INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine Universität Düsseldorf

Generation von Multi-Document Summarizations mittels Variational Auto Encoder Transformern

Lionel Schockenhoff

Masterarbeit

Beginn der Arbeit:	02. September 2020
Abgabe der Arbeit:	29. Februar 2021
Gutachter:	Prof. Dr. Stefan Conrad
	Prof. Dr. Martin Mauve

Erklärung

Hiermit versichere ich, dass ich diese Masterarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 29. Februar 2021

Lionel Schockenhoff

Zusammenfassung

Im Bereich des Natural Language Processing erzielten generative Modelle wie Variational Autoencoder große Fortschritte und ermöglichen die gesteuerte Generierung von Textsequenzen. Insbesondere im Bereich der Multi-Document Summarization bietet sich die Verwendung von Variational Autoencodern an, um gezielt Dokumente im Latentvektorraum miteinander zu verknüpfen und präzise Zusammenfassungen dieser zu generieren. Multi-Document Summarization bezeichnet das Zusammenfassen mehrerer Dokumente zu einem repräsentativen Dokument, welches Anwendern einen schnellen guten Überblick ermöglicht.

Im Bereich des E-Commerce und von Online-Vergleichsportalen entstehen eine große Anzahl an Rezensionen zu zahlreichen Produkten und Dienstleistungen. Das Ziel dieser Masterarbeit ist das automatisierte, unüberwachte Zusammenfassen von mehreren Rezensionen eines Produktes oder einer Dienstleistung unter Verwendung von Variational Autoencodern zu einer kongruenten repräsentativen Rezension. Es werden zwei unterschiedliche Variational Autoencoder Modelle verwendet, welche mittels eines Attributmodells optimiert werden. Durch das Attributmodell werden bei der Generierung die Token Wahrscheinlichkeiten restrukturiert, um weitaus bessere Ergebnisse zu erzielen.

In der Evaluation zeigen die entwickelten Modelle eine hervorragende Performance bei der Zusammenfassung und erreichen State-of-the-Art Ergebnisse auf unterschiedlichen Metriken. Es lassen sich Rezensionen zu unterschiedlichen Themengebieten automatisiert mittels Variational Autoencoder und Attributmodell zusammenfassen.

Inhaltsverzeichnis

1 Einleitung

Das automatisierte Zusammenfassen von Dokumenten ist im Bereich des Natural Language Processing (NLP) eine große Herausforderung. Natural Language Processing ist ein Unterbereich der künstlichen Intelligenz, der sich mit dem maschinellen Verarbeiten von natürlicher Sprache auseinandersetzt. Aufgrund von neuen Methoden und immer größeren, komplexeren Netzwerken lassen sich hervorragende Ergebnisse in diversen NLP Aufgabenbereichen erzielen. Insbesondere lassen sich große, kostspielig vortrainierte Modelle mit vielen Parametern durch Transfer Learning in diversen speziellen Aufgabenbereichen einsetzen.

Generative Pre-trained Transformers 2 (GPT-2) erzielte unter Verwendung von Transformern großartige Ergebnisse bei der Textgenerierung, unter anderem auch bei der abstraktiven Zusammenfassung von Texten. Die abstraktive Textzusammenfassung bezeichnet das Zusammenfassen von Texten zu einem kurzen, präzisen Text.

1.1 Motivation

Im Bereich des E-Commerce und von Online-Vergleichsportalen entstehen eine große Anzahl an Rezensionen zu unterschiedlichen Produkten, Dienstleistungen und Anbietern. Es ist für den Anwender eine Herausforderung sich einen repräsentativen Überblick über die einzelnen Rezensionen zu verschaffen. Ein gängiges Bewertungssystem ist das Berechnen des arithmetischen Mittels über die einzelnen Scores aller Bewertungen. Ein zusammengefasster Bewertungsscore ist nicht ausreichend repräsentativ für viele Produkte und stellt nicht spezifische Funktionen der einzelnen Produkte dar. Hier bietet es sich an, ein Produkt durch eine generierte Rezension zu repräsentieren, die die anderen Rezensionen inkludiert und auf die verschiedenen wichtigen Aspekte der Produkte eingeht. So kann ein Anwender sich zeitsparend einen ersten groben Überblick über bestimmte Produkte verschaffen.

1.2 Ziel und Aufbau der Arbeit

Ziel dieser Masterarbeit ist das unbeaufsichtigte abstraktive Zusammenfassen von mehreren Produktrezensionen mittels Variational Autoencodern zu einer repräsentativen Rezension. Die generierten Zusammenfassungen sollten möglichst viele Aspekte der ursprünglichen Rezension aufgreifen, konsistent in ihrem Inhalt sein und einen guten Gesamtüberblick über die Rezensionen zu einem Produkt oder Dienstleistungen (zum Beispiel Restaurants) geben. Als Datengrundlage gelten hier der Amazon-Review- und Yelp-Review-Datensatz, die zu Produkten und Dienstleistungen mehrere unterschiedliche diverse Rezensionen liefern.

Zunächst werden in Kapitel 2 die verwendeten Grundlagen zu Textzusammenfassung, Deep Learning, Word Embeddings und Sequence To Sequence Modellen erläutert. Anschließend werden in Kapitel 3 die Transformermodelle Bidirectional Encoder Representations from Transformers (BERT) und GPT-2 erklärt, wobei detailliert auf die besonderen Merkmale wie Transformer Architekturen und Attention Layer eingegangen wird.

Der Attention Layer Aufbau ist insbesondere wichtig für die spätere Injektion der adaptierten Latentvektoren.

In Kapitel 4 wird der Aufbau und die mathematischen Details von Variational Autoencodern vermittelt. Hier werden die zwei unterschiedlichen Variational Autoencoder Optimus und BIMEANVAE hergeleitet, die im weiteren Verlauf die Zusammenfassungen generieren. Optimus basiert auf den beiden bekannten Sprachmodellen BERT und GPT-2. BIMEANVAE besteht aus einem bidirektionalem Long Short Term Memory (LSTM)-Encoder und einem LSTM-Decoder.

Im Anschluss werden in Kapitel 5 die beiden Datensätze Amazon und Yelp vorgestellt, wobei die unterschiedlichen Aspekte und Herausforderungen der Datensätze dargestellt werden.

Danach werden in Kapitel 6 mehrere aktuelle Verfahren zur Multi-Review Summarization vorgestellt. Convex Aggregation for Opinion Summarization (COOP) ist eins dieser Verfahren, welches unterschiedliche Kombinationen von einzelnen Latentvektoren untersucht.

In Kapitel 7 wird das vorgestellte COOP Verfahren durch ein Attributmodell weiter verbessert. Hierzu wird bei den Variational Autoencodern der Generationsprozess in Bezug auf ein Attributmodell optimiert. Des Weiteren wird in diesem Kapitel eine neue Rankingfunktion zur Auswahl der Rezensionen eingeführt und die Hyperparameter für die Evaluation bestimmt.

Abschließend werden die konstruierten Modelle in Kapitel 8 auf den Datensätzen evaluiert. Ebenfalls werden einzelne generierte Rezensionen untereinander verglichen. In der abschließenden Zusammenfassung in Kapitel 9 wird ein Fazit und ein Ausblick auf weitere Forschungsgebiete gegeben.

2 Grundlagen

In diesem Kapitel werden die Grundlagen zu den verwendeten Technologien vorgestellt. Zunächst wird die in dieser Masterarbeit untersuchte Aufgabe des abstrakten Zusammenfassens von mehreren Dokumenten erläutert. Anschließend werden die Grundlagen zu neuronalen Netzen und insbesondere zu Deep Learning zusammengefasst dargestellt. Hier wird ebenfalls die Struktur von Sequence-To-Sequence Modellen und LSTM-Zellen erläutert.

Im folgenden Kapitel 3 wird anschließend aus den Grundlagen die verwendete Transformer Architektur, BERT und GPT-2 konkretisiert.

2.1 Textzusammenfassung

Das automatisierte Zusammenfassen von Texten ist ein Teilgebiet des NLP, welches sich mit dem Zusammenfassen von langen Texten zu einem kongruenten, kürzeren Text unter Beibehaltung von wichtigen Informationen befasst. Durch die zunehmenden Datenmengen wird automatisierte Textzusammenfassung immer relevanter, um akkurat aggregierte Zusammenfassungen und Überblicke zu geben. Automatisierte Textzusammenfassung lässt sich zum Beispiel bei der Generierung von Kurzzusammenfassungen zu Dokumenten verwenden.

Grundsätzlich wird beim automatisierten Zusammenfassen von Texten zwischen den extraktiven und den abstraktiven Methoden unterschieden.

2.1.1 Extraktive Textzusammenfassung

Extraktive Textzusammenfassung ist das Identifizieren und anschließende Extrahieren von wichtigen Phrasen oder Sätzen aus dem Ursprungstext. Hierbei werden die entsprechend ausgewählten Phrasen in der Zusammenfassung genauso wie sie im Ursprungstext vorkommen übernommen. Die zu extrahierenden Phrasen oder Sätze werden mithilfe einer Scoringfunktion gefunden und später aneinander gereiht. Hierzu gibt es unterschiedliche Methoden und Metriken.

2.1.2 Abstraktive Textzusammenfassung

Abstraktive Textzusammenfassung versucht durch Interpretation und Verständnis des Ursprungtexts eine kurze, kongruente Zusammenfassung zu produzieren. Die Zusammenfassung soll alle wichtigen Informationen enthalten und als zusammenhängender flüssiger Text erzeugt werden. Ein kongruenter Text wird erzeugt, da das Sprachmodell frei Sätze produzieren kann und nicht an vorher vorgegebene Sätze oder Phrasen gebunden ist, wie bei der extraktiven Zusammenfassung.

Große Fortschritte im Bereich der abstraktiven Textzusammenfassung ergaben sich in den letzten Jahren durch Sequence-To-Sequence Modelle. Diese encodieren den Eingabetext in eine Übergangsrepräsentation und generieren aus dieser durch Decodierung eine

Ausgangsrepräsentation.

2.1.3 Multi-Document Textzusammenfassung

Eine große Herausforderung ist das Zusammenfassen von mehreren Dokumenten über dasselbe Thema zu einem einzigen Dokument. Die entstehende Zusammenfassung soll Anwendern einen schnellen, umfassenden Überblick über eine große Anzahl an Dokumenten bieten. Die unterschiedlichen Dokumente enthalten diverse Informationen, die nicht immer im Ergebnis und Vokabular deckungsgleich sind. Somit ergibt sich die Herausforderung, unterschiedliche Perspektiven in den jeweiligen Dokumenten zusammenfassend zu repräsentieren. Da sich unterschiedliche Standpunkte nur unzureichend mittels extraktiven Methoden darstellen lassen, bieten sich bei der Multi-Document Textzusammenfassung die Verwendung abstraktiver Methoden an. Damit kann eine kongruente Zusammenfassung generiert werden, die die unterschiedlichen Aspekte der Dokumente darstellt.

In dieser Masterarbeit werden unterschiedliche Rezensionen zu Produkten und Dienstleistungen zusammengefasst. Insbesondere Rezensionen unterscheiden sich stark in ihren Nutzerperspektiven und können positiv, negativ oder neutral sein und auf sehr spezifische Eigenschaften der entsprechenden Produkte eingehen. Es ist eine große Herausforderung aus einer Menge aus Produktrezensionen eine repräsentative zusammenfassende Rezension zu produzieren, die klar strukturiert, kongruent, verständlich und Zielgruppenbezogen die entsprechenden Inhalte wiedergibt.

2.2 Deep Learning

Deep Learning ist ein Teilbereich des Machine Learning, bei dem nach dem Vorbild für das menschliche Gehirn neuronale Netze verwendet werden. Neuronale Netze werden unter anderem beim Natural Language Processing eingesetzt. Die neuronalen Netze bestehen aus mehreren Layern, die sequentiell den Output des vorherigen Layers weiterverarbeiten. Das Ziel von Deep Learning ist, durch Training der neuronalen Netze, Repräsentationen beziehungsweise Approximationen für Funktionen in den Daten zu finden. Zum Trainieren dieser Netze wird oft Backpropagation, ein Verfahren zur Berechnung der Gradienten in neuronalen Netzen und entsprechender Anpassung der Gewichte verwendet.

2.3 Word Embeddings

Word Embeddings werden im Natural Language Processing verwendet, um Wörter mit aussagekräftigen Vektoren zu repräsentieren. Hochdimensionale Objekte wie Wörter lassen sich mittels Word Embeddings in einen niedrigdimensionalen Raum darstellen und behalten dabei ihre semantischen Relationen bei. Auf diesen Vektoren lassen sich unterschiedliche arithmetische Operationen ausführen.

Bekannte kontextunabhängige Word Embedding Verfahren sind zum Beispiel word2vec (Mikolov et al., 2013) und GloVe (Global Vectors for Word Representation) (Pennington

et al., 2014). Diese Verfahren berechnen für jedes Wort einen eindeutigen Vektor, der alle unterschiedlichen Eigenschaften dieses Wortes enthält. Die errechneten Vektoren sind kontextunabhängig und für ein Wort wird stets der gleiche Vektor verwendet. Word2vec erlernt die entsprechenden Vektorrepräsentationen mittels eines SkipGram neuronalen Netzes (Mikolov et al., 2013), GloVe hingegen über die nicht Nulleinträge einer Co-occurrence Matrix von Wörtern untereinander (Pennington et al., 2014).

Kontextabhängige Verfahren, wie zum Beispiel ELMO (Peters et al., 2018) oder BERT (Devlin et al., 2018), generieren kontextabhängige Vektoren für Wörter und berücksichtigen dabei das Umfeld in dem das einzelne Wort auftritt. Bei diesen Verfahren wird zur Bestimmung eines Word Embeddings der gesamte Satz benötigt, um die erlernten kontextspezifischen Eigenarten für die Einbettung zu berücksichtigen.

Verfahren wie zum Beispiel BERT nutzen besondere Tokenisierungsmethoden, die es erlauben einzelne Wörter durch mehrere Tokens zu repräsentieren. Dieses Splittingverfahren von Wörtern in Subtokens ist als WordPiece Verfahren (Wu et al., 2016) für BERT und als BytePairEncoding (Sennrich et al., 2015) für GPT-2 bekannt. Durch das Splitten in Subtokens und Erlernen von Einbettungen für diese lässt sich ein kleines Wörterbuch erstellen. Da sich die Wörter stets in Subtokens zerlegen lassen, können Out-of-Vocabulary-Fehler vermieden werden.

2.4 Sequence-To-Sequence Modelle

Sequence-To-Sequence Modelle sind eine Gruppe von Deep Leaning Modellen zur Sprachverarbeitung, die eine Textsequenz $(x_1, x_2, \ldots, x_n)$ bestehend aus den Tokens x_i in eine andere Textsequenz $(y_1, y_2, \ldots, y_n)$ überführen (Sutskever et al., 2014). Die Besonderheit bei diesem Modell liegt darin, dass die Länge der Eingabesequenz und der Ausgabesequenz unterschiedlich sein kann. Die am meisten verwendeten Sequence-To-Sequence Modelle bestehen aus einer Encoder/Decoder Architektur. Der Encoder und Decoder besteht jeweils aus sequentiellen Rekurrenten Neuronalen Netzen (RNN), die jeweils als Eingabe ein Token und den Hidden-State des vorherigen RNNs erhalten. Somit summieren sich innerhalb des Encoders die Hidden-States auf und werden abschließend durch einen Hidden-State Vektor repräsentiert, der die Informationen aus der Eingabe enthält. Der abschließende Hidden-State Vektor lässt sich wie folgt iterativ bestimmen, wobei f für die entsprechende RNN-Zelle und W für die Gewichtsmatrizen steht:

$$h_t = f(W^{hh} h_{t-1} + W^{hx} x_t)$$

Oft werden als RNN-Zellen Long Short Term Memory (LSTM)- oder Gated Recurrent Unit (GRU)-Zellen verwendet. Der Decoder ist ebenfalls ein RNN, welcher aus dem Hidden-State Vektor des Encoders eine Ausgabesequenz generiert.

