten ermöglichen. Um die Effizienz der Modelle zu garantieren, werden sie auf unterschiedlichen Aufgaben getestet und erreichen bei einigen State Of The Art Performanz (Reimers & Gurevych, 2020).

Die Modelle generieren gleichgerichtete Vektorräume, so werden zum Beispiel ähnliche Sequenzen in verschiedenen Sprachen eng aneinander im Vektorraum abgebildet. Dabei ist es nicht notwendig, die Sprache der Eingabesequenz genauer zu spezifizieren (Reimers & Gurevych, 2020).

Das clip-ViT-B-32-multilingual-v1 Modell ist ein multilingualer Text Encoder, der aus dem clip-ViT-B-32 durch Multilingual Knowledge Distillation entworfen wurde. Dieses Modell kann Text in über 50 Sprachen kodieren, um Bildvektoren des clip-ViT-B-32 abzugleichen.

Um dieses Modell für diese Arbeit zu verwenden, wird ein Textmodell des clip-ViT-B-32-multilingual-v1 Modells und ein Bildmodell der Art clip-ViT-B-32 instanziert. Die Vorverarbeitung der Bilder und Captions folgt denen in den vorherigen Abschnitten. Zunächst wird die Größe der Bilder angepasst und sie werden für das CLIP-Modell zentriert. Die Captions werden tokenisiert und ihre Länge zugeschnitten. Daraufhin werden sie an die Bild- und Text Encoder weitergeleitet. Diese erstellen die jeweiligen Embeddings und richten diese Vektoren in einer Matrix aus. Hier wird wieder durch Contrastive Loss die Cosine Similarity berechnet.

4.5.2 Multilinguale Prompt Engineering

Für die erfolgreiche Umsetzung des multilingualen Prompt Engineering werden mehrere Schritte benötigt. Im Gegensatz zu der Herangehensweise der ersten Optimierungsstrategie wird in diesem Fall die Struktur des Textinputs verändert. Wie in der Konzeption beschrieben und in Abbildung 14 in Abschnitt 3.6.2 schon dargestellt, wird vor der eigentlichen Übergabe an den Text Encoder ein weiteres Transformermodell eingefügt.

Das in der Konzeption vorgestellte Modell facebook/m2m100_418M, das von Fan und Kollegen (2021) erstellt wurde, wird für die Erstellung der multilingualen Prompts herangezogen. Neben der Untersuchung von Radford und Kollegen (2021) zu der Nutzung von Prompt Engineering wurde auch durch die Arbeit von Zhou und Kollegen (2022) gezeigt, dass es möglich ist durch Hinzufügen von Textsequenzen die Performanz von Sprachmodellen zu beeinflussen. Weiterhin weisen sie darauf hin, dass es schwierig sein kann, die richtige Art von Prompts zu erstellen, um die Performanz zu erhöhen. “However, identifying the right prompt is a non-trivial task, which often takes a significant amount of time for words tuning—a slight change in wording could make a huge difference in performance” (Zhou et al., 2022, S. 2). Allerdings zeigten sie in ihrer Arbeit ebenfalls, dass das Einfügen eines beschreibenden *a* vor einer Klasse in dem monolingualen, englischen

Caltech101 Datensatz (Li et al., 2004) eine Verbesserung der Genauigkeit von 5% nach sich zog.

Deswegen werden im Folgenden die in der Konzeption erwähnten Prompts "[a photo of](#)" und "[From Wikipedia:](#)" auf ihre Auswirkung bezüglich ihrer die Retrievalperformanz der Modelle untersucht. Dazu werden zunächst die zwei Prompts durch das facebook/m2m100_418M Modell in alle 94 Sprachen des Datensatzes übersetzt. Daraufhin werden die übersetzten Prompts ihren entsprechenden Captions angefügt.

Bei der Übersetzung stellte sich heraus, dass 23 der Sprachen nicht durch das facebook/m2m100_418M Modell unterstützt werden, dazu gehören die Sprachen: 'zh-TW', 'eo', 'iw', 'eu', 'lah', 'sco', 'be-tarask', 'tg', 'te', 'nn', 'nds', 'ckb', 'sr-Latn', 'fil', 'bar', 'tt', 'ce', 'azb', 'la', 'yue', 'xmf', 'lmo', 'arz'. Diese behalten ihre ursprüngliche Struktur bei und werden unverändert an den Text Encoder übergeben.

Dieser berechnet mit den übergebenen, erweiterten und nicht erweiterten Captions die Embeddings. Der weitere Ablauf ist identisch mit dem schon bekannten Schema. Der Image Encoder gibt die Embeddings der Bilder aus und die beiden Vektoren werden aufeinander ausgerichtet. Schlussendlich wird durch Contrastive Loss die Ähnlichkeit der Bild- und Textpaare berechnet.

4.5.3 Ergebnis

Die Ergebnisse der beiden Strategien werden hier zur besseren Übersicht in einem Abschnitt dargestellt. Beginnend mit dem Ergebnis der Sentence Transformer Strategie. Für eine bessere Vergleichbarkeit zu den anderen Modellen wird wieder auf die beispielhafte Darstellung der Zuordnung des Stichprobensets der fünf häufigsten Sprachen aus dem Abschnitt der Benchmark Erstellung zurückgegriffen.