Um die Performance von Sequence-To-Sequence Modellen weiter zu optimieren, werden in Kapitel 3.2 Attention Modelle eingeführt.

2.5 Long Short Term Memory

Long **S**hort **T**erm **M**emory (LSTM) Netze sind eine Untergruppe der Rekurrenten Neuronalen Netze (RNN), die es ermöglichen, Langzeitbeziehungen in Daten zu erlernen (Hochreiter und Schmidhuber, 1997). Rekurrente Neuronale Netze zeichnen sich durch ihre rekurrenten Verbindungen innerhalb desselben Layers aus, wodurch die nächste RNN-Zelle die vorherigen Informationen weiterverarbeiten kann. Ein häufiges Problem ist das Modellieren von Langzeitbeziehungen mittels RNNs. LSTMs sind speziell darauf ausgelegt diese Langzeitbeziehungen abzubilden, indem sie stets einen Zellzustand mit übergeben. Eine LSTM-Zelle hat nun die Möglichkeit diesen Zellstatus zu manipulieren, indem Informationen hinzugefügt oder gelöscht werden können. Diese Manipulationen des Zellstatus werden durch drei Gates ermöglicht:

1. Forget Gate - ermöglicht der Zelle alte Werte zu vergessen.
2. Input Gate - steuert, zu welcher Gewichtung neue Eingabewerte in dem Hidden-State gespeichert werden.
3. Output Gate - steuert, welcher Teil des Hidden-States in die folgende Zelle propagiert wird.

Durch die Anpassung des Hidden-States in den einzelnen Zellen können relevante Informationen anhand des Hidden-States durch die Zellen geleitet werden, wodurch lange Beziehungen in den Textsequenzen modelliert werden können (Olah, o. D.). Das Propagieren durch eine Reihe von LSTM-Zellen sowie der Einfluss der einzelnen Gates ist in Abbildung 1 dargestellt.

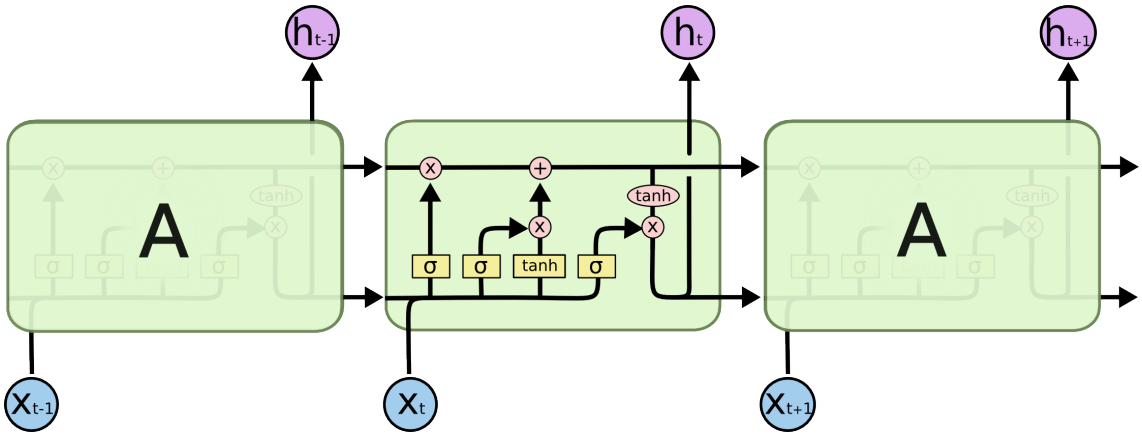

Abbildung 1: Reihe von LSTM-Zellen mit entsprechenden Gates (Olah, o. D.)

3 Transformer

Transformer sind eine spezifische Encoder-Decoder Deep Learning Architektur, die insbesondere im Natural Language Processing eingesetzt wird. Basierend auf Attention Layern ermöglichen Transformer die Leistung bei vielen NLP Anwendungen stark zu steigern. Ein großer Vorteil von Transformermodellen ist die Parallelisierbarkeit. Hierdurch lassen sich extrem große Sprachmodelle, wie zum Beispiel BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) oder GPT-2 (**G**enerative **P**re-trained **T**ransformer-2) auf Basis von Transformern trainieren. Beide Modelle erzielen in den unterschiedlichsten Benchmarks hervorragende Ergebnisse (Devlin et al., 2018).

Transformermodelle sind Sequence-To-Sequence Modelle, die aus einem Encoder und einem Decoder bestehen. Die vortrainierten Sprachmodelle BERT und GPT-2 lassen sich auf spezielle Aufgabenbereiche finetunen.

3.1 Transformer Encoder / Decoder

Transformer verwenden eine Encoder - Decoder Struktur. Der Encoder wandelt eine Eingabesequenz $(x_1, \ldots, x_n)$ in eine kontinuierliche Übergangsrepräsentation $z = (z_1, \ldots, z_n)$ um. Mit dieser Übergangsrepräsentation z kann anschließlich vom Decoder eine Ausgabesequenz $(y_1, \ldots, y_n)$ generiert werden. Der Transformer Decoder generiert Textsequenzen autoregressiv und bezieht somit für die Berechnung des nächsten Ausgabetokens die vorherigen Tokens als Eingabe mit ein.

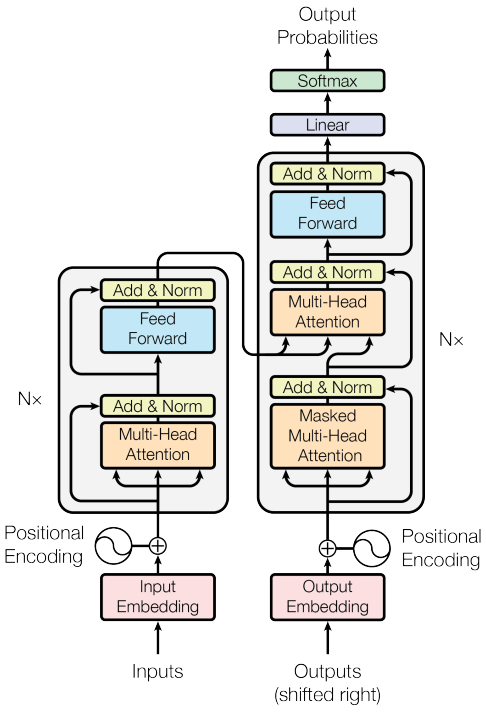

Abbildung 2: Transformer Encoder (links) und Transformer Decoder (rechts) (Vaswani et al., 2017)

Abbildung 2 stellt die Architektur und den Datenfluss durch die verschiedenen Layer von Transformer Encodern und Transformer Decodern dar. Grundsätzlich bestehen sowohl der Encoder sowie auch der Decoder aus aufeinanderfolgenden Multi-Head Attention-Layern gefolgt von Fully-Connected Feed-Forward-Netzen (Vaswani et al., 2017).

Die Transformer Encoder Blöcke verwenden Self-Attention-Layer und beziehen damit ihre Eingabe vollständig aus der Ausgabe des vorherigen Layers. Die meisten Modelle verwenden mehrere Transformer Encoder Blöcke und Transformer Decoder Blöcke sequentiell hintereinander.

Nach Encodierung der Eingabesequenzen durch den Transformer Encoder wird die Ausgabe in Attention Key- und Value-Vektoren umgewandelt. Diese Attentionvektoren werden vom Transformer Decoder in den Encoder-Decoder Attention Layern verwendet. Im Gegensatz zu den Self-Attention-Layern der Encoder verwendet der Decoder hier die Ausgabe des Encoders im Attention-Layer in den Berechnungen mit, um sich auf die korrekten Stellen im Input bei der Generierung der finalen Ausgabesequenz zu fokussieren.

3.2 Attention-Layer

Durch die Attention-Layer von Transformermodellen lassen sich relevante Informationen erkennen und das Modell kann sich auf diese fokussieren. Der Attention-Layer innerhalb des Transformers berechnet die Relevanz von Values zu bestimmten Keys und Queries.

Falls Attention-Layer ihre gesamte Eingabe wie bei Transformer Encodern aus einer Datenquelle beziehen, handelt es sich um Self-Attention-Layer. Bei Self-Attention-Layern werden die Query- (Q), Key- (K) und Value- (V) Vektoren aus dem Eingabevektor (X) durch Multiplikation mit erlernten Gewichtsmatrizen (W) generiert (Vaswani et al., 2017).

$$Q = X \times W^Q, K = X \times W^K, V = X \times W^V$$

Durch ein Dot-Product zwischen Query- und Key-Vektoren der entsprechenden Eingabewörtern mit anschließender Softmax Normalisierung lässt sich ein Score errechnen, der den Fokus im Valuevektor auf das jeweilige Wort gewichtet (Vaswani et al., 2017).

$$Attention(Q, K, V) = Softmax(\frac{Q \times K^T}{\sqrt{d_k}}) \times V \tag{1}$$

In Gleichung 1 werden die Berechnungen an den Matrizen Q für die Queries, K für die Keys und V für die entsprechenden Values durchgeführt. Als Skalierungsfaktor wird $\sqrt{dk}$, die Dimension der Key Vektoren verwendet, um einen stabilen Gradienten zu erhalten.

Attention Funktionen lassen sich parallel ausführen. Diese Eigenschaft nutzen Transformermodelle aus und verwenden Multi-Head Attention Layer, um mehrere Attention Funktionen parallel auszuführen. Es werden parallel h voneinander unabhängige Attention Layer ausgeführt, die ihre einzelnen Ergebnisse für die Weiterverarbeitung aneinanderfügen und linear in die gewünschte Dimension transformieren.

$$\begin{aligned} MultiHeadAttention(Q, K, V) &= [head_1, \ldots, head_h] \times W^O \\ \text{mit } head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2}$$

In Gleichung 2 wird deutlich, dass die unterschiedlichen Attention-Heads voneinander unabhängig sind. Hierdurch können sich die einzelnen Attention-Heads auf unterschiedliche Merkmale in den Daten fokussieren.

3.3 Bidirectional Encoder Representations from Transformers

BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) ist ein von Google (Devlin et al., 2018) entwickeltes Encoder-Sprachmodell, welches sich durch die bidirektionale kontextuelle Einbettung von Worten auszeichnet. Bei der Veröffentlichung von BERT konnten in vielen NLP Benchmarks Bestleistungen erzielt werden (Devlin et al., 2018).

BERT zeichnet sich ebenfalls durch die Möglichkeit des Transfer Learnings aus. Es existieren vortrainierte BERT-Sprachmodelle, die jeweils auf spezifische Aufgaben finegetuned werden können.

Die Modellarchitektur von BERT sind sequentielle Transformerencoderlayer. Das Basis BERT-Modell verwendet 12 Transformerencoderblöcke mit jeweils 12 Multi-Attention-Heads und einer Hidden-Size von 768. BERT wird mit den beiden Pretrainings Aufgaben Masked Language Modeling und Next Sentence Prediction vortrainiert. Durch den besonderen Trainingsprozess betrachtet BERT Sequenzen von links und rechts gleichzeitig also bidirektional. Einige andere Verfahren betrachten Sequenzen nur einseitig beziehungsweise kombinieren zwei einseitige Betrachtungen wie zum Beispiel ELMO (Peters et al., 2018) und haben demnach keine tiefe bidirektionale Repräsentation der Sequenzen.

Beim Masked Language Modeling werden die bidirektionalen Repräsentationen trainiert, indem zufällig 15% der Worttokens in einer Sequenz gegen ein **[MASK]** Token ersetzt werden. Das Trainingsziel ist, die maskierten Tokens korrekt vorherzusagen. Zum Verständnis der Zusammenhänge zwischen den Sätzen wird beim Next Sentence Prediction Pretrainingsschritt vorhergesagt, ob ein zu 50% zufällig gewählter Satz B auf Satz A folgt (Devlin et al., 2018).

3.4 Generative Pre-trained Transformer-2

GPT-2 (**G**enerative **P**re-trained **T**ransformer-2) ist ein von OpenAI (Radford et al., 2019) veröffentlichtes autoregressives Sprachmodell auf Basis von Transformerdecodern. Autoregressive Sprachmodelle generieren Textsequenzen, indem sie einzeln Tokens generieren und diese an die Eingabesequenz anhängen. Die neue verlängerte Textsequenz wird erneut als Eingabe zur Generierung genutzt. Als generatives Modell erzielt es hervorragende Ergebnisse beim Generieren von kongruenten Textpassagen.

GPT-2 small besteht aus 12 sequentiellen Transformerdecodern. Um die autoregressiven Eigenschaften beizubehalten, verwendet GPT-2 Masked Self-Attention Layer. Durch die Masked Self-Attention Layer kann das GPT-2 Modell beim Training im Gegensatz zu BERT lediglich die bereits bekannten Tokens betrachten.

Zur Generierung von Textpassagen kann GPT-2 unkontrolliert Samples aus einer leeren Eingabesequenz generieren. Kontrolliert werden kann die Generierung durch das Festlegen einer spezifischen Starteingabe an die GPT-2 generierte Tokens anfügt. Weiterhin kann GPT-2 auf einen spezifischen Datensatz finegetuned werden, um so die zu erwartende Ausgabe zu beeinflussen. Während der Generierung selbst lässt sich die Generierung allerdings nicht kontrollieren.

GPT-2 verwendet zur Tokenisierung das BytePairEncoding Verfahren (Sennrich et al., 2015). Das Pretraining von GPT-2 wurde auf einem Datensatz mit ca. 8 Millionen Webseiten durchgeführt, mit der Aufgabenstellung jeweils in Texten das nächste Token vorherzusagen.

4 Variational Autoencoder

Variational Autoencoder (Diederik P Kingma und Welling, 2014) gehören zu den probabilistischen generativen Modellarchitekturen. In den letzten Jahren haben generative Modelle, insbesondere auch Variational Autoencoder, beeindruckende Möglichkeiten aufgezeigt, hochrealistische Daten wie zum Beispiel Bilder, Texte oder Audios zu generieren. Generative Modelle versuchen die Merkmale und Verteilung eines Datensatzes zu verstehen und anschließend neuartige Datenbeispiele, die ähnlich zum Trainingsdatensatz sind zu generieren.

Autoencoder sind neuronale Netze, die trainiert werden, um Eingabedaten zu komprimieren und anschließend zu rekonstruieren. Sie bestehen aus einem Encoder $\mathbf{E} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ und einem Decoder $\mathbf{D} : \mathbb{R}^m \rightarrow \mathbb{R}^n$, wobei der Encoder versucht, eine komprimierte Darstellung $c \in \mathbb{R}^m$ für die Eingabedaten $x \in \mathbb{R}^n$ im Latentspace zu finden. Die komprimierte Darstellung im Latentspace wird vom Decoder rekonstruiert mit dem Ziel, eine möglichst identische Rekonstruktion zur Eingabe $x \in \mathbb{R}^n$ zu erzeugen. Um stets eine komprimierte Darstellung im Latentraum zu erhalten und nicht eine Identitätsabbildung für den Encoder und Decoder wird die Dimension der Latentrepräsentation niederiger als die Eingabedimension gewählt, also $m < n$ (Diederik P. Kingma und Welling, 2019).

Da Autoencoder nur einen diskreten Latentraum erzeugen existieren im Latentraum viele leere Stellen, wodurch Interpolation durch den Latentraum zur Rekonstruktion neuer Ausgaben kaum möglich ist. Dies ist darauf zurückzuführen, dass ein Autoencoder auf möglichst geringen Verlust beim Encodieren und Decodieren trainiert wird und somit keinen Wert auf die Organisation des Latentraums legt.

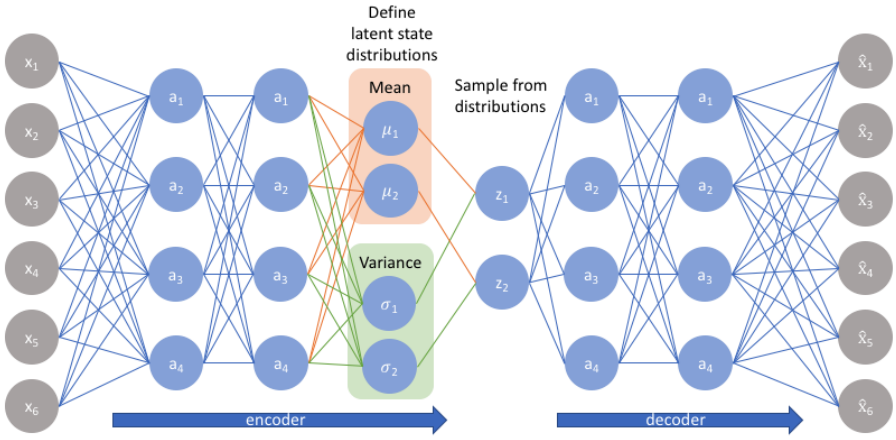

Abbildung 3: VAE Modellarchitektur (Jordan, 2018)

Im Gegensatz zu Autoencodern verwenden Variational Autoencoder einen probabilistischen Ansatz, um Datenpunkte x_i eines Datensatzes X, der durch die unbekannte Wahrscheinlichkeitsfunktion $P(\mathbf{X})$ beschrieben wird, im Latentraum Z zu repräsentieren. Die Architektur und der probabilistische Ansatz wird in Abbildung 3 dargestellt. Für den Datensatz X mit Datenpunkten x_i wird angenommen, dass ein generatives Modell mit den Parametern θ existiert, welches mittels Latentvektor z_i diesen Datenpunkt x_i generieren kann.

Dieser generative Teil des Variational Autoencoders ist der Decoder des Netzes, welcher approximativ versucht, die Daten als Verteilung $p_\theta(\mathbf{x})$ zu modellieren.

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}) dz \tag{3}$$

In Gleichung 3 wird für $p_\theta(\mathbf{z})$ die A-priori-Wahrscheinlichkeit angenommen und für $p_\theta(\mathbf{x} \mid \mathbf{z})$ die Likelihood. Der Decoder zieht demnach eine Latentvariable z aus der A-priori-Menge $p_\theta(\mathbf{z})$ und berechnet mit der Likelihood $p_\theta(\mathbf{x} \mid \mathbf{z})$ Datenpunkte x (Bank et al., 2020).

Die inferierten Latentvariablen für die bekannten Datenpunkte X lassen sich über die A-posteriori-Wahrscheinlichkeit $p_\theta(\mathbf{z} \mid \mathbf{x})$ aus Gleichung 4 herleiten.

$$p_\theta(\mathbf{z} \mid \mathbf{x}) = \frac{p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})} \tag{4}$$

Der Encoder des Variational Autoencoder approximiert die A-posteriori-Wahrscheinlichkeit durch ein Inferenzmodell $q_\phi(\mathbf{z} \mid \mathbf{x})$, da sich $p_\theta(\mathbf{z} \mid \mathbf{x})$ nicht trivial berechnen lässt. Die Parameter ϕ des Encoders werden optimiert, um mit $q_\phi(\mathbf{z} \mid \mathbf{x}) \approx p_\theta(\mathbf{z} \mid \mathbf{x})$ die wahre A-posteriori-Verteilung zu approximieren, indem die Kullback-Leibler-Divigenz zwischen den beiden Wahrscheinlichkeiten minimiert wird, wie in Abschnitt 4.1 beschrieben.

4.1 Evidence Lower Bound

Die Lossfunktion von Variational Autoencodern ist die **E**vidence **L**ower **Bo**und (ELBO). Bei Variational Autoencodern ist das Optimierungsziel die gleichzeitige Minimierung des Rekonstruktionsfehlers des generativen Modells θ zwischen Eingabe- und Ausgabedaten sowie die Approximation von $p_\theta(\mathbf{z} \mid \mathbf{x})$ mit $q_\phi(\mathbf{z} \mid \mathbf{x})$ durch Minimierung der Kullback-Leibler-Divergenz$D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x}))$. Für eine beliebige Wahl der Decoder Parameter ϕ gilt (Diederik P. Kingma und Welling, 2019):

$$\begin{aligned}
log(p_\theta(\mathbf{x})) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[log(p_\theta(\mathbf{x}))] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]\right]}_{=\mathcal{L}_{\theta,\phi}(\mathbf{x}) \text{ (ELBO)}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[log\left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right]}_{=D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))}
\end{aligned} \tag{5}$$

In Zeile 5 ist der erste Term die Evidence Lower Bound und der zweite Teil die Kullback-Leibler Divergenz. Die Kullback-Leibler Divergenz zwischen $q_\phi(\mathbf{z} \mid \mathbf{x})$ und $p_\theta(\mathbf{z} \mid \mathbf{x})$ ist nicht negativ und genau null, wenn $q_\phi(\mathbf{z} \mid \mathbf{x})$ der wahren A-posteriori-Verteilung entspricht.

$$D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x})) \geq 0 \tag{6}$$

Aus Gleichung 5 lässt sich durch Umformung die Evidence Lower Bound Funktion aufstellen:

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = log(p_\theta(\mathbf{x})) - D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x})) \leq log(p_\theta(\mathbf{x})) \tag{7}$$

Aufgrund der nicht Negativität der Kullback-Leibler Divergenz in Gleichung 6 ist die Evidence Lower Bound eine untere Schranke der Log-Likelihood der Datenmenge. Bei Maximierung der Evidence Lower Bound Funktion in Gleichung 7 wird die Evidenz $log(p_\theta(\mathbf{x}))$ maximiert, wodurch das generative Modell verbessert wird. Ebenfalls wird die Kullback-Leibler Divergenz minimiert, wodurch der Encoder die A-posteriori-Verteilung besser approximieren kann.

Somit wird als Lossfunktion $\mathbf{L}_{\theta,\phi}$ die negative Evidence Lower Bound Funktion gewählt, die es zu minimieren gilt:

$$\begin{aligned} \mathbf{L}_{\theta,\phi} &= -\mathcal{L}_{\theta,\phi}(\mathbf{x}) = -log(p_\theta(\mathbf{x})) + D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x})) \\ \hat{\theta}, \hat{\phi} &= \arg\min_{\theta,\phi} \mathbf{L}_{\theta,\phi} \end{aligned} \tag{8}$$

Diese Lossfunktion erlaubt das simultane Optimieren der beiden Parameter θ und ϕ.

4.1.1 Reparametisierung

Zum Training des Variational Autoencoder wird mittels Backpropagation der Fehler des Netzwerkes propagiert. Da das Samplen von $z \sim q_\phi(\mathbf{z} \mid \mathbf{x})$ nicht deterministisch ist, kann hier keine Backpropagation durchgeführt werden. Um dennoch Backpropagation verwenden zu können wird mittels Reparametisierung z durch eine deterministische Funktion $z = f_\phi(x, \epsilon)$ dargestellt mit ϵ als externe unabhängige Hilfsvariable (Diederik P Kingma und Welling, 2014; Jordan, 2018). Bei einer multivariaten Gaussverteilung für $q_\phi(\mathbf{z} \mid \mathbf{x})$ wäre die Reparametisierung wie folgt, wobei $\times$ die elementweise Multiplikation denotiert:

$$\begin{aligned} \mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{x}) &= \mathcal{N}(\mathbf{z}; \mu, \sigma\mathcal{I}) \\ \mathbf{z} = \mu + \sigma \times \epsilon \text{ , mit } \epsilon &\sim \mathcal{N}(0, \mathcal{I}) \end{aligned}$$

4.2 Cyclical Annealing Schedule

Trainieren von Variational Autoencodern als Sprachmodell im Bereich des Natural Language Processing ist aufgrund des KL-Vanishing Problems besonders schwierig. VAE Sprachmodelle sollen bei der Textgeneration den lokalen Kontext, allerdings auch globale Eigenschaften wie zum Beispiel Thema, Sprachstil oder Stimmung beachten. Trotzdem werden von VAE Modellen bei der Generierung oft die globalen Eigenschaften vernachlässigt (Fu et al., 2019). Dieses Problem ist als KL-Vanishing bekannt, da die KL-Regularisierung des Loss-Terms beim Trainieren von autoregressiven Decodern innerhalb von VAE Sprachmodellen sehr klein wird. Somit sind die erlernten Features nahezu identisch zu der vorher angenommenen Normalverteilung und der Decoder nutzt keine Latentfeatures bei der Generation.

Abhilfe schafft die Verwendung eines β-Variational Autoencoder (Fu et al., 2019) mit zyklischem Erhöhen des β Wertes. β-Variational Autoencoder sind Variational Autoencoder, die mit einen Lagrangemultiplikator β die KL-Divergenz der Loss-Funktion gewichten, um besser separierte Latentfaktoren zu finden.

$$\mathbf{L}_\beta(\beta, \theta, \phi) = -\mathbb{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})}[logp_\theta(\mathbf{x} \mid \mathbf{z})] - \beta D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z}))$$

Falls $\beta = 0$ ist wird das Modell wie ein normaler Autoencoder trainiert, bei $\beta = 1$ wie ein normaler Varitional Autoencoder.

Beim zyklischen Erhöhen wird der β-Wert des VAE während des Trainings monoton von $\beta = 0$ auf zum Ende des Trainings $\beta = 1$ in kleinen Abständen erhöht. So kann beim Training des VAE bei $\beta < 1$ zunächst der Fokus darauf gelegt werden, mehr Information für die Rekonstruktion zu erlernen. Abschließend enthalten bei $\beta = 1$ die vorher trainierten z Vektoren bereits mehr Informationen, die zu einer besseren Anpassung als eine zufällige Initialisierung führen.

4.3 Optimus

Optimus (**O**rganizing **S**entences via **P**re-trained **M**odeling of a **L**atent **S**pace) (Li et al., 2020) ist ein großes vortrainiertes Deep Latent Variable Modell für natürliche Sprache. Die Modelarchitektur von Optimus ist ein Variational Autoencoder, der als Encoder BERT und als Decoder GPT-2 verwendet wie in Abbildung 4 zu erkennen ist. Verwendet wird jeweils das vortrainierte BERT Base Modell ϕ_{BERT} und das vortrainierte GPT-2 Base Modell θ_{GPT-2} beide mit 12 Layern, 12 Attention Heads und einer Hiddensize von 768. Trainiert wird Optimus mit dem Ziel, Sätze in einem universellen Latentspace zu

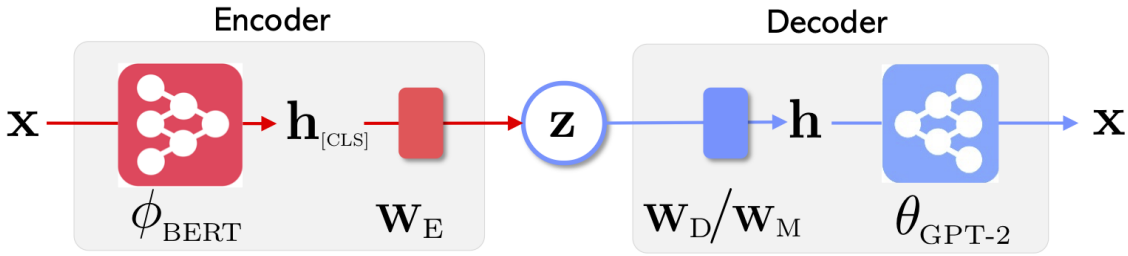

Abbildung 4: VAE Modellarchitektur von Optimus mit BERT als Encoder und GPT-2 als Decoder (Li et al., 2020)

organisieren und somit übergreifende semantische Muster für die einzelnen Sätze zu finden. Somit kann durch gezieltes Verändern des Latentvektors z kontrolliert Text generiert werden.

BERT und GPT-2 über eine VAE Architektur miteinander zu verbinden hat die Herausforderung, die unterschiedlichen Tokenisierungsschemen der einzelnen Modelle zu verwenden und den Latentvektor bei der Textgeneration von GPT-2 zu injizieren. Die Eingabetokens von BERT verwenden das WordPiece Embeddings Verfahren (Wu et al., 2016) mit einer Vokabulargröße von 28.996 Tokens. Die Ausgabe erfolgt über die Byte Pair Encoding Tokenisierung (Sennrich et al., 2015) von GPT-2 mit einer Vokabulargröße

von 50.260 Tokens. Innerhalb des Netzwerkes wird im Latentvektor ein Token durch eine Einbettung h_{Emb}, die das Token, die Position und das Segment Embedding wiedergibt, repräsentiert. Um beim Training den Loss der Rekonstruktionsaufgabe zu berechnen, werden die Sätze mit beiden Tokenisierungen tokenisiert.

Als Latentvektor $z \in \mathbb{R}^P$ wird die gepoolte Ausgabe des letzten Hiddenlayers $h_{[CLS]} \in \mathbb{R}^H$ von BERT mit einer Gewichtsmatrix multipliziert $W_E \in \mathbb{R}^{P \times H}$ gewählt. Somit kann ein Latentvektor wie folgt bestimmt werden $z = W_E h_{[CLS]}$.

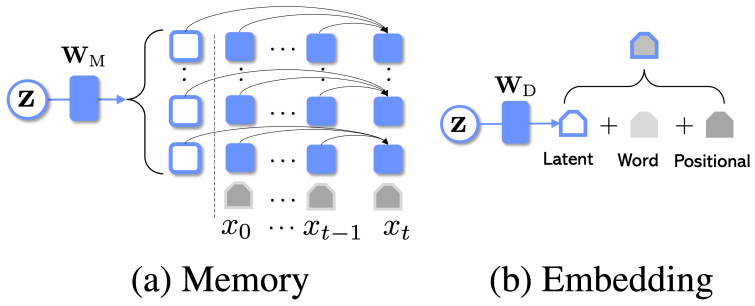

Abbildung 5: Methoden, um den Latentvektor in GPT-2 zu injizieren (Li et al., 2020)

Um mit GPT-2 kontrolliert unter Verwendung des Latentvektors Text zu generieren, kann der Latentvektor entweder als Memoryvektor in die Past-Gewichtsmatrix injeziert oder auf die vorherigen Embeddings addiert werden (Li et al., 2020).

Beim Embedding wird z direkt auf den Embedding Layer addiert. Somit ergibt sich der neue Embedding Layer durch $h'_{Emb} = h_{Emb} + W_D z$ mit der Gewichtsmatrix $W_D \in \mathbb{R}^{H \times P}$. Der Decoder kann hier die zusätzlichen Informationen des Embeddings im Input und Output Layer verwenden.

Bei der Injezierung von z als Memoryvektor $h_{Mem} \in \mathbb{R}^{L \times H}$ wird der Latentvektor in den Past-Key-Vektor von GPT-2 injeziert. Der Past-Key-Vektor beschleunigt normalerweise den Decodiervorgang von GPT-2, da bei einem Decodierungsdurchlauf zu den vorherigen Tokens bereits berechnete Key- und Valuevektoren der Attentionlayern gespeichert und wiederverwendet werden. Diese Key-, Valuevektoren des Attentionlayers werden durch $h_{Mem} = W_M z$ mit $W_M \in \mathbb{R}^{LH \times P}$ ersetzt. GPT-2 kann so beim Decodieren auf den injezierten Latentvektor bei jeder Attentionberechnung zugreifen.

Die Parameter $\phi_{BERT}, \theta_{GPT-2}, W_E, W_M, W_D$ des VAEs werden mittels Cyclical Annealing Schedule (Fu et al., 2019) trainiert, um das KL Vanishing Problem beim Trainieren eines VAEs zu verhindern. Die β Variable, die wie in Abschnitt 4.2 dargestellt den KL-Regularisierungs Anteil steuert, wird für einen Trainingsdurchlauf für die erste halbe Menge der Trainingsdaten auf $\beta = 0$ gesetzt. Somit wird zu Beginn lediglich ein Autoencoder trainiert. Anschließend wird β für das nächste Viertel der Trainingsmenge schrittweise von 0 auf 1 erhöht und für das letzte Viertel der Trainingsmenge auf 1 belassen, um das VAE Modell zu trainieren.