Abbildung 23

Darstellung der Zuweisung durch das Sentence Transformer Modell...

Cosine similarity zwischen Text und Bild features



Wie zu erkennen, konnte in diesem Beispiel (Abbildung 23), im Gegensatz zum letzten Test während der Benchmark, durch das Sentence Transformer Modell jede multilinguale Caption einem Bild zugeordnet werden. Auch beim Evaluationstest durch die Top 20 Genauigkeit kann diese Verbesserung festgestellt werden. Das Sentence Transformer Modell erreicht mit einer Retrievalqualität von 44,09% das beste Ergebnis von allen Modellen.

Die Anwendung von Prompt Engineering in Verbindung mit den multilingualen Captions zeigt eine Veränderung der Performanz in beiden Modellen. Im Fall des ersten Prompts "[a photo of](#)" wurde die Retrievalqualität des erstellten multilingualen Modells von 32,72% auf 29,4% reduziert. Auch bei der Performanz des Sentence Transformer Modells wurde das Ergebnis von 44,09% auf 42,99% reduziert. Auch im Fall des zweiten Prompt "[From Wikipedia:](#)" konnte keine Verbesserung der Modelle erreicht werden. Die Genauigkeit sank bei dem erstellten multilingualen Modell durch den zweiten Prompt noch stärker auf 25,0% und bei dem Sentence Transformer Modell auf 38,91%.

5. Evaluation

Für die genauere Evaluation der Modelle, werden hier die Ergebnisse der Implementation noch einmal zusammengefasst und erläutert. Wie in der Konzeption bereits erwähnt, wurde die Top 20 Accuracy als Evaluationsmittel für den Vergleich der Retrievalqualität der Modelle gewählt. Die angegebenen Prozentzahlen für die Performanz der Modelle beschreiben also die Anzahl an korrekten Captions unter den gesamten Top 20 zurückgegebenen Captions, welche ein gesuchtes Bild beschreiben.

Die erste Untersuchung der Performanz des CLIP-Modells in Abschnitt 4.3 auf dem monolingualen englischen Datensatzanteil ergab eine sehr gute Retrievalqualität von 78,9%. Der weitere Retrievaltest durch Zero-Shot-Zuweisung von CLIP auf den multilingualen Daten reduzierte dieses Ergebnis und setzte die Benchmark der Arbeit auf 35,6%. Der Performanzvergleich mit den anderen Modellen ist in Tabelle 1 für eine bessere Übersicht aufgeführt.

Tabelle 1

Vergleich der Performanz der einzelnen Modelle

Modell	Retrieval Qualität	Retrieval Qualität mit Prompt 1	Retrieval Qualität mit Prompt 2
CLIP Zero Shot (Benchmark)	0.356	-	-
Multilinguale Modell(25-Epochen)	0.21	0.159	0.153
Multilinguale Modell(45-Epochen)	0.327	0.294	0.25
Sentence Transformer Modell	0.441	0.429	0.389

Anmerkung. Angabe der Treffgenauigkeit in Prozent.

Nach der Setzung der Benchmark mit Zero Shot CLIP folgt die Genauigkeit des multilingualen Modells nach 25 Epochen Training. Dieser Zwischenschritt ist aufgeführt, um die Verbesserung des Modells über die Epochen hinweg hervorzuheben. Wie zu erkennen ist, wurde eine Steigerung der Performanz des Modells von 21% auf 32,7% nach 45 Epochen erreicht. Dies bringt die Genauigkeit des multilingualen Modells nach 45 Epochen

in den Bereich des CLIP-Modells, verbessert die Benchmark jedoch nicht. Erst mit der Verwendung des multilingualen Sentence Transformer Modells kann die Benchmark um fast zehn Prozentpunkte auf 44,99% verwendet werden. Die zweite Optimierungsstrategie durch Prompt Engineering zog jedoch keine Verbesserung nach sich. Durch beide Versuche mit der Erweiterung der multilingualen Captions mit beschreibenden Prompts wurde eine Verschlechterung der Performanz, sowohl bei dem erstellten multilingualen Modell, als auch bei dem Sentence Transformer Modell erzielt.

6. Fazit

In dieser Arbeit wurde das Ziel verfolgt, mithilfe von CLIP als Ausgangspunkt ein multilinguales, multimodales Machine Learning Modell zu entwickeln, mit dem das Problem der Wikipedia Image/Caption-Competition bearbeitet werden kann. Im Verlauf der Arbeit wurde gezeigt, wie mit der Architektur eines Dual-Encoder-Systems und dem durch den Wettbewerb bereitgestellt WIT-Datensatz dieses Ziel erreicht wurde.