Insgesamt zeigt Optimus gute Ergebnisse in den Bereichen des Language Modeling, der kontrollierten Text Generation und des Language Understanding.

4.4 BiMeanVAE

BiMeanVAE ist ein von (Iso et al., 2021) vorgestelltes Modell, bestehend aus einem bidirektionalem LSTM Encoder und einem LSTM Decoder. Die Hiddensize der einzelnen LSTMs beträgt 512. Nach dem BiLSTM Encoder wird ein Mean Pooling Layer verwendet, um einen Latentvektor z zu erhalten. BiMeanVAE verwendet die Standard Variational Autoencoder Ziele, dass Minimieren des Rekonstruktionsfehlers mit KL-Regularisierung. Die verwendete Prior Verteilung $p_\theta(z)$ ist die Gausssche Normalverteilung $\mathcal{N}(0, \mathcal{I})$. Beim Training wird ebenfalls das in Abschnitt 4.2 beschriebene Cyclical Annealing Verfahren verwendet, um graduell vom Autoencoder zum Variational Autoencoder Trainingsziel überzuleiten.

4.5 Latent Space Operationen

VAEs erlernen bidirektionale Abbildungen zwischen dem Datenraum und ihrem Latentspace. Im Latentspace werden Eingabesequenzen von Optimus und BiMeanVAE jeweils durch einen Latentvektor dargestellt. Die zur Generation verwendeten Latentvektoren können durch arithmetische Operationen verändert werden, wodurch sich die gesampelte Ausgabe des VAEs gezielt verändern und steuern lässt.

Durch die KL-Regularisierung beim Training von VAEs entsteht ein kontinuierlicher Latentspace, wodurch semantisch ähnliche Inhalte im Latentspace nah beieinander organisiert werden. Somit kann zwischen mehreren Datenobjekten im Latentspace interpoliert werden.

Insbesondere für das kontrollierte Generieren von Bewertungen sind diese Latent Space Operationen, die VAEs ermöglichen, elementar. So kann zum Beispiel zwischen mehreren Bewertungen interpoliert werden, um eine durchschnittliche Meinung zu generieren.

Sei zum Beispiel $\mathcal{Z}$ die Menge der encodierten Bewertungen zu einer Produktgruppe, so lässt sich aus einer Untergruppe der Vektoren $\mathcal{Z}' \subseteq \mathcal{Z}$ durch Kombinieren eine gute Durchschnittsrepräsentation im Latentvektorraum finden. Das genaue Vorgehen zum Finden einer solchen Untergruppe wird in Abschnitt 6.1 beschrieben

5 Datensätze

Zum Training und zur Evaluation von neuronalen Netzen sind große Datensätze erforderlich. Zur Lösung der Aufgabenstellung, Durchschnittsrezensionen aus Textbewertungen zu erzeugen, werden Datensätze mit mehreren Bewertungen und Zusammenfassungen zu einem Produkt benötigt.

Es bietet sich an, Bewertungen von großen Webportalen wie Amazon oder Yelp zu verwenden. Diese Portale haben zu den unterschiedlichsten Produkten und Dienstleistungen zahlreiche Bewertungen.

Es existieren bereits bestehende Amazon und Yelp Datensätze mit menschlich erstellten Gold-Standard Zusammenfassungen (Brazinskas et al., 2019). Die Variational Autoencoder Modelle lassen sich somit mit den Review Daten mit dem Trainingsziel der Rekonstruktion trainieren, um einen aussagekräftigen Latentraum mit Bewertungen und deren spezifischen Eigenschaften zu erlernen. Anschließend können die trainierten Modelle auf den respektiven Testdatensätzen evaluiert werden, da jeweils Gold-Standard Zusamenfassungen vorliegen, mit denen sich mittels in Abschnitt 8 erklärten Metriken die generierten Rezensionen bewerten lassen.

Der Umfang der zum Training und Testen verwendeten Datensätze ist in Tabelle 1 angegeben.

	Amazon			**Yelp**		
	Train	Dev	Test	Train	Dev	Test
# Produkte / Dienstleister	244.652	28	32	75.988	100	100
# Rezensionen	13.053.202	224	256	4.658.968	800	800

Tabelle 1: Enthaltene Rezensionen der einzelnen Datensätze (Iso et al., 2021)

5.1 Amazon Datensatz

Der Amazon Review Datensatz (Bražinskas et al., 2020) umfasst zahlreiche Rezensionen zu den unterschiedlichsten Produkten, die auf dem Amazon Marktplatz gelistet sind. Die Rezensionen wurden von Amazon Nutzern erstellt und geben eine unabhängige Meinung über das erworbene Produkt wieder. Bei Betrachtung der einzelnen Bewertungen fällt auf, dass die meisten Bewertungen von Produkten eher objektiv formuliert sind. Die Bewertungen gehen oft auf unterschiedliche Aspekte der Produkte ein und erläutern positive sowie negative Erfahrungen, die auf einzelne Produkteigenschaften zurückzuführen sind.

Die für das Training relevanten Bewertungen wurden aus den Kategorien Kleidung, Schuhe, Schmuck, Elektronikartikeln, Gesundheit- und Pflegeprodukte, Einrichtung und Küchenartikeln gewählt. Gefiltert wurden die Bewertungen indem Produkte mit einer maximalen Länge von 128 Tokens gewählt wurden und nur Produkte mit über 10 existierenden Bewertungen ausgewählt wurden. Die entsprechenden Bewertungen sind bereits in Trainings- und Validierungsdaten gesplittet.

Produkt: NuGo Protein Bar, Vanilla Yogurt

Review: Apart from the obvious nutritional value, the taste was surprisingly better than I anticipated. Normally, if I find soy-based nutritional bars tend to have a chalky or tree-bark-like texture and taste. I dare say, the texture reminds me of a healthy-not-too-sweet 'Rice Krispy Treat'.

Abbildung 6: Beispiel Bewertung zu einem Produkt des Amazon Datensatzes

Die Validierungsdaten des Amazon Datensatz sind manuell erstellte Zusammenfassungen, die für den Dev / Test Split im Verhältnis von 28 / 32 vorliegen. Für jedes Produkt der Validierungsdaten existieren jeweils drei manuell erstellte Referenzzusammenfassungen. Somit ergibt sich insgesamt ein Trainingsdatensatz mit 244.652 Produkten und den zugehörigen 13.053.202 Bewertungen.

5.2 Yelp Datensatz

Der Yelp Review Datensatz (Chu und Liu, 2019) enthält Rezensionen zu unterschiedlichen Gastronomiebetrieben und Unternehmen der Bewertungsplattform Yelp. Die von den Yelpnutzern erstellten Bewertungen unterscheiden sich von den Amazon Bewertungen durch mehr Subjektivität in den Bewertungen. Die Bewertungen enthalten wesentlich mehr persönliche Details und Erfahrungen der Nutzer. Das Variational Autoencoder Modell hat hier die Aufgabe, wichtige Informationen zu extrahieren und unwichtiges Rauschen in den Daten herauszufiltern, um eine gute Durchschnittsrepräsentation zu bestimmen. Insbesondere in den Yelp Reviews befinden sich teilweise unnötige Zusatzinformationen, die nicht zur Bewertung der Dienstleistung an sich beitragen, wie zum Beispiel das Erwähnen von privat ausgetragenen Geburtstagsfeiern.

Dienstleister Id: 87YsVbCN_kfzheY79Fzjkg

Review: Great experience! Went cake tasting for our wedding and Jessica was a huge help!! She brought out the tastings for us after giving us time to look through their amazing book. She explained every cake and filling and had great recommendations for mixing the fillings. The champagne cake with cream cheese filling was amazing!! Chose our cake and design within an hour. Great service and knowledge. Thank you Jessica!

Abbildung 7: Beispiel Bewertung zu einem Restaurant des Yelp Datensatzes

Der Yelp Review Datensatz wurde ebenfalls nach einer maximalen Länge von 128 Tokens mit mindestens 10 existierenden Bewertungen je Unternehmen gefiltert und besteht somit aus 13.053.202 Bewertungen zu 244.652 Unternehmen, die bereits in Trainings- und Validierungsdaten gesplittet sind. Zusätzlich enthält der Datensatz 200 menschlich durch Amazon Mechanical Turk (AMT) Arbeiter erstellte Zusammenfassungen, die zu einem Dev / Test Verhältnis von 100 / 100 gesplittet werden. Bei dem Yelp Datensatz liegt für jede der 200 Dienstleistungen der Validierungsdaten lediglich eine Referenzzusammenfassung vor.

6 Multi-Document Summarization

In den letzten Jahren erzielten unterschiedliche Verfahren große Fortschritte beim automatisierten Zusammenfassen von mehreren Dokumenten zu einem repräsentativen Dokument. Im Gegensatz zum normalen Zusammenfassen von einzelnen Dokumenten enthalten bei der Multi-Document Summarization mehrere Dokumente über ein Thema viele redundante Informationen, die so gefiltert werden müssen, dass redundante Informationen nur einmal im Ausgabedokument dargestellt werden. Ebenfalls ist das Verarbeiten von widersprüchlichen Informationen eine große Herausforderung bei der Zusammenfassung von mehreren Dokumenten. Das Ziel von Multi-Document Summarization ist das zusammengefasste Repräsentieren der wichtigen Aspekte aller Dokumente in einem einzelnen Dokument, um einen guten allgemeinen Überblick zu schaffen.

Im folgenden werden Multi-Document Summarization Systeme zur Zusammenfassung von Bewertungen verwendet. Sei $R = \{x_i\}$ ein Datensatz von Bewertungen mit einzelnen Bewertungen $x = (x_1, ..., x_{\|x\|})$, die aus einer Sequenz aus Tokens bestehen. Ziel ist es für ein gegebenes Produkt p und die entsprechenden Bewertungen $R_p \subseteq R$ eine Zusammenfassung s_p zu generieren, die alle relevanten Informationen faktisch korrekt repräsentiert.

Insbesondere im Bereich der Multi-Review Summarization existieren viele unterschiedliche Ansätze. Zum einen existieren extraktive Ansätze, wie zum Beispiel LexRank (Erkan und Radev, 2011), ein unüberwachter Algorithmus, der repräsentative Sätze für eine Bewertung basierend auf ihrer Zentralität in einem TF-IDF (**T**erm **F**requency - **I**nverse **D**ocument **F**requency) (Sammut und Webb, 2010) gewichteten Graphen selektiert. Zum anderen existieren abstraktive Ansätze, wie zum Beispiel MeanSum (Chu und Liu, 2019), CopyCat (Brazinskas et al., 2019) oder COOP (Iso et al., 2021), die generative Ansätze verwenden, um neuartige Sätze in den Bewertungen zu erzeugen.

Die abstraktiven Modelle basieren auf Autoencoder Architekturen. Es wird ein Encoder $E_\phi : X \to Z$ verwendet, um Textrepräsentationen in einen Latentvektor im Latentraum Z umzuwandeln. Aus den Latentvektoren z kann anschließend unter Verwendung von einem Decoder $D_\theta : Z \to \hat{X}$ eine Textrepräsentation generiert werden.

MeanSum ist ein Autoencoder Modell (Chu und Liu, 2019) mit LSTM Encoder und Decoder und errechnet zur Zusammenfassung von Bewertungen den Durchschnitt von den einzelnen Latentvektoren der Bewertungen $z_p = \bar{z}$, mit $z = \{z_{p_1}, z_{p_2}, ...z_{p_k}\}$. Von diesem Durchschnittslatentvektor z_p wird anschließend eine Bewertung generiert.

CopyCat basiert auf einem Variational Autoencoder Modell (Brazinskas et al., 2019), welches GRU Encoder und Decoder verwendet. Für jede Gruppe von Bewertungen berechnet CopyCat einen Latentvektor c, der die gesamte Semantik der Gruppe beschreibt. Weiterhin wird jede einzelne Bewertung einer Gruppe mit einem einzelnen Latentvektor z beschrieben. Bei der Generation von Durchschnittsbewertungen ermöglicht es CopyCat dem Decoder, die einzelnen Latentvektoren für die Bewertungen z_i, den allgemeinen Gruppenvektor c und die Bewertungen x_i an sich zu betrachten. Durch den Zugriff auf die anderen Bewertungen x_i kann der Decoder spezifische Worte von diesen „kopieren“ und somit übernehmen.

6.1 Convex Aggregation for Opinion Summarization

Convex Aggregation for **Op**inion Summarization kurz COOP (Iso et al., 2021) ist eine Methode, die unterschiedliche Kombinationenen von einzelnen Latentvektoren zum Berechnen einer Durchschnittsbewertung untersucht. Bei der Verwendung von Variational Autoencodern werden die einzelnen Eingabebewertungen durch einen Latentvektor repräsentiert. Häufig wird im nächsten Schritt zur Berechnung einer Durchschnittsbewertung der normale Durchschnitt aller verwendeten Latentvektoren bestimmt und von diesem eine neue Durchschnittsbewertung generiert.

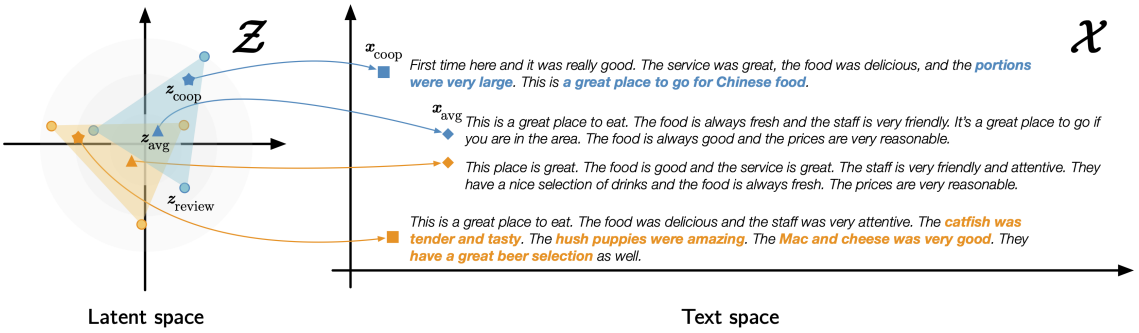

Abbildung 8: Latentraum Z mit den entsprechenden generierten Bewertungen X (Iso et al., 2021)

Abbildung 8 zeigt, dass sich Bewertungen sehr ähneln, die durch Bestimmung des Durchschnitts der Latentvektoren generiert worden sind. Auffällig ist die Korrelation zwischen der Ausdrucksstärke und dem Informationsgehalt eines Latentvektors und seiner L_2-Norm $\|z\|$. Latentvektoren, die durch den Durschnitt bestimmt werden, haben eine geringere L_2-Norm und somit in den erzeugten Bewertungen einen geringeren Informationsgehalt (Iso et al., 2021).

Um ausdrucksstarke Latentvektoren für die Generation der Bewertungen zu erhalten, formuliert COOP (Iso et al., 2021) die Bestimmung des optimalen Latentvektors als Optimierungsproblem, um die beste Kombination von einzelnen Latentvektoren zu finden.

$$\max_{z} \quad Overlap(R_p, D_\theta(z))$$

$$\text{unter der Nebenbedingung:} \qquad z = \sum_{i=1}^{|R_p|} w_i z_i$$

$$\sum_{i=1}^{|R_p|} w_i = 1, \quad \forall w_i \in \mathbb{R}^+$$

Maximiert wird der *Input-Output-Word Overlap* zwischen den Eingabebewertungen R_p und der generierten Bewertung $D_\theta(z)$. Die *Overlap*-Metrik in Gleichung 9 basiert auf dem ROUGE-1 F1 Score und ermöglicht es, Bewertungen zu generieren, die konsistent zu den Eingabebewertungen sind.

$$Overlap(X, Y) = \text{ROUGE-1}_{F_1}(X, Y) \tag{9}$$

Die Suche der einzelnen Kombinationen wird auf die Potenzmenge der Eingabebewertungen R_p beschränkt. Der Zusammenfassungslatentvektor z_p aus Gleichung 10 wird anschließend aus dem Durchschnitt der ausgewählten Eingabebewertungslatentvektoren berechnet.

$$z_p = \frac{1}{|R_p^{'}|} \sum_{i=1}^{R_p^{'}} z_i, \text{ mit } R_p^{'} \in 2^{R_p} \setminus \{\emptyset\} \tag{10}$$

Die COOP Methode zur Kombination der einzelnen Latentvektoren, um ausdrucksstarke Bewertungen zu erhalten, erreicht State-of-the-Art Ergebnisse im Vergleich zu den anderen vorgestellten Methoden, die lediglich den normalen Durchschnittslatentvektor verwenden. Die erzeugten Bewertungen sind repräsentativ für eine Gruppe von Bewertungen und erhalten ein hohes Maß an Informationen. Trotzdem stellt sich die Frage, ob und inwiefern sich diese Ergebnisse noch weiter optimieren lassen.

Im weiteren Verlauf wurde ein Verfahren entwickelt, welches entsprechende Latentvektoren weiterhin adaptiert, um ein noch höheres *Input-Output-Overlapping* zu erhalten und somit präzisere Informationen in den Durchschnittsbewertungen zu generieren.

Die beiden Modelle Optimus und BIMEANVAE sind bereits von den Autoren (Iso et al., 2021) vortrainiert und werden in dieser Arbeit bereits vortrainiert verwendet. Als Optimierer wurde Adam verwendet mit einer Learningrate von 10^{-5} für Optimus und 10^{-3} für BIMEANVAE. Die Cyclical Annealing Learning Strategie wurde wie in Kapitel 4.2 beschrieben angewandt.

7 Kontrollierbare Textgeneration von Sprachmodellen

Die in Abschnitt 6.1 vorgestellte COOP Methode zur Suche der optimalen Kombination von Latentvektoren zur Maximierung des *Input-Output-Overlaps* erzielt beeindruckende Resultate (Iso et al., 2021). COOP verwendet die VAE Modelle Optimus und BIMEAN-VAE, welche zur Generation den kombinierten Latentvektor z mittels Decoder $p_\theta(x|z)$ zu einem Text $\hat{x}$ rekonstruieren. Optimus verwendet zur Decodierung ein GPT-2 Modell. BIMEANVAE decodiert den Latentvektor mit einem LSTM Decoder.

Obwohl durch die Kombination mehrerer Latentvektoren bereits gute Ergebnisse erzielt werden, stellt sich die Frage, ob sich für diese kombinierten Latentvektoren durch weitere Optimierungen noch bessere Ergebnisse erzielen lassen.

Unkontrollierte Sprachmodelle modellieren Texte über die Wahrscheinlichkeit $p(X)$ für eine Sequenz $X = (x_0, ..., x_{\|x\|})$. In Kapitel 3 wurde die Funktionsweise von Transformer-Sprachmodellen und die autoregressive Generierung von Textsequenzen erklärt. Bei der Generierung werden die vorherigen Key-Value Paare der Attention-Layer in einer Vergangenheitsmatrix $H_t = [(K_t^{(1)}, V_t^{(1)}), \ldots, (K_t^{(n)}, V_t^{(n)})]$ gespeichert, wobei K und V die einzelnen Key-, Value-Vektoren im Layer n zum Zeitpunkt t repräsentieren. Diese Vergangenheitsmatrizen werden verwendet, um bei der Generierung auf bereits vorher berechnete Key-, Value-Werte zurückgreifen zu können und somit effizienter Text zu generieren.

Das Unternehmen Uber hat mit der Einführung von **P**lug and **P**lay **L**anguage **M**odels (PPLM) (Dathathri et al., 2019) es ermöglicht, die Textgeneration von großen Sprachmodellen wie zum Beispiel GPT-2 kontrolliert zu beeinflussen. Kontrollierbare Generierung von Texten mittels Sprachmodellen entspricht dem Modellieren von $p(x|a)$, wobei hier a für ein kontrollierbares Attribut in Bezug auf den generierten Text x steht. Mit dem Satz von Bayes lässt sich das kontrollierbare Sprachmodell zu $p(x|a) \propto p(a|x)p(x)$ umformulieren (Dathathri et al., 2019). Das Attribut Modell $p(a|x)$ bewertet einen Satz x auf den Besitz eines Attributs a mit einer Wahrscheinlichkeit.

Zur kontrollierbaren Generierung werden bei PPLM-Modellen Gradienten für die generierten Sequenzen über die Log-Likelihood des normalen Sprachmodells $log(p(x))$ und der Log-Likelihood des Attribut-Modells $log(p(a|x))$ in Bezug auf die Vergangenheitsmatrix errechnet. Durch Veränderung der Vergangenheitsmatrix $\tilde{H}_t = (H_t + \Delta H_t)$ wird die Wahrscheinlichkeit, das nächste Token mit den gewünschten Attributen zu erhalten, erhöht. Hierbei wird ΔH_t schrittweise durch den Gradienten des Attribut-Modells bestimmt. Die Gradientenmatrix ΔH_t wird als Null-Matrix initialisiert. Um den Gradienten des Attribut-Modells zu bestimmen, wird dieses zu $p(a|H_t + \Delta H_t)$ umformuliert mit folgendem Iterationsschritt 11 zur Berechnung:

$$\Delta H_t \leftarrow \Delta H_t + \alpha \frac{\nabla_{\Delta H_t} \log p(a|H_t + \Delta H_t)}{\|\nabla_{\Delta H_t} \log p(a|H_t + \Delta H_t)\|^\gamma} \tag{11}$$

Der Iterationsschritt 11 kann mehrmals hintereinander ausgeführt werden.

Die so durch die modifizierte Vergangenheitsmatrix $\tilde{H}_t$ bestimmte Ausgangsverteilung $\tilde{p}_{t+1}$ wird mit der nicht modifizierten Ausgangsverteilung p_{t+1} kombiniert, um den generierten Text ebenfalls durch die Sprachmodell-Verteilung zu beeinflussen. Somit lässt sich das nächste Token wie in Gleichung 12 beschrieben von den kombinierten Verteilungen p_{t+1} und $\tilde{p}_{t+1}$ samplen (Dathathri et al., 2019):

$$\hat{x}_{t+1} \sim \frac{1}{\beta}(\tilde{p}_{t+1}^{\gamma} \cdot p_{t+1}^{1-\gamma}) \tag{12}$$

Durch Veränderung von γ bei der Kombination lässt sich der Einfluss des unmodifizierten Sprachmodells auf die Ausgabe festlegen. Hier konvergiert bei $\gamma \to 1$ die Ausgabe zur Verteilung des modifizierten Sprachmodells und $\gamma \to 0$ gegen die Ausgabe des unmodifizierten Sprachmodells.

7.1 Bag of Words Attribut-Modell

Als Attribut Modell wird ein Bag of Words Modell verwendet, welches einen Loss über die Summe der Wahrscheinlichkeiten der einzelnen vorhergesagten Wörter bildet. Sei $\{w_0, \ldots, w_n\}$ eine Gruppe von Tokens, die ein bestimmtes Thema repräsentieren und p_{t+1} die Ausgabeverteilung über die Tokens des Sprachmodells. Dann wird die Log-Likelihood des Attributmodells durch Gleichung 13 beschrieben:

$$\log p(a|x) = \log \left(\sum_{i=0}^{n} p_{t+1}[w_i]\right) \tag{13}$$

Um den *Input-Output-Overlap* zwischen den Eingabebewertungen und den generierten Bewertungen zu maximieren, kann das Bag of Words Modell auf unterschiedliche Weise erzeugt werden. Eine Methode ist es, die k am häufigsten vorkommenden Wörter über alle Eingabebewertungen eines Produktes zu wählen. Vor dem Auswählen der häufigsten vorkommenden Wörter werden Stoppwörter entfernt.

Des Weiteren können die ausgewählten Bag of Words Tokens gewichtet werden, indem die k häufigsten Wörter nach ihrer Anzahl der Vorkommnisse über eine Softmax-Funktion in eine Wahrscheinlichkeitsverteilung transformiert werden. Hierdurch erhalten besonders häufig vorkommende Wörter ein höheres Gewicht als weniger häufig vorkommende Wörter.

Die Bag of Words Menge kann auch durch anderweitige Keyword-Extraktionsmethoden wie zum Beispiel durch YAKE (Campos et al., 2020) generiert werden.

In Tabelle 2 wurden auf dem Amazon Dev-Datensatz die unterschiedlichen Verfahren, um Bag of Words Mengen zu erzeugen, evaluiert. Es zeigt sich, dass bei dem Bag of Words Attributmodell eine Wortanzahl von 150 optimal ist, um gute Ergebnisse zu erzielen. Eine höhere Wörteranzahl ergibt keine weitere Leistungssteigerung. Dies lässt sich darauf zurückführen, dass gar nicht mehr Wörter ausgewählt werden können und diese bereits den gesamten Text umfassen würden.

Modell	Amazon		
BIMEANVAE	**R1**	**R2**	**RL**
Baseline	39.50	8.62	22.79
k = 50	40.72	**9.06**	**23.40**
k = 150	**40.75**	9.03	**23.40**
k = 500	**40.75**	9.03	**23.40**
k = 50 + Softmax	40.72	**9.06**	**23.40**
k = 150 + Softmax	**40.75**	9.03	**23.40**
k = 500 + Softmax	**40.75**	9.03	**23.40**
YAKE	40.72	9.00	**23.40**

Tabelle 2: Ergebnisse für die unterschiedlichen Attributionsmodelle mit BIMEANVAE auf dem Amazon Dev-Datensatz

Darüber hinaus fällt auf, dass entgegen der Erwartung eine leichte Gewichtung mittels Softmax Funktion nach der Anzahl der Vorkommnisse entsprechender Wörter zu keinen besseren Ergebnissen führt. Keyword Extraction über das YAKE-Verfahren zeigt ebenfalls keine Leistungssteigerung.

Demnach wird im folgenden für die weitere Evaluierung auf dem Dev- und Testdatensatz das Bag of Words Attributmodell mit der Wörterauswahl aus den $k = 150$ am häufigst vorkommenden Wörtern initialisiert.

7.2 Verbessern der Textgeneration von Optimus

Den Variational Autoencoder Optimus mit einem Attributions-Modell zu kombinieren ist aufgrund der Injektion des Latentvektors nicht ohne weiteres möglich. Optimus kann bereits unter Einbezug des Latentvektors Texte kontrolliert generieren $p(x|z)$. Die Berücksichtigung eines Attribut-Modells bei der Generierung des Textes entspricht $p(x|a,z) \propto p(a|x,z)p(x|z)$. Da der Latentvektor z in die Vergangenheitsmatrix H_t injeziert wird, wird der Latentvektor direkt optimiert und Δz ergibt sich durch die folgende Iteration 14:

$$\Delta z \leftarrow \Delta z + \alpha \frac{\nabla_{\Delta z} \log p(a|z+\Delta z)}{\|\nabla_{\Delta z} \log p(a|z+\Delta z)\|^{\gamma}} \tag{14}$$

In Abschnitt 7.6 wird die vorgeschlagene Optimierung der Latentvektoren anhand der Test-Datensätze evaluiert und nach geeigneten Parametern untersucht.

7.3 Verbessern der Textgeneration von BIMEANVAE

BIMEANVAE ist ein Variational Autoencoder, bestehend aus einem bidirektionalem LSTM Encoder, gepaart mit einem LSTM Decoder. Der LSTM Decoder erhält als Eingabe den Latentvektor z. Der Hiddenstate h_t und Cellstate c_t wird aus dem Embedding des Latentvektor z initialisiert und anschließend autoregressiv über die LSTM Architektur aktualisiert. Um diesen LSTM Decoder mit einem Attributmodell zu optimieren, bieten sich drei unterschiedliche Ansätze:

1. Optimieren über den Latentvektor z
2. Optimieren des vorherigen Hiddenstates h_t vor der nächsten Berechnung
3. Optimieren des vorherigen Cellstates c_t vor der nächsten Berechnung

Es ist möglich, zu allen drei aufgezählten Optionen einen Gradienten zur Veränderung der entsprechenden Variabel zu bestimmen.

Nachfolgend wurden alle drei Optionen auf dem Amazon-Testdatensatz evaluiert, um die zu optimierende Variabel mit der bestmöglichen Performance zu finden. Die gewählte Schrittgröße ist $\alpha = 0.1$. Bei der Selektion der Ausgabebewertung wurde die Bewertung ausgewählt, die die höchste ROUGE-1 Übereinstimmung mit den Zusammenfassungsbewertungen hat. Somit wird stets die bestmögliche generierte Bewertung ausgewählt.

Modell	Amazon		
BIMEANVAE	**R1**	**R2**	**RL**
Baseline	39.50	<u>8.62</u>	22.79
optimze z	<u>40.04</u>	8.35	**23.64**
optimze h_t	39.96	8.34	22.94
optimze c_t	**40.81**	**8.91**	<u>23.49</u>

Tabelle 3: Ergebnisse für die Optimierung der unterschiedlichen Variablen z, h_t, c_t über ein Attributionsmodell

In Tabelle 3 sind die ROUGE-Scores, welche in Abschnitt 8 genauer erklärt werden, für die separiert optimierten Variablen z, h_t, c_t aufgelistet. Beim Evaluieren der unterschiedlichen Optimierungsversuche zeigt sich, dass jede Optimierung bessere Ergebnisse als die unoptimierte Baseline auf den Gold-Summaries erzielt. Die größte Leistungssteigerung erzielt eine Optimierung der Variabel c_t. Somit beschreibt folgende Gleichung 15 den Optimierungsschritt Δc_t bei BIMEANVAE.

$$\Delta c_t \leftarrow \Delta c_t + \alpha \frac{\nabla_{\Delta c_t} \log p(a|c_t + \Delta c_t, z)}{\|\nabla_{\Delta c_t} \log p(a|c_t + \Delta c_t, z)\|^{\gamma}} \tag{15}$$

7.4 Latentvektoroptimierung mit Beam Search

Beam Search ist ein Suchalgorithmus zum Auffinden eines Ausgabesatzes mit der höchsten Wahrscheinlichkeit. Im Gegensatz zur normalen Greedy Suche, bei der nachfolgend das bestmögliche Token ausgewählt wird, wird beim Beam Search Algorithmus eine Gruppe von N bestmöglichen Tokens für eine Position ausgewählt. Somit wird ein Suchbaum erstellt, der eine maximale Breite von N hat und die Äste mit den N höchsten Wahrscheinlichkeiten weiterführt. Demnach ist Beam Search nicht optimal, da die bestmögliche Lösung möglicherweise nicht fortgeführt wird. Trotzdem lassen sich mit Beam Search erfolgreich gute Ausgabesätze finden, die einer hohen Wahrscheinlichkeit entsprechen.

7.5 Moverscore Ranking

Da dem Modell zur Bestimmung des *Input-Output-Overlaps* mittels ROUGE-1 nur die Eingabebewertungen zur Verfügung stehen, erhalten teilweise generierte Rezensionen mit gleicher Semantik aber anderen Formulierungen schlechte *Input-Output-Overlap*-Scores. Der in Abschnitt 8.2 vorgestellte Moverscore ermöglicht einen semantischen Vergleich von Textsequenzen. So können insbesondere Rezensionen mit unterschiedlich formulierten Meinungen, die die gleiche Aussage treffen, ein hoher Ähnlichkeitswert zugewiesen werden.