Dazu wurden in Abschnitt 2 zunächst die genauere Auseinandersetzung mit der Struktur des Wettbewerbs und den allgemeinen theoretischen Grundlagen für die Umsetzung vorgenommen. Die erste Untersuchung der Zusammensetzung des Datensatzes in diesem Abschnitt ergab, dass dieser eine sehr umfangreiche Anzahl an Bildern, Metadaten, multilingualen Artikelinformationen und Bildbeschreibungen enthält. Anhand der Struktur der Daten wurde deutlich, dass die wichtigsten Inhalte für die Bearbeitung die Bilder und die caption_title_and_reference_description sind. Nachdem ein Überblick über den Aufbau des Datensatzes gegeben wurde, sind die Grundlagen der hinter dem Modell stehenden Theorien untersucht worden. Hier wurde zur besseren Veranschaulichung zunächst der Werdegang der NLP-Forschung wiedergegeben, welcher die Nutzung von multimodalen Modellen auf Basis der CLIP Architektur heute möglich macht. Dabei wurde der Paradigmenwechsel Ende der 1980er Anfang der 1990er Jahre hervorgehoben, welcher die Einführung des statistischen NLP nach sich zog und den Weg ebnete für die erfolgreiche Verwendung von Deep Learning Methoden in den letzten Jahren. Vor allem die Entwicklung der Transformer-Architektur 2017 von Vaswani und Kollegen zeigte sich als Katalysator für den Entwurf vieler Sprachmodelle. Drei dieser Modelle der letzten Jahre wurden genauer dargestellt. Das wichtigste davon für diese Arbeit ist XLM-RoBERTa (XLM-R), da es als Text Encoder für das multilinguale Modell verwendet wird. Die beiden anderen sind die Vorgängermodelle BERT und RoBERTa, die die Erstellung von XLM-RoBERTa (XLM-R)

ermöglichen. Als Nächstes wurde die Architektur des Basismodells der Arbeit genauer untersucht. Die Eigenschaften des CLIP-Modells und auch die Methode des Pre-Training waren hier Thema. Als abschließender Punkt wurden weitere multilinguale, multimodale Modelle der letzten Jahre dargestellt.

Abschnitt 3 drehte sich um die Konzeption der Benchmark, des multilingualen, multimodalen Modelles, der Optimierungsstrategien und um die Evaluation der Modelle. Dafür wurde anhand des CRISP-DM Prozessmodells ein Ablaufplan vorgestellt, der die Entwicklung dieser einzelnen Schritte besser strukturiert. Angefangen mit der Planung des Data Understanding, in dem es darum geht, die genaue Zusammensetzung des Datensatzes zu verstehen und einen tieferen Einblick zu ermöglichen als der erste Überblick in Abschnitt 2. Das Data Understanding hängt eng mit dem Vorgang der Data Preparation zusammen. In diesem zweiten Schritt der Planung wurde konzipiert, wie der Datensatz mithilfe von Pythonbibliotheken wie Pandas oder Numpy so bearbeitet werden kann, sodass mit diesem das multilinguale Modell trainiert werden kann. Nach der Darstellung der Data Preparation wurde der Ablauf vorgestellt, wie mit Zero Shot Klassifikation durch CLIP eine Benchmark für die Arbeit erstellt werden kann. Im Anschluss wurde die Modellierung des multilingualen, multimodalen Modells ausgeführt. Hier wird die Konzeptionierung der einzelnen Aspekte des Modells vorgenommen. Dazu gehören die Planung der Struktur des Text Encoders, des Image Encoders und des Projection Heads, der die Ausgaben der Encoder in die gleiche Dimension bringt. Ebenso wird für den Ablauf des Trainings die Verlustfunktion des Contrastive Loss vorgestellt. An diesen Teil schließt sich die Konzeptionierung der Optimierungsstrategien an. Durch das Einführen eines weiteren Dual Encoder Modells durch clip-ViT-B-32-multilingual-v1 und der Verwendung von Prompt Engineering wird geplant, die Benchmark noch weiter zu verbessern. Als Evaluationsmethode der Modelle wird abschließend die Top k Genauigkeit herangezogen.

Der Inhalt von Abschnitt 4 dreht sich um die Implementation der in der Konzeptionierung vorgestellten Abläufe. Das Data Understanding zusammen mit der Data Preparation zeigen die wichtigsten Merkmale des Datensatzes und bereinigen diesen, sodass ein fertiges Trainings- und Testset entstehen. Die Umsetzung der Benchmark durch Zero Shot Klassifikation mit CLIP wird daraufhin auf den Daten vorgenommen. Anschließend wird die Erstellung des multilingualen, multimodalen Modells mithilfe der PyTorch Lightning Bibliothek erläutert. Ebenfalls wird hier ein Überblick über den rechenintensiven Trainingsverlauf gegeben und gezeigt, dass nach 35 Epochen Training die Performanz des erstellten Modells einen etwas schlechteren Wert erreicht, als der der Benchmark. Entsprechend des Konzeptionsablaufs wird dann die Implementierung der Optimierungsstrategien vorgenommen. Durch die Anwendung von Prompt Engineering konnte hier eine

Verschlechterung der Retrievalqualität der Modelle beobachtet werden. Das Ergebnis des multilingualen Sentence Transformer Modells hingegen verbessert die Benchmark.

In der sich anschließenden Evaluation werden diese Ergebnisse noch einmal im Vergleich gegenübergestellt. Es zeigt sich, dass die Benchmark mit einer Top $k = 20$ Genauigkeit von 35,6% durch das multilinguale Modell nach 35 Epochen mit 32,7% nicht ganz erreicht werden konnte. Durch das Prompt Engineering konnten die Ergebnisse von Radford und Kollegen (2021) nicht umgesetzt werden und es zeigte sich eine Verschlechterung der Performanz in allen Modellen. Schlussendlich zeigte die Verwendung des clip-ViT-B-32-multilingual-v1 Sentence Transformer Modells jedoch eine Verbesserung der Benchmark auf 44,1%.

6.1 Limitation und Bias

Durch die Nutzung des Image Encoders des CLIP-Modells werden auch Limitationen und Bias übernommen. So haben Radford und Kollegen (2021) festgestellt, dass im Vergleich zu aufgabenspezifischen Modellen die Leistung von Zero Shot bei mehreren Arten von spezifischer Klassifizierung wie der Unterscheidung von Automodellen, Blumenarten und Flugzeugvarianten noch recht schwach ist. Auch bei abstrakten und systematischen Aufgaben wie dem Zählen der Anzahl von Objekten in einem Bild hat CLIP Schwierigkeiten.

Im Hinblick auf die große Anzahl an unterschiedlichen Sprachen im Datensatz ist auch die Fairness des multilingualen Text Encoders eine Frage. So stellen Wang und Kollegen (2022) in ihrer Arbeit die Frage: “Nevertheless, the principle of multilingual fairness is rarely scrutinized: do multilingual multimodal models treat languages equally? Are their performances biased towards particular languages?”(S. 1). Es ist also nicht auszuschließen, dass durch die große Anzahl an unterrepräsentierten Sprachen im Datensatz einige weniger beachtet werden als andere. Während der Datenvisualisierung wurde gezeigt, dass viele Sequenzen eine Länge besitzen, die größer ist als die Länge von 77 Zeichen, die für den Text Encoder benötigt werden. Hier stellt sich ebenfalls die Frage, ob durch die Begrenzung der Captions auf eine geringere Länge einige Informationen, die für die Zuordnung im Modell hilfreich sein könnten, verloren gehen.

Um Bias in ihrem Modell zu bemessen haben Radford und Kollegen (2021) die Performanz von Zero Shot CLIP und einem Logistic Regression Classifier durch CLIP auf dem FairFace Datensatz (Kärkkäinen & Joo, 2019) getestet. Sie haben herausgefunden, dass der Logistic Regression Classifier eine höhere Genauigkeit als Kärkkäinen und Kollegens (2019) Modell

erreicht. Die Performanz von Zero Shot CLIP variiert hingegen von Kategorie zu Kategorie und ist in einigen besser als das Modell von Kärkkäinen und Kollegen (2019) sowie in einigen schlechter.

6.2 Ausblick

Nachdem das multilinguale, multimodale Modell erstellt und die Optimierungsstrategien implementiert wurden, sind mehrere Ansatzpunkte für Verbesserungen deutlich geworden. Zum Ersten könnte das Training des Modells um mehrere Epochen erweitert werden, um zu beobachten, bis zu welchem Zeitpunkt die Performanz weiter steigt. Eine weitere Verbesserungsmöglichkeit wäre, das Training an die Teacher-Student-Architektur von Reimers und Gurevych (2020) anzupassen. Dafür müsste jedoch zunächst ein Datensatz erstellt werden, der als Anker Texte in englischer Sprache besitzt und Übersetzungen in alle 94 Sprachen des WIT Datensatzes beinhaltet.

Auch die Fokussierung auf die häufigsten Sprachen könnte eine Verbesserung der Performanz nach sich ziehen. Denn wie Bianchi und Kollegen (2021) hervorheben, ist es so, dass multilinguale Modelle in der Regel nicht so gut abschneiden wie sprachspezifische Modelle. Allerdings wäre diese Herangehensweise konträr zu dem Ziel des Wettbewerbs ein Modell zu entwickeln, dass möglichst umfangreich multilingual agieren kann.

Weiterhin könnte die Einbeziehung anderer Attribute des Datensatzes wie *page_titel* oder *context_page_description* einen positiven Einfluss auf die Zuweisung des Modells generieren. Als weitere Option könnte die Untersuchung anderer Prompts auch mithilfe von Context Optimization (Zhou et al., 2022) zu Verbesserungen führen. Mit diesen Ausblicken auf mögliche Optimierungen in weiteren Forschungen schließt diese Arbeit.

Literaturverzeichnis

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., ... Zheng, X. (2016). {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (S. 265–283). Usenix.
<https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi>
- Abeywardana, S. (2021). Multilingual CLIP with Huggingface + PyTorch Lightning. GitHub.
https://github.com/sachinruk/blog/blob/master/_notebooks/2021-03-07-CLIP.ipynb
- Apache Arrow. (o.D.). *Feather File Format*.
<https://arrow.apache.org/docs/python/feather.html>
- Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). *Layer normalization*. arXiv.
<https://doi.org/10.48550/arXiv.1607.06450>
- Balakrishnan, V., & Lloyd-Yemoh, E. (2014). Stemming and lemmatization: A comparison of retrieval performances. *Lecture Notes on Software Engineering*, 2(3), 262–267.
- Barrault, L., Bougares, F., Specia, L., Lala, C., Elliott, D., & Frank, S. (2018). Findings of the third shared task on multimodal machine translation. In *Proceedings of the third conference on machine translation: Shared task papers* (S. 304–323). Association for Computational Linguistics. <https://doi.org/10.18653/v1/W18-6402>
- Bebis, G., & Georgopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials*, 13(4), 27–31. <https://doi.org/10.1109/45.329294>
- Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2006). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 1–8.
<https://proceedings.neurips.cc/paper/2006/hash/5da713a690c067105aeb2fae32403405-Abstract.html>
- Bianchi, F., Attanasio, G., Pisoni, R., Terragni, S., Sarti, G., & Lakshmi, S. (2021). *Contrastive language-Image pre-training for the Italian language*. arXiv.
<https://doi.org/10.48550/arXiv.2108.08688>
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877–1901.
<https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html>