Zwischen den Eingabebewertungen und den Gold-Summaries besteht eine hohe semantische Ähnlichkeit, weshalb der Moverscore miteinbezogen wird, um ein besseres Ranking der Ausgabewertungen zu erzeugen. Des Weiteren wird die *Input-Output-Overlap* Funktion dahingehend abgeändert, dass sie ebenfalls die ROUGE-2 und ROUGE-L Scores miteinbezieht. Somit ergibt sich eine neue Rankingfunktion 16 zwischen den generierten Ausgabebewertungen $\hat{x_i}$ und den Eingabereviews R:

$$\begin{aligned} \textit{Input-Output-Overlap}(\hat{x_i}, R) &= x \cdot \text{R1}(\hat{x_i}, R) + y \cdot \text{R2}(\hat{x_i}, R) + z \cdot \text{RL}(\hat{x_i}, R) \\ \text{SCORE}(\hat{x_i}) &= \textit{Input-Output-Overlap}(\hat{x_i}, R) + v \cdot \textit{Moverscore}(\hat{x_i}, R) \end{aligned} \tag{16}$$

Die Variablen x, y, z, v geben die Gewichtung der einzelnen Metriken an. Die Hyperparameter für die Gewichtung werden im folgenden Abschnitt auf dem Dev-Datensatz ermittelt.

Mit dieser neuen Rankingfunktion wird bei den einzelnen generierten Rezensionen nun die semantische Ähnlichkeit mit den Eingabebewertungen miteinbezogen. Hierdurch können auch generierte Bewertungen mit weniger überlappenden N-Grammen aber einer hohen semantischen Ähnlichkeit mit einer höheren Wahrscheinlichkeit ausgewählt werden.

7.6 Hyperparameter Optimierung mittels Dev-Datensatz

Die Hyperparameter x, y, z, v der Rankingfunktion 16 legen bei der Auswahl der generierten Rezensionen fest, wie stark der ROUGE-1, ROUGE-2, ROUGE-L oder Moverscore berücksichtigt werden soll. Ein weiterer wichtiger Hyperparameter ist die Schrittweite α des Attributmodells. Die Schrittweite bestimmt den Grad der Korrektur des Gradienten pro Iteration des Attributmodells. Da die Datensätze jeweils unterschiedliche Eigenschaften haben, wie in Abschnitt 5 erläutert, bietet es sich an für die jeweiligen Datensätze eigene Hyperparameter zu finden.

Mit diesen adaptierten Rankingfunktionen ergeben sich bei der Auswertung auf dem Dev-Datensatz folgende in Tabelle 4 dargestellte Ergebnisse. Die Gewichtungen wurden unter Verwendung eines Greedy-Search-Algorithmus bestimmt. Für jede Schrittweite wurden jeweils die besten bestimmten Gewichtungen ausgewählt.

DEV-Dataset	Amazon				Yelp			
Method	**R1**	**R2**	**RL**	**MV**	**R1**	**R2**	**RL**	**MV**
COOP + Attribute Model	Stepsize $\alpha = 0.05$							
Optimus	36.31	7.17	20.75	**56.82**	36.31	8.12	20.02	56.71
BIMEANVAE	36.40	7.19	21.65	**56.55**	36.16	7.51	20.49	56.91
COOP + Attribute Model	Stepsize $\alpha = 0.1$							
Optimus	36.40	**7.78**	**20.99**	56.78	36.31	8.08	19.91	56.67
BIMEANVAE	36.45	7.08	21.64	56.54	36.33	7.61	**20.50**	56.92
COOP + Attribute Model	Stepsize $\alpha = 0.3$							
Optimus	**36.43**	7.64	20.76	56.79	36.47	**8.19**	**20.21**	**56.78**
BIMEANVAE	36.55	**7.32**	**21.78**	56.43	36.33	7.53	20.39	**57.00**
COOP + Attribute Model	Stepsize $\alpha = 0.5$							
Optimus	35.84	6.93	19.75	56.23	**36.54**	8.05	19.84	56.74
BIMEANVAE	**36.58**	7.07	21.39	56.49	**36.45**	**7.67**	20.49	56.94
COOP								
Optimus	35.97	7.16	20.15	56.60	35.50	7.84	19.26	56.63
BIMEANVAE	35.67	6.53	21.07	56.27	36.16	7.25	20.09	56.78

Tabelle 4: ROUGE und Moverscore Ergebnisse auf den DEV-Benchmarkdatensätzen für das COOP Modell und das optimierte COOP Attribut-Modell. Die besten Ergebnisse je Modellgruppe sind fett markiert und die zweitbesten Ergebnisse unterstrichen.

Die in Tabelle 4 dargestellten Ergebnisse zeigen die Performance des kombinierten COOP + Attribut Modell im Vergleich zum normalen COOP Modell auf den Dev-Datensätzen. Die Ergebnisse zeigen eine Leistungssteigerung durch Verwendung des Attribut-Modells. Eine genaue Auswertung und Evaluierung findet in Abschnitt 8.3 auf den Test-Datensätzen statt.

Folgende in Tabelle 5 dargestellte Hyperparameter erzielen auf den Dev-Datensätzen die besten Ergebnisse und werden in dieser Form in der späteren Evaluation auf den Test-Datensätzen verwendet.

	Amazon					Yelp				
	α	R1(x)	R2(y)	RL(z)	MV(v)	α	R1(x)	R2(y)	RL(z)	MV(v)
Optimus	0.3	1.0	0.5	1.5	3.5	0.5	3.5	1.0	0.0	2.5
BIMEANVAE	0.3	3.5	0.0	1.0	0.5	0.5	0.5	0.0	0.0	1.0

Tabelle 5: Gewichtung der einzelnen Metriken zur Bestimmung des Input-Output-Overlap gesamt Scores

Aufgrund der individuellen Aspekte der verwendeten Modelle und Datensätzen ergeben sich jeweils unterschiedliche Hyperparameter, die die einzigartigen Eigenschaften der unterschiedlichen Kombinationen adaptieren.

8 Evaluierung der Modelle

Die unterschiedlichen Optimierungen der Latentvektoren für ein Optimus Modell und des Cellstates für BIMEANVAE werden auf dem Amazon und dem Yelp Datensatz verglichen und mit dem aktuellem State-of-the-Art Modell COOP in Relation gesetzt. Es existieren im Dev-Datensatz und Test-Datensatz jeweils 8 Eingabebewertungen und drei Gold-Summaries zum Evaluieren der generierten Ausgabebewertungen. Die Eingabebewertungen werden durch ein Optimus VAE Modell und ein BIMEANVAE Modell in Latentvektoren umgewandelt. Anschließend werden mittels in Abschnitt 6.1 dargestellter COOP Herangehensweise die Latentvektoren kombiniert, um den *Input-Output-Overlap* zu maximieren. Weiterhin werden die Latentvektoren mittels in dieser Arbeit eingeführtem Attribut-Modell optimiert, um detailreiche und umfassende Durchschnittsrezensionen zu erhalten.

8.1 Evaluationsmetriken

Die generierten Textbewertungen werden mit den drei Gold-Summaries verglichen. Zum Vergleich der generierten Rezensionen wird die ROUGE (**R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation)-Metrik verwendet (Lin, 2004). Die ROUGE-N Metrik misst die Anzahl der übereinstimmenden N-Grams zwischen dem generierten Text und den Referenztexten. Ein N-Gram ist eine N-lange sequentielle Folge von Wörtern innerhalb der Texte.

Zur Bewertung der generierten Rezensionen werden die vorhergesagten Ergebnisse mit den korrekten Ergebnissen verglichen. Die Konfusionsmatrix in Abbildung 9 ist eine Wahrheitsmatrix, welche die Einteilung der vorhergesagten Ergebnisse ermöglicht. True Positive (TP) und True Negative (TN) sind von dem Modell korrekt vorhergesagte Ergebnisse, False Positive (FP) und False Negative (FN) ist eine Klasse von falsch vorhergesagten Ergebnissen.

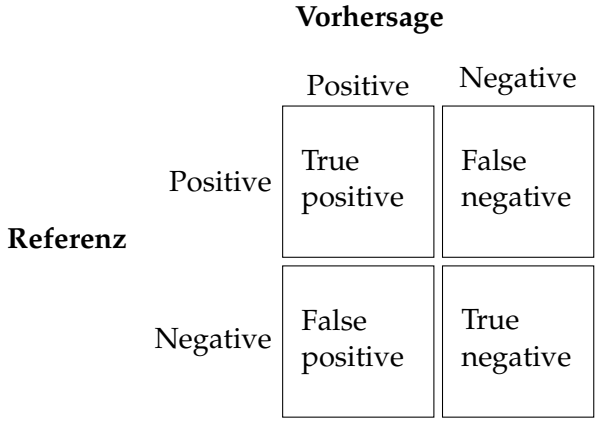

Abbildung 9: Konfusionsmatrix zur Berechnung des ROUGE-N Scores

Zur Berechnung des ROUGE-N Scores werden die einzelnen Textabschnitte in eine Menge aus N-Grams zerlegt. Mittels der Konfusionsmatrix in Abbildung 9 lassen sich Preci-

sion (P) und Recall (R) definieren:

Precision: Der Precision Wert ergibt sich aus dem Verhältnis der korrekt vorhergesagten N-Grams und der Anzahl der insgesamt vorhergesagten N-Grams.

$$P = \frac{TP}{TP + FP}$$

Recall: Recall ist als Verhältnis zwischen den korrekt vorhergesagten N-Grams und den N-Grams aus der Referenz definiert.

$$R = \frac{TP}{TP + FN}$$

F_1: Das F1-Maß beschreibt das harmonische Mittel zwischen Precision und Recall.

$$F_1 = \frac{2PR}{P + R}$$

In der Evaluation werden die ROUGE-1, ROUGE-2 und ROUGE-L Werte miteinander verglichen. ROUGE-1 verwendet als N-Gram Unigramme, ROUGE-2 Bigramme und ROUGE-L misst die längste gleiche Subsequenz zwischen Vorhersage und Referenz.

Da die ROUGE-Scores lediglich die einzelnen Wortsequenzen miteinander vergleichen, findet die semantische Bedeutung und Ähnlichkeit der Bewertungen mit der Referenz keinen Einfluss. Hier erzielen zum Beispiel Synonyme keine guten ROUGE-Scores, obwohl sie eine semantische Übereinstimmung haben. Um trotzdem die semantische Ähnlichkeit zwischen Bewertungen und Referenz zu messen, wird als weitere Metrik der Moverscore aus Abschnitt 8.2 verwendet. Der Moverscore basiert auf BERT und vergleicht Context-Embeddings mittels Earth-Mover-Distance (Rubner et al., 2000). Als Metrik konnte der Moverscore hohe Korrelationen mit menschlichem Urteilsvermögen aufweisen.

8.2 Moverscore

Der Moverscore (Zhao et al., 2019) ist eine Evaluationsmetrik, die semantische Inhalte zwischen zwei Textsequenzen vergleicht und diesen einen Ähnlichkeitswert zuweist. Das Ziel vom Moverscore ist es eine Metrik abzubilden, die einer menschlichen Bewertung der Ähnlichkeit von zwei Sequenzen am nähesten ist. Im Gegensatz zu anderen Textähnlichkeitsmetriken, die lediglich die Überlappungen von Tokens innerhalb der Sequenzen messen ohne die Semantik der Wörter zu bewerten, bildet sich der Moverscore aus einer Kombination bestehend aus einer im Kontext eingebetteten Repräsentation der einzelnen Textsequenzen, die eine semantische Distanz untereinander abbilden. Die semantische Distanz wird über die Word Mover Distance (Kusner et al., 2015), einer Metrik basierend auf der Earth Mover Distance, bestimmt. Es wird ein minimaler Transportfluss zwischen den einzelnen Sequenzen errechnet. Die Worteinbettungen werden durch ein BERT Modell erzeugt.

Insgesamt ist der Moverscore für die Bewertungsgenerierung ein wichtiger Leistungsindikator, da nicht nur übereinstimmende N-Gramme an Wörtern gemessen werden, sondern die Semantik der einzeln Wörter miteinbezogen wird. Da insbesondere in Bewertungen ähnliche Meinungen auf unterschiedliche Weise ausgedrückt werden können, bietet sich der Moverscore hier gut als Metrik an, um diese Übereinstimmungen zu finden, siehe Abschnitt 7.5.

8.3 Ergebnisse

In Tabelle 6 ist die Performance der unterschiedlichen untersuchten und erstellten Modelle dargestellt. In dieser Arbeit wurde das „*COOP*+Attribute Model“ Modell entwickelt. Es basiert auf dem *COOP* Modell und verbessert die Generierung von neuen Rezensionen durch die Verwendung eines Attribut-Modells. Unterschieden werden die *COOP* Modelle durch ihre grundlegend verwendete Variational Autoencoder Architektur in Optimus und BIMEANVAE, siehe Kapitel 4. Das Optimus Modell kombiniert BERT und GPT-2 in einem Variational Autoencoder Modell. BIMEANVAE hingegen besteht aus einem BiLSTM Encoder mit einem LSTM Decoder trainiert als Variational Autoencoder Modell. Ebenfalls werden die weiteren aktuellen Modelle LexRank, MeanSum und CopyCat in dem Vergleich miteinbezogen.

Verglichen wird die Performance der unterschieldichen Modelle mit den zuvor beschriebenen Evaluationsmetriken, dem ROUGE-1, ROUGE-2, ROUGE-L Score und dem Moverscore. Diese Metriken lassen ausreichend Rückschlüsse auf die erreichte Performance der Modelle und einer Leistungssteigerung zwischen den *COOP* Basismodellen und den modifizierten *COOP* mit Attributionsmodellen zu.

Grundsätzlich lassen sich die Ergebnisse in Tabelle 6 in die Kategorien abstraktive und extraktive Zusammenfassung unterteilen. Hier ist eindeutig zu erkennen, dass LexRank als extraktive Methode in allen Bereichen den abstraktiven Methoden unterliegt. Die Gruppe der abstraktiven Methoden umfasst die *SimpleAVG* Gruppe, die *COOP* Gruppe und die *COOP+Attribute Model* Gruppe, die alle auf Variational Autoencodern basieren. Diese Gruppen wurden nach der Kombinationsmethode der einzelnen Latentvektoren zu einem repräsentativen Latentvektor unterschieden.

Die *SimpleAvg* Gruppe umfasst unterschiedliche abstraktive Textzusammenfassungsmethoden auf Basis von Variational Autoencodern. Bei dieser Gruppe wird von allen erzeugten Latentvektoren ein normaler Durchschnittsvektor errechnet, von dem anschließend gesamplet wird. Es ist erkennbar, dass die beiden Methoden Optimus und BIMEANVAE den Methoden CopyCat und MeanSum überlegen sind, da diese in allen Messwerten bessere Ergebnisse erzielen. Besonders hervozuheben ist hier, dass Optimus bessere Moverscore Ergebnisse als BIMEANVAE erzielt, allerdings in den ROUGE Werten minimal schlechter abschneidet.

Eine große Leistungssteigerung ergibt sich durch die *COOP* Methode, die eine optimale Kombination der einzelnen Latentvektoren findet. Hier erzielen sowohl Optimus als auch BIMEANVAE in allen Metriken bessere Ergebnisse als die *SimpleAvg* Vergleichsgruppe. Demnach ist das Durchsuchen der Kombinationen von Latentvektoren sinnvoll. Insbesondere der ROUGE-1 Score übertrifft die *SimpleAVG* Scores bei Optimus und BI-

Test-Dataset	Amazon				Yelp			
Method	**R1**	**R2**	**RL**	**MV**	**R1**	**R2**	**RL**	**MV**
COOP + Attribute Model								
Optimus	35.68	**7.55**	20.68	**56.73**	33.85	6.98	18.82	56.47
BIMEANVAE	**37.94**	7.20	**21.75**	56.69	34.97	7.06	19.86	**56.90**
COOP								
Optimus ⋆	35.32	6.22	19.84	56.41	33.68	7.00	18.95	56.41
BIMEANVAE ⋆	36.57	7.23	21.24	56.49	**35.37**	**7.35**	**19.94**	56.78
SimpleAvg								
Optimus ⋆	33.54	6.18	19.34	56.49	31.23	6.48	18.27	23.05
BIMEANVAE⋆	33.60	6.64	20.87	20.85	32.87	6.93	19.89	22.41
CopyCat ⋆	31.97	5.81	20.16	-	29.47	5.26	18.09	-
MeanSum ⋆	29.20	4.70	18.15	-	28.46	3.66	15.57	-
Extractive								
LexRank ⋆	28.74	5.47	16.75	-	25.01	3.62	14.67	-

Tabelle 6: ROUGE und Moverscore Ergebnisse auf den Test-Benchmarkdatensätzen der unterschiedlichen Modelle. Die besten Ergebnisse sind fett markiert und die zweitbesten Ergebnisse unterstrichen. Der Stern ⋆ denotiert, dass die ROUGE-Ergebnisse aus den Ergebnissen von (Iso et al., 2021) übernommen wurden.

MEANVAE signifikant im Durchschnitt um 1.93%. Die größte Leistungssteigerung zwischen der *COOP* Kombinationsstrategie und *SimpleAvg* erfährt BIMEANVAE. Dies ist sehr beeindruckend, da BIMEANVAE mit 13 Millionen Parametern weitaus weniger Parameter hat als Optimus mit 239 Millionen Parametern und auch nicht auf vortrainierte Sprachmodelle zurückgreifen kann. Demnach lassen sich mittels Variational Autoencoder Textsequenzen hervorragend in Latentvektoren encodieren und diese mittels Vektoroperationen kombinieren.

Eine weitere Leistungssteigerung der verwendeten Variational Autoencoder Optimus und BIMEANVAE mit *COOP* Kombinatorik lässt sich durch die in dieser Arbeit verwendeten Attributmodelle feststellen. Die aus dem Latentvektoren decodierten Textsequenzen lassen sich so noch stärker in die gewünschte Richtung bei der Generierung adaptieren. Das verwendete Attributmodell ist ein Bag of Words Modell, welches aus den 150 am häufigsten vorkommenden Tokens besteht. Somit wird die Gewichtung bei der Generierung auf die häufig vorkommenden Tokens gelenkt und die generierten Textsequenzen haben eine höhere Überlappung. Beispielsweise fällt auf, dass beim Generieren teilweise mehrere vorgeschlagene Tokens ein hohes Ranking erhalten, wobei am Ende durch das Attributionsmodell das am besten passende mit einer höheren Wahrscheinlichkeit ausgewählt wird. Insbesondere spezifische Begriffe die ausschlaggebend für die erzeugten Rezensionen sind, allerdings in einer normalen Sprachverteilung eine geringe Gewichtung erhalten würden, werden durch das Bag of Words Attributmodell stark hervorgehoben.

Die Performance des kombinierten *COOP + Attributmodells* zeigt insbesondere auf dem

Amazon Datensatz im Optimus sowie auch BIMEANVAE Modell deutliche Verbesserungen gegenüber dem COOP Modellen. Die ROUGE Scores, sowie auch der Moverscore übertrifft für das jeweilige Modell mit Attributmodell die Scores des unoptimierten COOP Modells. Demnach konnte erfolgreich mittels Attributmodell die Generierung auf dem Amazon Datensatz optimiert werden.

Auf dem Yelp Datensatz zeigt sich keine deutliche Leistungssteigerung durch Verwendung des Attributmodells. Die erreichten Scores sind ähnlich wie die des unoptimierten COOP Modells. Das BIMEANVAE + Attributmodell erreicht minimal schlechtere Scores als das Baseline COOP Modell. In den Moverscores und den ROUGE-1 Score des Optimus Modell zeigt sich eine minimale Leistungssteigerung. Die fehlende Leistungssteigerung in den Metriken ist mit der Aufbau des Yelp Datensatzes erklärbar. Der Yelp Datensatz hat jeweils nur eine einzige Referenzrezension an der die Scores errechnet werden. Somit können durchaus Bewertungen, die semantisch den Inhalt perfekt wiedergeben, äußerst schlechte Scores erhalten, da keine überlappenden N-Gramme zu finden sind.

Um die Optimierungen des Attributmodells weiter zu untersuchen werden in Abschnitt 8.4 die Ergebnisse, die durch Auswahl der finalen Rezension aus den Rezensionen durch ein Orakel entstehen, ausgewertet. Somit lässt sich der Einfluss der Rankingfunktion auf die Ergebnisse ausschließen. Anschließend werden einzelne Rezensionen aus den beiden Datensätzen in Abschnitt 8.5 betrachtet und untereinander verglichen.

8.4 Vergleich der Modelle mit Rezensionsauswahl durch Orakel

Zum Vergleich von COOP + Attributmodell mit dem COOP Modell in Bezug auf die reine Generationsleistung, wurden die Rezensionen mittels Orakel ausgewählt, um den Einfluss der Rankingfunktion zu unterdrücken. Bei der Auswertung in Abschnitt 8.3 wurden die Rezensionen anhand einer Rankingfunktion in Bezug zu den Eingaberezensionen ausgewählt. In diesem Vergleich werden die Rezensionen anhand eines Orakels, welches die Rezensionen, die mit den Gold-Zusammenfassungen die größte ROUGE-1 Übereinstimmung haben, ausgewählt werden. Somit sind die Ergebnisse in diesem Vergleich gleichzeitig eine obere Schranke für die Leistungsfähigkeit der entsprechenden Modelle.

Test-Dataset	Amazon				Yelp			
Method	**R1**	**R2**	**RL**	**MV**	**R1**	**R2**	**RL**	**MV**
COOP + Attribute Model								
Optimus	40.75	<u>8.82</u>	21.54	56.88	<u>43.81</u>	<u>11.08</u>	22.43	57.34
BIMEANVAE	**42.15**	8.67	**23.52**	**57.18**	**45.13**	**11.23**	**24.77**	**57.82**
COOP								
Optimus	39.84	7.88	21.40	<u>57.02</u>	42.05	10.03	22.24	57.32
BIMEANVAE	<u>40.80</u>	**8.99**	<u>23.48</u>	56.83	42.72	10.21	<u>24.00</u>	<u>57.44</u>

Tabelle 7: ROUGE und Moverscore Ergebnisse auf den Test-Benchmarkdatensätzen der unterschiedlichen Modelle mit Auswahl der generierten Rezensionen durch ein Orakel. Die besten Ergebnisse sind fett markiert und die zweitbesten Ergebnisse unterstrichen.

In Tabelle 7 sind die Ergebnisse für die COOP + Attributionsmodell Modelle und die COOP Modelle mit Rezensionsauswahl durch ein Orakel dargestellt. In allen Metriken sind hier die COOP + Attributionsmodell Ergebnisse den COOP Modell Ergebnissen überlegen. Die Ergebnisse des BiMeanVAE + Attributionsmodell Modells übertreffen die Ergebnisse des COOP Modells auf beiden Datensätzen in allen Metriken stark, bis auf den ROUGE-2 Wert des Amazon Datensatzes. Das Optimus + Attributionsmodell Modell übertrifft ebenfalls die Ergebnisse des COOP Modells, außer dem Moverscore auf dem Amazon und den ROUGE-L Score auf dem Yelp Datensatz.

Somit lässt sich eine starke Leistungssteigerung durch Verwendung des Attributmodells feststellen. Das Attributmodell ermöglicht bei der Generierung die Gewichtung von Tokens im Bag Of Words des Attributmodells zu forcieren, wodurch wiederum die exakten Wörter der Eingaberezensionen verwendet werden. Die Metriken lassen ebenfalls Rückschlüsse auf eine bessere Anpassungsfähigkeit durch das Attributmodell zu, da in fast allen Metriken das Referenzmodell übertroffen wurde. Insbesondere auch der Moverscore, der die semantische Ähnlichkeit abbildet, zeigt eine Leistungssteigerung durch das Attributmodell. Zwischen den beiden Modellen Optimus und BiMeanVAE ist BiMeanVAE mit Attributionsmodel das Modell mit den besten Ergebnissen.

8.5 Beispiele für generierte Rezensionen

Folgend werden einige generierte Textrezensionen des COOP Modells und des COOP + Attributionsmodell Modells direkt miteinander verglichen. Es wurden jeweils zwei generierte Rezensionen des COOP und des COOP + Attributionsmodell Modells zu den Datensätzen Amazon und Yelp ausgewählt. Zur besseren Darstellung sind bei den generierten Rezensionen die mit der Gold-Zusammenfassung übereinstimmenden Unigrams rot markiert, Bigrams sind farblich blau hinterlegt und die längste übereinstimmende Textsequenz unterstrichen. Der Moverscore lässt sich nicht visualisieren.

Authentische Rezensionen sollten konsistent in ihrem Inhalt sein, präzise auf Eigenschaften und Aspekte der Produkte oder Dienstleistungen eingehen und der Inhalt der generierten Rezension sollte mit den Eingabebewertungen deckungsgleich sein. Anhand dieser Kriterien werden nachfolgend mehrere generierte Rezensionen bewertet.

8.5.1 Amazon Rezensionen

Nachfolgend werden zwei generierte Rezensionen des Amazon Datensatzes evaluiert. Zu den Amazon Pordukten existieren jeweils drei Referenzzusammenfassungen, wodurch eine bessere Vergleichbarkeit mittels Metriken möglich ist.

Die in Rezension 1 generierten Rezensionen für eine Halskette des Amazons Datensatzes zeigen insgesamt gute Ergebnisse. Der Inhalt der beiden durch BiMeanVAE generierten Rezensionen ist konsistent. Das Sentiment ist positiv und die Kette wird abschließend zum Kaufen empfohlen. Besonders auffällig ist die höhere Präzision des Attributmodells. Das Produkt und die Eigenschaften dieses werden explizit erwähnt, wie zum Beispiel „great quality“, „perfect size“, „very nice looking necklace“, „color is beautiful“ und „perfect gift for the price“. Im Gegensatz dazu ist die vom COOP Modell generierte

Produkt: Sterling Silver „Love“ Open Heart Pendant Necklace, 18“

BIMEANVAE COOP+Attribute-Model: This necklace is a great quality and so far it’s the perfect size. It is a very nice looking necklace, and the color is beautiful . The chain is a perfect gift for the price of the necklace. I love it!
Scores: R-1: 48.63, R-2: 16.78, R-L: 28.08, MV: 60.05

BIMEANVAE COOP: It was a perfect gift for a friend of her birthday. She loves it, the look is great, and the price was great. It was easy to set up, and she loves it. I would recommend this product to anyone .
Scores: R-1: 33.56, R-2: 5.71, R-L: 20.27, MV: 56.00

Optimus COOP+Attribute-Model: This is a great gift for the price. She loves it, so much better than the necklace and it looks beautiful. You can’t get a gift on the chain , but it is not worth the money! It looks great!
Scores: R-1: 42.65, R-2: 9.28, R-L: 26.57, MV: 58.31

Optimus COOP: Its a beautiful necklace, and it looks great for the price. She said it was a gift for her birthday present, but it is very thin and not worth the money. If you want to go wrong with this necklace!
Scores: R-1: 37.76, R-2: 8.57, R-L: 21.67, MV: 57.69

Rezension 1: Vergleich der generierten Rezensionen zwischen dem COOP und COOP + Attributionsmodell zu Produkt B0040EIHQQ des Amazon Datensatzes

Rezension allgemeiner und erwähnt lediglich das gute Aussehen der Halskette „the look is great“ und den Preis „price was great“ des Produktes ohne auf spezifische Eigenschaften des Produktes wie zum Beispiel die Farbe einzugehen. Diese Auswertung spiegelt sich auch in den Metriken wieder, in denen das Attributmodell die Metriken des COOP Modells in allen Werten übertrifft. Insbesondere der Moverscore, der die semantische Ähnlichkeit zu den Gold-Zusammenfassungen angibt, ist mit 60.05 beim Attributmodell bedeutend höher als die 56.00 des COOP Modells. In dieser Beispielrezension bestätigt sich die Hypothese, dass das Attributmodell durch geeignete Tokens in einem Bag of Words Modell die Generierung präzise konditionieren können.

Die durch Optimus erstellten Rezensionen sind sowohl beim COOP, wie auch beim Attributmodell inkonsistent in ihrem Inhalt. Die Rezensionen haben beide hauptsächlich ein positives Sentiment, widersprechen sich allerdings beide jeweils mit den Formulierungen „great for the price“ und „not worth the money“. Trotzdem gehen beide Modelle auf unterschiedliche Aspekte der zu bewertenden Halskette ein und beschreiben den Stil der Kette als „it looks beautiful“,„It looks great!“ und „a beautiful necklace“. Eine Steigerung der Leistung der generierten Rezensionen des Attributmodells gegenüber dem COOP Modell kann hier nur an den Metriken festgestellt werden und spiegelt sich nur

minimal in der generierten Rezension wieder. Hier fällt auf, dass das Attributmodell versucht die Kette „the chain“ der Halskette anzusprechen, allerdings keinen semantisch korrekten Satz mit diesen Tokens bildet.

Produkt: Coaster CO-150430 5 Pc Dining Set, Chestnut

BiMeanVAE COOP+Attribute-Model: The table is very nice and sturdy. It was easy to put together and the color is perfect. The chairs are a bit small for the price, but it's not a big deal. It was very easy to assemble .
Scores: R-1: 42.75, R-2: 7.03, R-L: 23.66, MV: 54.49

BiMeanVAE COOP: The table is very nice and sturdy. The chairs are easy to put together and the color is great. It was easy to assemble, and the price was right. The size is perfect for a small kitchen, so it's not a big deal.
Scores: R-1: 41.60, R-2: 9.70, R-L: 24.81, MV: 54.53

Optimus COOP+Attribute-Model: Just received this table top and sturdy. The table is perfect for the price, so it was easy to assemble. One of the chairs are not sturdy, sturdy and the table was not worth the money. Other than that it is a great purchase.
Scores: R-1: 32.85, R-2: 5.22, R-L: 20.44, MV: 53.52

Optimus COOP: Just received this table and sturdy. The table is perfect for the price, and it was easy to assemble. One of the chairs are not perfect. It is sturdy and not worth the money.
Scores: R-1: 30.32, R-2: 6.72, R-L: 19.67, MV: 53.49

Rezension 2: Vergleich der generierten Rezensionen zwischen dem COOP und COOP+Attributmodell zu Produkt B001EQJ5AU des Amazon Datensatzes

Rezension 2 bewertet ein Set bestehend aus einem Tisch und 4 Stühlen. Die durch BiMeanVAE generierten Rezensionen sind in ihrem Inhalt konsistent und beschreiben präzise den Inhalt des Sets. Beide Rezensionen erwähnen die Eigenschaften „easy to put together“, „easy to assemble“, die Farbe und die Qualität des Sets. Somit werden die einzelnen Aspekte des Produktes in den beiden Rezensionen aufgegriffen. Anschaulich ist die Optimierung des Attributmodells die Phrase „it's not a big deal“ semantisch korrekt zu Integrieren und den Bezug zur Größe der Stühle zu setzen. Das COOP Modell widerspricht sich hier in der Semantik mit „The size is perfect for a small kitchen, so it's not a big deal“, da kein negativer Punkt angesprochen wird. In den Metriken erzielen beide Modelle gute Ergebnisse, wobei das Attributmodell einen höheren ROUGE-1 Score hat und das COOP Modell in den anderen Metriken bessere Ergebnisse erzielt.

Die durch Optimus erzeugten Rezensionen beschreiben ebenfalls den Inhalt des Sets und erwähnen die Eigenschaften „easy to assemble“ und die Qualität. Teilweise sind die erzeugten Sätze syntaktisch nicht korrekt. In diesem Beispiel ist erkennbar, dass das Attributmodell höchstwahrscheinlich ab der Stelle „One of the chairs“ die Tokens bei der Generierung stakr beeinflusst hat und somit die Semantik des verbleibenden Teils positiv beeinflusst hat. In den Metriken erzielt das Attributmodell in allen Metriken, außer dem

ROUGE-2 Score, bessere Ergebnisse.

Insgesamt zeigt auf dem Amazon Datensatz das Attributmodell mit BIMEANVAE auf beiden Rezensionen 1 und 2 die besten Ergebnisse in Bezug auf konsistenten Inhalt, Semantik und präzisen Formulierungen der wichtigen Aspekte der Produkte. Durch das Attributmodell sind die BIMEANVAE Rezensionen wesentlich ausdrucksstärker, da seltene Wörter, die in Bezug auf die Rezensionen eine hohe Relevanz haben, häufiger miteinbezogen werden.