- Calixto, I., Liu, Q., & Campbell, N. (2017). *Multilingual multi-modal embeddings for natural language processing*. arXiv. <https://doi.org/10.48550/arXiv.1702.01101>
- Carlsson, F., Eisen, P., Rekathati, F., & Sahlgren, M. (2022). Cross-lingual and multilingual CLIP. In N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk, & S. Piperidis (Hrsg.), *Proceedings of the thirteenth language resources and evaluation conference* (S. 6848–6854). European Language Resources Association. <https://aclanthology.org/2022.lrec-1.739>
- Chambers, C., Raniwala, A., Perry, F., Adams, S., Henry, R. R., Bradshaw, R., & Weizenbaum, N. (2010). FlumeJava: easy, efficient data-parallel pipelines. *ACM Sigplan Notices*, 45(6), 363–375. <https://doi.org/10.1145/1809028.1806638>
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide. *SPSS inc*, 9(13), 1–73.
- Changpinyo, S., Sharma, P., Ding, N., & Soricut, R. (2021). Conceptual 12M: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (S. 3558–3568). IEEE. <https://doi.org/10.1109/CVPR46437.2021.00356>
- Chen, Y. C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., Cheng, Y., & Liu, J. (2020). *UNITER: Learning universal image-text representations*. arXiv. <https://doi.org/10.48550/arXiv.1909.11740>
- Clark, A. (2015). *Pillow (pil fork) documentation*. readthedocs.
- Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettmoyer, L., & Stoyanov, V. (2020). *Unsupervised cross-lingual representation learning at scale*. arXiv. <https://doi.org/10.48550/arXiv.1911.02116>
- Dask. (o.D.). *Dask*. <https://docs.dask.org/en/stable/>
- Desai, C. (2022). Understanding Linear Discriminant Analysis for Classification and Dimensionality Reduction. *International Journal of Research and Analytical Review*, 9(4), 202–209.
- Debut, L. [lysandre], & Bourdois, L. [lbourdois] (2022). *xlm-roberta-base* [Sprachmodell]. Hugging Face. <https://huggingface.co/xlm-roberta-base>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *Bert: Pre-training of deep bidirectional transformers for language understanding*. arXiv. <https://doi.org/10.48550/arXiv.1810.04805>
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). *An image is worth 16x16 words: Transformers for image recognition at scale*. arXiv. <https://doi.org/10.48550/arXiv.2010.11929>

- Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2121–2159.
- Elliott, D., Frank, S., Barrault, L., Bougares, F., & Specia, L. (2017). *Findings of the second shared task on multimodal machine translation and multilingual image description*. arXiv. <https://doi.org/10.48550/arXiv.1710.07177>
- Elliott, D., Frank, S., Sima'an, K., & Specia, L. (2016). *Multi30k: Multilingual english-german image descriptions*. arXiv. <https://doi.org/10.48550/arXiv.1605.00459>
- Erhan, D., Courville, A., Bengio, Y., & Vincent, P. (2010). Why does unsupervised pre-training help deep learning? *Proceedings of Machine Learning Research*, 9, 201–208. <http://proceedings.mlr.press/v9/erhan10a.html>
- Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., Baines, M., Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, O., Edunov, S., Auli, M., & Joulin, A. (2021). Beyond english-Centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107), 1–48. <https://jmlr.org/papers/v22/20-1307.html>
- Falcon, W. (2019). *Pytorch lightning* (Version) [Python Library]. GitHub. <https://github.com/PyTorchLightning/pytorch-lightning>
- Gella, S., Sennrich, R., Keller, F., & Lapata, M. (2017). *Image pivoting for learning multilingual multimodal representations*. arXiv. <https://doi.org/10.48550/arXiv.1707.07601>
- Google. (o.D.). Frequently asked questions. <https://research.google.com/colaboratory/faq.html>
- Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (S. 1735–1742). IEEE. <https://doi.org/10.1109/CVPR.2006.100>
- Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (S. 770–778). IEEE. <https://doi.org/10.1109/CVPR.2016.90>.
- He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., & Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In *IEEE/CVF Conference on*

- Computer Vision and Pattern Recognition (CVPR)* (S. 558–567). IEEE.
<https://doi.org/10.1109/CVPR.2019.00065>
- Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the knowledge in a neural network*. arXiv.
<https://doi.org/10.48550/arXiv.1503.02531>
- Hitschler, J., Schamoni, S., & Riezler, S. (2016). *Multimodal pivots for image caption translation*. arXiv. <https://doi.org/10.48550/arXiv.1601.03916>
- Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., ... Sifre, L. (2022). *Training Compute-Optimal Large Language Models*. arXiv.
<https://doi.org/10.48550/arXiv.2203.15556>
- Holz, N. (2022). *What is CRISP DM?* Data Science Process Alliance.
<https://www.datascience-pm.com/crisp-dm-2/>
- Hommel, B. E., Wollang, F. J. M., Kotova, V., Zacher, H., & Schmukle, S. C. (2022). Transformer-based deep neural language modeling for construct-specific automatic item generation. *Psychometrika*, 87(2), 749–772.
<https://doi.org/10.1007/s11336-021-09823-9>
- Huang, P. Y., Chang, X., & Hauptmann, A. (2019). *Multi-head attention with diversity for learning grounded multilingual multimodal representations*. arXiv.
<https://doi.org/10.48550/arXiv.1910.00058>
- Huang, Z., Zeng, Z., Liu, B., Fu, D., & Fu, J. (2020). *Pixel-bert: Aligning image pixels with text by deep multi-modal transformers*. arXiv.
<https://doi.org/10.48550/arXiv.2004.00849>
- Hugging Face. (o.D.). *Autotokenizer* [Software].
https://huggingface.co/docs/transformers/main_classes/tokenizer
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- Jain, A., Guo, M., Srinivasan, K., Chen, T., Kudugunta, S., Jia, C., Yang, Y., & Baldridge, J. (2021). MURAL: Multimodal, multitask representations across languages. In *Findings of the association for computational linguistics: EMNLP 2021* (S. 3449–3463).
<https://doi.org/10.18653/v1/2021.findings-emnlp.293>
- Jia, C., Yang, Y., Xia, Y., Chen, Y. T., Parekh, Z., Pham, H., Le, Q., Sung, Y.-H. Li, Z., & Duerig, T. (2021). Scaling up visual and vision-language representation learning with noisy text supervision. *Proceedings of Machine Learning Research*, 139, 4904–4916.
<http://proceedings.mlr.press/v139/jia21b.html>
- Joshi, A. V. (2020). *Machine learning and artificial intelligence*. Springer Cham.
<https://doi.org/10.1007/978-3-030-26622-6>

- Jurafsky, D., & Martin, J. H. (2021). *Speech and language processing* (3. Aufl. Entwurf). Standford. <https://web.stanford.edu/~jurafsky/slp3/>
- Kaggel. (2022). <https://www.kaggle.com/>
- Kärkkäinen, K., & Joo, J. (2019). *Fairface: Face attribute dataset for balanced race, gender, and age*. arXiv. <https://doi.org/10.48550/arXiv.1908.04913>
- Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). *On large-batch training for deep learning: Generalization gap and sharp minima*. arXiv. <https://doi.org/10.48550/arXiv.1609.04836>
- Khyani, D., Siddhartha, B. S., Niveditha, N. M., & Divya, B. M. (2020). An interpretation of lemmatization and stemming in natural language processing. *Journal of University of Shanghai for Science and Technology*, 22(10), 350–357.
- Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization*. arXiv. <https://doi.org/10.48550/arXiv.1412.6980>
- Klein, D. (2005). CS 294-5: Statistical natural language processing. Berkeley. <https://people.eecs.berkeley.edu/~klein/cs294-5/index.html>
- Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., Bernstein, M. S., & Fei-Fei, L. (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1), 32–73. <https://doi.org/10.1007/s11263-016-0981-7>
- Lawaniya, H. (2020). *Getting Started with PyTorch Lightning*. https://www.researchgate.net/publication/341490081_Getting_Started_with_PyTorch_Lightning
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). *BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*. arXiv. <https://doi.org/10.48550/arXiv.1910.13461>
- Li, F.-F., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop* (S. 178-178). IEEE. <https://doi.org/10.1109/CVPR.2004.383>
- Li, W., Gao, C., Niu, G., Xiao, X., Liu, H., Liu, J., Wu, H., & Wang, H. (2020). *Unimo: Towards unified-modal understanding and generation via cross-modal contrastive learning*. arXiv. <https://doi.org/10.48550/arXiv.2012.15409>
- Li, X., Xu, C., Wang, X., Lan, W., Jia, Z., Yang, G., & Xu, J. (2019). COCO-CN for cross-lingual image tagging, captioning, and retrieval. *IEEE Transactions on Multimedia*, 21(9), 2347–2360. <https://doi.org/10.1109/TMM.2019.2896494>

- Lin, Z., Feng, M., Santos, C. N. D., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). *A structured self-attentive sentence embedding*. arXiv. <https://doi.org/10.48550/arXiv.1703.03130>
- Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (S. 740–755). Springer. https://doi.org/10.1007/978-3-319-10602-1_48
- Lin, J., Men, R., Yang, A., Zhou, C., Ding, M., Zhang, Y., Wang, P., Wang, A., Jiang, Xianyan Jia, Jie Zhang, Jianwei Zhang, Xu Zou, Zhikang Li, Xiaodong Deng, Jie Liu, L., Xue, J., Zhou, H., Ma, J., ... Yang, H. (2021). *M6: A chinese multimodal pretrainer*. arXiv. <https://doi.org/10.48550/arXiv.2103.00823>
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A robustly optimized BERT pretraining approach*. arXiv. <https://doi.org/10.48550/arXiv.1907.11692>
- Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., & Shazeer, N. (2018). *Generating wikipedia by summarizing long sequences*. arXiv. <https://doi.org/10.48550/arXiv.1801.10198>
- Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). *ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks*. arXiv. <https://doi.org/10.48550/arXiv.1908.02265>
- Manning, C. D. (2008). *Introduction to information retrieval*. Syngress Publishing.
- McEwan, T., & Weerts, B. (2007). ALT text and basic accessibility. In D. Ramduny-Ellis & D. Rachovides (Hrsg.), *Proceedings of the 21st BCS HCI Group conference* (Bd. 2, S.71–74). British Computer Society.
- McKinney, W. (2010). Data structures for statistical computing in python. In S. van der Walt & K. J. Millmann (Hrsg.), *Proceedings of the 9th Python in Science Conference* (S. 51–56).
- McKinney, W., (2013). *Python for data analysis*. O'Reilly Media.
- Messina, N., Amato, G., Esuli, A., Falchi, F., Gennaro, C., & Marchand-Maillet, S. (2021). Fine-grained visual textual alignment for cross-modal retrieval using transformer encoders. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 17(4), 1–23. <https://doi.org/10.1145/3451390>
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv. <https://doi.org/10.48550/arXiv.1301.3781>
- Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heinz, I., & Roth, D. (2021). *Recent advances in natural language processing via large*