8.5.2 Yelp Rezensionen

Nachfolgend werden zwei generierte Rezensionen des Yelp Datensatzes evaluiert. Bei dem Yelp Datensatz ist zu beachten, dass je Dienstleisung jeweils nur eine Referenzzusammenfassung existiert. Demnach ist eine Vergleichbarkeit an Metriken nur unzureichend möglich und muss mit Vorsicht betrachtet werden, da insbesondere bei Textsequenzen Rouge Metriken die Semantik der Textsequenzen nicht erfassen.

Restaurant: Harbour 60 Steakhouse

Referenz: This place has some amazing, tasty steaks! They are breathtaking! The menu also has seafood options that are just as delicious! The only downside is that its pretty expensive, however you get top quality meat for what you pay for. You won't even notice the price when your stomach is filled with the best seasoned meat you've ever had.

BiMeanVAE COOP+Attribute-Model: This place is a great place to eat. The food is very good and the service is excellent. It's a little pricey but it's worth it for the quality of the food and the service. The steak was cooked to perfection and the lobster bisque was delicious. The best part of the meal was the wine.
Scores: R-1: 21.85, R-2: 5.13, R-L: 15.13, MV: 56.57

BiMeanVAE COOP: Excellent food and great service. The staff is very friendly and the food is delicious. The steak was cooked to perfection and the service was a little slow but it was worth the wait. It's a great place to go for a date night.
Scores: R-1: 18.86, R-2: 1.92, R-L: 9.43, MV: 55.69

Optimus COOP+Attribute-Model: One of the best steakhouse. It's a great meal. The service was amazing and the food was delicious. If you're looking for a long time to eat at the strip. You can't wait for dinner. This is a little pricey for the steak and it was perfect. Highly recommend this place to anyone in town.
Scores: R-1: 26.89, R-2: 3.42, R-L: 15.13, MV: 57.28

Optimus COOP: This restaurant is a great place to die for. The food is amazing and the service is fantastic. It's a little pricey but the food was delicious. Everything was perfect for the steak and delicious! The service was great, the best part of the restaurant in the city. If you want to go back for dinner, it's not worth it.
Scores: R-1: 24.39, R-2: 3.30, R-L: 16.26, MV: 57.52

Rezension 3: Vergleich der generierten Rezensionen zwischen dem COOP und COOP+Attributmodell zu Dienstleistung 4POPYEONJpkfhWOMx_PyGg des Yelp Datensatzes

Rezension 3 bewertet ein Steakhouse in Toronto. Insgesamt generieren alle Modelle gute Rezensionen und geben den Inhalt der Referenzbewertung wieder. Bei den durch BiMeanVAE generierten Bewertungen ist der Inhalt konsistent, das Essen und der Service wird als exzellent bewertet. Die durch das Attributmodell erzeugte Rezension ist detaillierter und erwähnt zusätzlich noch den Wein und Hummer, der auch in der Referenz vorhanden ist. Ebenfalls erwähnt das Attributmodell die hohen Kosten, die auch in der Referenz vorkommen. Die höhere Deckungsgleichheit ist auch in den Metriken wieder-

zufinden, in denen das Attributmodell in allen Metriken dem COOP Modell überlegen ist.

Die Rezensionen der Optimus Modelle sind Inhaltlich korrekt und erwähnen beide, dass es sich um ein leckers Steakhouse mit gutem Service handelt. Das COOP Modell erzeugt in dieser Rezension teilweise untypische Formulierungen wie: „This restaurant is a great place to die for“. In den Metriken sind die erzeugten Rezensionen der beiden Modelle ähnlich.

Restaurant: Anna Marie's Italian Cuisine

Referenz: Great place to go food authentic Italian food. Pizza is amazing, lots of topping choices and they aren't stingy with them. Drinks and desert wont disappoint either. The atmosphere is pleasant, and provides a nice escape from the heat. Staff can be hit or miss but the place is clean and they have some good happy hour deals.

BiMeanVAE COOP+Attribute-Model: Had a great experience here. The food was very good and the service was excellent. It's a nice place to go with a group of friends and family. They have a great happy hour menu and the food is very good. If you are in the area, this is a must try.
Scores: R-1: 41.07, R-2: 9.09, R-L: 21.43, MV: 57.87

BiMeanVAE COOP: Came here for the first time and it was a great experience. The food was delicious and the staff was very friendly. The atmosphere is nice and the service was good. It's a great place to go if you are in the area.
Scores: R-1: 33.00, R-2: 9.90, R-L: 17.47, MV: 56.94

Optimus COOP+Attribute-Model: Came here for a few weeks ago. It was very friendly and the food was delicious. They have a great experience. The food is very nice and it was a bit of the best pizza. If you want to go back to try the menu. Definitely will definitely be back and try this place.
Scores: R-1: 35.40, R-2: 3.60, R-L: 15.93, MV: 56.91

Optimus COOP: Absolutely love this place! The food was very friendly and the service was excellent. The staff is very nice and attentive. They have a nice atmosphere and made it out of the menu. It's a must try to come back. Definitely will definitely be back!
Scores: R-1: 34.28, R-2: 3.88, R-L: 19.04, MV: 56.75

Rezension 4: Vergleich der generierten Rezensionen zwischen dem COOP und COOP+Attributmodell zu Dienstleistung 6jDD-Z8QcsKTdIDWwM8gog des Yelp Datensatzes

In der Rezension 4 wird ein italienisches Restaurant bewertet. Die Rezensionen von allen Modellen geben ein positives Feedback über das italienische Restaurant. Die Refrenzbewertung ist allerdings wesentlich spezifischer als BiMeanVAE oder Optimus in der Beschreibung des Restaurants. BiMeanVAE erzeugt, sowohl als Attributmodell wie auch als COOP Modell eine in sich schlüssige Rezension. Die Metriken des Attributmodells sind minimal höher als die des COOP Modells.

Das Optimus Modell mit Attributionsmodell ist das einzige Modell, welches das Gericht “best pizza„ erwähnt. Das normale Optimus Modell begeht den semantischen Fehler das Essen als freundlich zu beschrieben. Trotzdem erzeugen beide Modelle akzeptable Rezensionen und sind in den Metriken untereinander ausgeglichen.

Insgesamt sind die generierten Rezensionen für die beiden Restaurants des BIMEAN-VAE + Attributmodell Modells am präzisesten und im Inhalt am umfangreichsten. Die BIMEANVAE Modelle zeigen allgemein eine bessere Fähigkeit aus den Latentvektoren Text zu generieren und haben weniger semantische Fehler. Durch das Attributmodell in Verbindung mit dem BIMEANVAE werden bei der Generierung noch spezifischere Details erwähnt, die durch das erneute Ranking der Tokens in Bezug auf das Attributmodell erzeugt werden.

9 Zusammenfassung und Ausblick

In dieser Masterarbeit konnte ein umfassender Überblick über State-of-the-Art NLP-Techniken im Bereich der Multi-Document Summarization gegeben werden. Es wurden unterschiedliche Architekturen wie Transformer, BERT, GPT-2, LSTM und Attention-Layer dargestellt. Insbesondere Variational Autoencoder, Optimierungen dieser Modelle und die Verwendung dieser im Bereich von Sprachmodellen wurde explizit erläutert.

Das Ziel dieser Arbeit war das Zusammenfassen von mehreren Rezensionen eines Produkts oder einer Dienstleistung zu einer generierten repräsentativen Rezension, die die anderen Rezensionen inkludiert und auf die verschiedenen wichtigen Aspekte der zu bewertenden Produkte oder Dienstleistung eingeht.

Als Datensätze wurden Amazon und Yelp-Review Datensätze verwendet. Diese bieten eine Vielzahl an unterschiedlichen Rezensionen zu unterschiedlichen Produkten und Dienstleistungen. Insbesondere zur Multi-Review Summarization existieren menschlich erstellte Rezensionszusammenfassungen auf beiden Datensätzen.

Weiterhin wurden unterschiedliche extraktive und abstraktive Methoden zur Multi-Review Summarization vorgestellt und untereinander verglichen. Die in dieser Arbeit verwendete Convex Aggregation for Opinion Summarization Methode basiert darauf, dass generierte Rezensionen aus normalen Durchschnittslatentvektoren ihre Expressivität verlieren. Die *COOP* Methode findet eine optimale Kombination der einzelnen Latentvektoren für einen Durchschnittslatentvektor, um eine maximale Expressivität beizubehalten.

In dieser Masterarbeit wurde die *COOP* Methode durch ein Bag of Words Attributmodell weiter optimiert, um bei der Generierung von Rezensionen die berechneten Wahrscheinlichkeiten der Tokens in Bezug auf das gewählte Attributmodell zu optimieren. Das Attributmodell wurde bei dem Optimus Modell in die Vergangenheitsmatrix von GPT-2 injiziert. Bei dem BIMEANVAE Modell wurde bei der Generierung jeweils der vorherige Cellstate des LSTM-Decoders optimiert.

Durch Anwendung der Beam Search Methode auf den durch das Attributmodell optimierten Latentvektoren konnten erfolgreich präzise Zusammenfassungen der Rezensionen generiert werden. Des Weiteren wurde ein neuartiger Rankingalgorithmus zur Auswahl der besten generierten Rezension auf Basis der Kombination von mehreren ROUGE-Werten und dem Moverscore erstellt.

In der Evaluation zeigt das in dieser Masterarbeit entwickelte *COOP*+Attributmodell eine hervorragende Performance bei der Zusammenfassung von Rezensionen. Die wichtigen Aspekte der unterschiedlichen Rezensionen werden wiedergegeben und insbesondere durch das Attributmodell zeigt sich eine Steigerung der Präzision bei den generierten Rezensionen. Im Vergleich zu den bisherigen Modellen übertrifft das *COOP*+Attributmodell die Performance der anderen Modelle in allen gewählten Metriken und erzielt demnach State-of-the-Art Ergebnisse.

Diese Ergebnisse bilden eine Grundlage für weiterführende Untersuchungen inwiefern Variational Autoencoder durch Verwendung von Attributmodellen bei der Generierung unterstützt werden können. Ebenfalls wäre eine Auswertung der unterschiedlichen Me-

thoden mittels menschlicher Evaluation sinnvoll, insbesondere auch um den Einfluss des Moverscore gestützten Ranking auf die empfundene Übereinstimmung von menschlichen Evaluatoren zu erörtern. Somit könnte das in dieser Masterarbeit erstellte Modell verwendet werden, um auf Webportalen Benutzern einen schnellen allgemeingültigen Überblick über unterschiedliche Rezensionen von Produkten und Dienstleistungen zu ermöglichen.

Literaturverzeichnis

Dor Bank, Noam Koenigstein und Raja Giryes (2020). „Autoencoders“. In: *CoRR* abs/2003.05991. arXiv: 2003.05991.

Arthur Brazinskas, Mirella Lapata und Ivan Titov (2019). „Unsupervised Multi-Document Opinion Summarization as Copycat-Review Generation“. In: *CoRR* abs/1911.02247. arXiv: 1911.02247.

Arthur Bražinskas, Mirella Lapata und Ivan Titov (Juli 2020). „Unsupervised Opinion Summarization as Copycat-Review Generation“. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, S. 5151–5169.

Ricardo Campos, Vitor Mangaravite, Arian Pasquali, Alipio Jorge, Celia Nunes und Adam Jatowt (2020). „YAKE! Keyword extraction from single documents using multiple local features“. In: *Information Sciences* 509, S. 257–289.

Eric Chu und Peter Liu (Juni 2019). „MeanSum: A Neural Model for Unsupervised Multi-Document Abstractive Summarization“. In: *Proceedings of the 36th International Conference on Machine Learning*. Hrsg. von Kamalika Chaudhuri und Ruslan Salakhutdinov. Bd. 97. Proceedings of Machine Learning Research. PMLR, S. 1223–1232.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski und Rosanne Liu (2019). „Plug and Play Language Models: A Simple Approach to Controlled Text Generation“. In: *CoRR* abs/1912.02164. arXiv: 1912.02164.

Jacob Devlin, Ming-Wei Chang, Kenton Lee und Kristina Toutanova (2018). „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“. In: *CoRR* abs/1810.04805. arXiv: 1810.04805.

Günes Erkan und Dragomir R. Radev (2011). „LexRank: Graph-based Lexical Centrality as Salience in Text Summarization“. In: *CoRR* abs/1109.2128. arXiv: 1109.2128.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz und Lawrence Carin (2019). „Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing“. In: *CoRR* abs/1903.10145. arXiv: 1903.10145.

Sepp Hochreiter und Jürgen Schmidhuber (Dez. 1997). „Long Short-term Memory“. In: *Neural computation* 9, S. 1735–80.

Hayate Iso, Xiaolan Wang, Yoshihiko Suhara, Stefanos Angelidis und Wang-Chiew Tan (Nov. 2021). „Convex Aggregation for Opinion Summarization“. In: *Findings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jeremy Jordan (Juli 2018). *Variational autoencoders.*

Diederik P Kingma und Max Welling (2014). *Auto-Encoding Variational Bayes*. arXiv: 1312.6114 [stat.ML].

Diederik P. Kingma und Max Welling (2019). „An Introduction to Variational Autoencoders“. In: *CoRR* abs/1906.02691. arXiv: 1906.02691.

Matt Kusner, Yu Sun, Nicholas Kolkin und Kilian Weinberger (Juli 2015). „From Word Embeddings To Document Distances“. In: *Proceedings of the 32nd International Conference on Machine Learning*. Hrsg. von Francis Bach und David Blei. Bd. 37. Proceedings of Machine Learning Research. PMLR, S. 957–966.

Chunyuan Li, Xiang Gao, Yuan Li, Xiujun Li, Baolin Peng, Yizhe Zhang und Jianfeng Gao (2020). „Optimus: Organizing Sentences via Pre-trained Modeling of a Latent Space“. In: *CoRR* abs/2004.04092. arXiv: 2004.04092.

Chin-Yew Lin (Juli 2004). „ROUGE: A Package for Automatic Evaluation of Summaries“. In: *Text Summarization Branches Out*. Association for Computational Linguistics, S. 74–81.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado und Jeffrey Dean (2013). „Distributed Representations of Words and Phrases and their Compositionality“. In: *CoRR* abs/1310.4546. arXiv: 1310.4546.

Christopher Olah (o. D.). *Understanding LSTM networks*.

Jeffrey Pennington, Richard Socher und Christopher D Manning (2014). „Glove: Global Vectors for Word Representation.“ In: *EMNLP*. Bd. 14, S. 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee und Luke Zettlemoyer (2018). „Deep contextualized word representations“. In: *CoRR* abs/1802.05365. arXiv: 1802.05365.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei und Ilya Sutskever (2019). „Language Models are Unsupervised Multitask Learners“. In.

Yossi Rubner, Carlo Tomasi und Leonidas Guibas (Nov. 2000). „The Earth Mover's Distance as a Metric for Image Retrieval“. In: *International Journal of Computer Vision* 40, S. 99–121.

„TF–IDF“ (2010). In: *Encyclopedia of Machine Learning*. Hrsg. von Claude Sammut und Geoffrey I. Webb. Springer US, S. 986–987.

Rico Sennrich, Barry Haddow und Alexandra Birch (2015). „Neural Machine Translation of Rare Words with Subword Units“. In: *CoRR* abs/1508.07909. arXiv: 1508.07909.

Ilya Sutskever, Oriol Vinyals und Quoc V. Le (2014). „Sequence to Sequence Learning with Neural Networks“. In: *CoRR* abs/1409.3215. arXiv: 1409.3215.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser und Illia Polosukhin (2017). „Attention Is All You Need“. In: *CoRR* abs/1706.03762. arXiv: 1706.03762.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes und Jeffrey Dean (2016). „Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation“. In: *CoRR* abs/1609.08144. arXiv: 1609.08144.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer und Steffen Eger (2019). „MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance“. In: *CoRR* abs/1909.02622. arXiv: 1909.02622.

Abbildungsverzeichnis

Tabellenverzeichnis