- pre-trained language models: A survey.* arXiv.
<https://doi.org/10.48550/arXiv.2111.01243>
- Miyazaki, T., & Shimizu, N. (2016). Cross-lingual image caption generation. In *Proceedings of the 54th annual meeting of the association for computational linguistics* (S. 1780–1790). Association for Computational Linguistics.
- Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5), 544–551. <https://doi.org/10.1136/amiajnl-2011-000464>
- Nakamura, A., & Harada, T. (2019). *Revisiting fine-tuning for few-shot learning*. arXiv.
<https://doi.org/10.48550/arXiv.1910.00216>
- Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Ng, A. Y. (2011). Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning* (S.1–8). https://openreview.net/forum?id=Hk4OO3W_bS
- OpenAI. (2021). *CLIP*. Github.
https://github.com/openai/CLIP/blob/main/notebooks/Interacting_with_CLIP.ipynb
- OpenAI. (2021). *CLIP: Connecting Text and Images*. <https://openai.com/blog/clip/>
- Parkhi, O. M., Vedaldi, A., Zisserman, A., & Jawahar, C. V. (2012). Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition* (S. 3498–3505). IEEE.
<https://doi.org/10.1109/CVPR.2012.6248092>
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 1–12.
<https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html>
- Paulus, R., Xiong, C., & Socher, R. (2017). *A deep reinforced model for abstractive summarization*. arXiv. <https://doi.org/10.48550/arXiv.1705.04304>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Burcher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of NAACL-HLT 2018* (S. 2227–2237). Association for Computational Linguistics.
<https://paperswithcode.com/paper/deep-contextualized-word-representations>

- Qi, D., Su, L., Song, J., Cui, E., Bharti, T., & Sacheti, A. (2020). *ImageBERT: Cross-modal pre-training with large-scale weak-supervised image-text data*. arXiv. <https://doi.org/10.48550/arXiv.2001.07966>
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. *Proceedings of Machine Learning Research*, 139, 8748–8763. <https://proceedings.mlr.press/v139/radford21a.html>
- Radford, A., & Narasimhan, K. (2018). *Improving language understanding by generative pre-training*. <https://www.semanticscholar.org/paper/Improving-Language-Understanding-by-Generative-Radford-Narasimhan/cd18800a0fe0b668a1cc19f2ec95b5003d0a5035>
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners*. <https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe>
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140), 1–67. <http://jmlr.org/papers/v21/20-074.html>
- Rajendran, J., Khapra, M. M., Chandar, S., & Ravindran, B. (2016). *Bridge correlational neural networks for multilingual multimodal representation learning*. arXiv. <https://doi.org/10.48550/arXiv.1510.03519>
- Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-shot text-to-image generation. In International Conference on Machine Learning. *Proceedings of Machine Learning Research*, 139, 8821–8831. <http://proceedings.mlr.press/v139/ramesh21a.html?ref=https://githubhelp.com>
- Reimers, N. (o.D.). *Pretrained models*. SBERT.net. https://www.sbert.net/docs/pretrained_models.html
- Reimers, N., & Gurevych, I. (2020). *Making monolingual sentence embeddings multilingual using knowledge distillation*. arXiv. <https://doi.org/10.48550/arXiv.2004.09813>
- Ridnik, T., Ben-Baruch, E., Noy, A., & Zelnik-Manor, L. (2021). *ImageNet-21K pretraining for the masses*. arXiv. <https://doi.org/10.48550/arXiv.2104.10972>
- Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling. In *Proceedings of the 14th python in science conference* (Bd. 130, S. 126–132). SciPy.
- Salama, K. (2021). *Natural language image search with a Dual Encoder [Code]*. Keras. https://keras.io/examples/nlp/nl_image_search/

- Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2020). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. arXiv. <https://doi.org/10.48550/arXiv.1910.01108>
- See, A., Pappu, A., Saxena, R., Yerukola, A., & Manning, C. D. (2019). *Do massively pretrained language models make better storytellers?* arXiv. <https://doi.org/10.48550/arXiv.1909.10705>
- Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In *54th Annual Meeting of the Association for Computational Linguistics* (S. 1715–1725). Association for Computational Linguistics. <https://doi.org/10.18653/v1/P16-1162>
- Sharma, M., Gogineni, A., & Ramakrishnan, N. (2022). *Innovations in neural data-to-text generation*. arXiv. <https://doi.org/10.48550/arXiv.2207.12571>
- Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)* (S. 464–472). IEEE. <https://doi.org/10.1109/WACV.2017.58>
- Srinivasan, K., Raman, K., Chen, J., Bendersky, M., & Najork, M. (2021). Wit: Wikipedia-based image text dataset for multimodal multilingual machine learning. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval* (S. 2443–2449). Association for Computing Machinery. <https://doi.org/10.1145/3404835.3463257>
- Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Müller, K. R. (2021). Towards CRISP-ML (Q): A machine learning process model with quality assurance methodology. *Machine Learning and Knowledge Extraction*, 3(2), 392–413. <https://doi.org/10.3390/make3020020>
- Su, W., Zhu, X., Cao, Y., Li, B., Lu, L., Wei, F., & Dai, J. (2019). *VL-BERT: Pre-training of generic visual-linguistic representations*. arXiv. <https://doi.org/10.48550/arXiv.1908.08530>
- Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *Proceedings of Machine Learning Research*, 97, 6105–6114. <http://proceedings.mlr.press/v97/tan19a.html>
- The pandas development team. (2022). pandas-dev/pandas: Pandas (Version v1.4.1) [Software Library]. Zenodo. <https://doi.org/10.5281/zenodo.3509134>
- Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., & Li, L. J. (2016). YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2), 64–73. <https://doi.org/10.1145/2812802>
- Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: *Neural Networks for Machine Learning*, 4, 26–31.

- Torfi, A., Shirvani, R. A., Keneshloo, Y., Tavaf, N., & Fox, E. A. (2020). *Natural language processing advancements by deep learning: A survey*. arXiv. <https://doi.org/10.48550/arXiv.2003.01200>
- Turovsky, B. (2016). *Found in translation: More accurate, fluent sentences in Google Translate*. Google The Keyword. <https://blog.google/products/translate/found-translation-more-accurate-fluent-sentences-google-translate/>
- Umesh, P. (2012). Image processing in python. *CSI Communications*, 23.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30, 1–11. <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fb053c1c4a845aa-Abstract.html>
- Wang, J., Liu, Y., & Wang, X. E. (2022). *Assessing multilingual fairness in pre-trained multimodal representations*. arXiv. <https://doi.org/10.48550/arXiv.2106.06683>
- Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, 6(60), Artikel 3021. <https://doi.org/10.21105/joss.03021>
- Wenzek, G., Lachaux, M. A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., & Grave, E. (2019). CCNET: Extracting high quality monolingual datasets from web crawl data. arXiv. <https://doi.org/10.48550/arXiv.1911.00359>
- Wikimedia Foundation. (2021). *Wikipedia - image/caption matching: Retrieve captions based on images*. Kaggle. <https://www.kaggle.com/c/wikipedia-image-caption/overview>
- Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining* (Bd. 1, S. 29–39). <http://www.cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf>
- Xu, P., Zhu, X., & Clifton, D. A. (2022). *Multimodal learning with transformers: a survey*. arXiv. <https://doi.org/10.48550/arXiv.2206.06488>
- Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2021). mT5: A massively multilingual pre-trained text-to-text transformer. arXiv. <https://doi.org/10.48550/arXiv.2010.11934>
- Young, P., Lai, A., Hodosh, M., & Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2, 67–78. https://doi.org/10.1162/tacl_a_00166
- Zhang, R. (2019). Making convolutional networks shift-invariant again. arXiv. <https://doi.org/10.48550/arXiv.1904.11486>

- Zhang, Y., Jiang, H., Miura, Y., Manning, C. D., & Langlotz, C. P. (2022). *Contrastive learning of medical visual representations from paired images and text*. arXiv. <https://doi.org/10.48550/arXiv.2010.00747>
- Zhang, Y., Kang, B., Hooi, B., Yan, S., & Feng, J. (2021). *Deep long-tailed learning: A survey*. arXiv. <https://doi.org/10.48550/arXiv.2110.04596>
- Zhang, T., Wu, F., Katiyar, A., Weinberger, K. Q., & Artzi, Y. (2021). *Revisiting few-sample BERT fine-tuning*. arXiv. <https://doi.org/10.48550/arXiv.2006.05987>
- Zhou, K., Yang, J., Loy, C. C., & Liu, Z. (2022). Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9), 2337–2348. <https://doi.org/10.1007/s11263-022-01653-1>
- Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision* (S. 19–27). IEEE. <https://doi.org/10.1109/ICCV.2015.11>

Selbstständigkeitserklärung



Name: Rogall	<u>Bitte beachten:</u>
Vorname: Marten	1. Bitte binden Sie dieses Blatt am Ende Ihrer Arbeit ein.
geb. am: 21.07.1990	
Matr.-Nr.: 461832	

Selbstständigkeitserklärung*

Ich erkläre gegenüber der Technischen Universität Chemnitz, dass ich die vorliegende **Masterarbeit** selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Datum: 10.11.2022

Unterschrift: 

* Statement of Authorship

I hereby certify to the Technische Universität Chemnitz that this thesis is all my own work and uses no external material other than that acknowledged in the text.

This work contains no plagiarism and all sentences or passages directly quoted from other people's work or including content derived from such work have been specifically credited to the authors and sources.

This paper has neither been submitted in the same or a similar form to any other examiner nor for the award of any other degree, nor has it previously been published.