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:	20. August 2021
Abgabe der Arbeit:	31. September 2010
Gutachter:	Prof. Dr. Stefan Conrad
	Prof. Dr. Gunnar Klau

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 31. September 2010

_____

Lionel Schockenhoff

Zusammenfassung

Hier kommt eine ca. einseitige Zusammenfassung der Arbeit rein.

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, kostenspielig vortrainierte Modelle mit vielen Parametern durch Transfer Learning in diversen speziellen Aufgabenbereichen einsetzen.

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 NLP (**N**atural **L**anguage **P**rocessing) Bereich gibt es viele unterschiedliche Ansätze zur Textzusammenfassung von einzelnen Dokumenten. Da des Öfteren unterschiedliche Dokumente mit diversen Inhalten zu identischen Themen existieren, stellt sich die Frage wie diese unterschiedlichen Dokumente zu einem umfassenden Dokument zusammengefasst werden können.

Als Datengrundlage gelten hier der Amazon Review und Yelp Review Datensatz, der zu Produkten oder Restaurants mehrere unterschiedliche Bewertungen liefert. Ziel dieser Masterarbeit ist es einen Variational Auto Encoder unter Verwendung von Transformern zu entwickeln, der die entsprechend unterschiedlichen Bewertungen zu einem Produkt beziehungsweise Restaurant abstraktiv zusammenfasst. In der Zusammenfassung sollen sich möglichst viele Aspekte der ursprünglichen Bewertungen wiederfinden.

1.2 Ziel und Aufbau der Arbeit

Das Ziel dieser Arbeit ist mehrere Dokumente abstraktiv zu einem Dokument zusammenzufassen. Hierzu wird der Variational Auto Encoder OPTIMUS, der auf BERT und GPT-2 basiert verwendet.

1.3 Terminologie

Adaptive Moment Estimation (Adam) ist ein Optimierer für neuronale Netze.

Fine-Tuning ist ein Trainingsprozess, der ein bereits vortrainiertes neuronales Netz an eine spezielle Aufgabenstellung anpasst und auf diese trainiert. Die bereits im Pre-Training gefundenen Parameter werden weiter angepasst.

Global Vectors for Word Representation (GloVe) ein Verfahren für das Zuweisen von Vektorrepräsentationen zu Worten.

JavaScript Object Notation (JSON) ist ein in Textform vorliegendes Datenformat.

Natural Language Processing (NLP) beschreibt das maschinelle Verarbeiten und Analysieren von natürlichen Sprachen unter Verwendung von unterschiedlichen Techniken und Verfahren.

Pre-Training beschreibt den initialen Trainingsprozess eines neuronalen Netzes. Im Bereich des NLP wird Pre-Training verwendet, um ein allgemeines Sprachmodell zu trainieren, welches später auf individuelle Aufgabenstellungen nachtrainiert wird.

Tokenisierung ist die Segmentierung eines Textes in eine Folge von Tokens. Die einzelnen Tokens sind Zeichenketten bestehend aus einem oder mehreren Zeichen.

2 Grundlagen

In diesem Kapitel werden die Grundlagen zu den verwendeten Technologien erklärt. Zunächst wird die in dieser Masterarbeit untersuchte Aufgabe des abstrakten Zusammenfassens erläutert. Anschließend werden die Grundlagen zu neuronalen Netzen und insbesondere zu Deep Learning erklärt.

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

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 akkurate Zusammenfassungen und Überblicke zu geben. Automatisierte Textzusammenfassung lässt sich zum Beispiel bei der Generierung von Kurzzusammenfassungen in Zeitungen oder zur Generierung von Überschriften einsetzen.

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 im 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 reproduzieren. Die Zusammenfassung soll alle wichtigen Informationen enthalten und als zusammenhängenden flüssigen 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 Übergangsrepresentation und generieren aus dieser durch Decodierung eine Ausgangsrepresentation.

2.1.3 Multi-Document Textzusammenfassung

Eine große Herausforderung ist das Zusammenfassen von mehreren Dokumenten zum selben Thema zu einem einzigen Dokument. Die entstehende Zusammenfassung soll Anwendern einen guten und schnellen Überblick über eine große Anzahl an Dokumenten bieten. Die unterschiedlichen Dokumente enthalten diverse Informationen die nicht immer deckungsgleich sind. Somit ergibt sich die Herausforderung unterschiedliche Perspektiven in den jeweiligen Dokumenten zusammenfassend zu representieren. Da sich unterschiedliche Standpunkte schlecht mittels extraktiven Methoden darstellen lassen, bieten sich bei der Multi-Document Textzusammenfassung abstraktive Methoden an.

In dieser Masterarbeit werden unterschiedliche Bewertungen zu Produkten oder Restaurants zusammengefasst. Insbesondere Bewertungen unterscheiden sich stark in Ihren Standpunkten 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 Produktbewertungen eine allgemeingültige zusammenfassende Bewertung zu produzieren, die klar strukturiert, kongruent und gut lesbar die entsprechenden Inhalte wiedergibt.

2.2 Deep Learning

2.3 Word Embeddings

3 Transformer

Transformer sind eine spezifische 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 auf den unterschiedlichsten Benchmarks hervorragende Ergebnisse.

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 Self-Attention-Layer

Durch die Self-Attention-Layer von Transformermodellen lassen sich relevante Informationen erkennen und das Modell sich auf diese fokussieren. Der Self-Attention-Layer innerhalb des Transformers berechnet die Relevanz von Values zu bestimmten Keys und Queries. Die Query- (Q), Key- (K) und Value- (V) Vektoren werden aus dem Eingabevektor (X) durch Multiplikation mit erlernten Gewichtsmatrizen (W) generiert (**AttentionIALYN**).

$$Q = X \times W^Q, K = X \times W^K, V = X \times W^V \tag{1}$$

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 angibt (**AttentionIALYN**).

$$Attention(Q, K, V) = Softmax(\frac{Q \times K^T}{\sqrt{d_k}}) \times V \tag{2}$$

In Gleichung 2 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.

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

$$\begin{aligned} MultiHeadAttention(Q, K, V) &= [head_1, \ldots, head_h] \times W^O \\ &\text{mit } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{3}$$

In Gleichung 3 wird deutlich, dass die unterschiedlichen Attention-Heads voneinander unabhängig sind. Hierdurch können sich die einzelnen Attention-Heads auf unterschiedliche Merkmale in den Daten fokussieren.

3.2 Transformer Encoder / Decoder

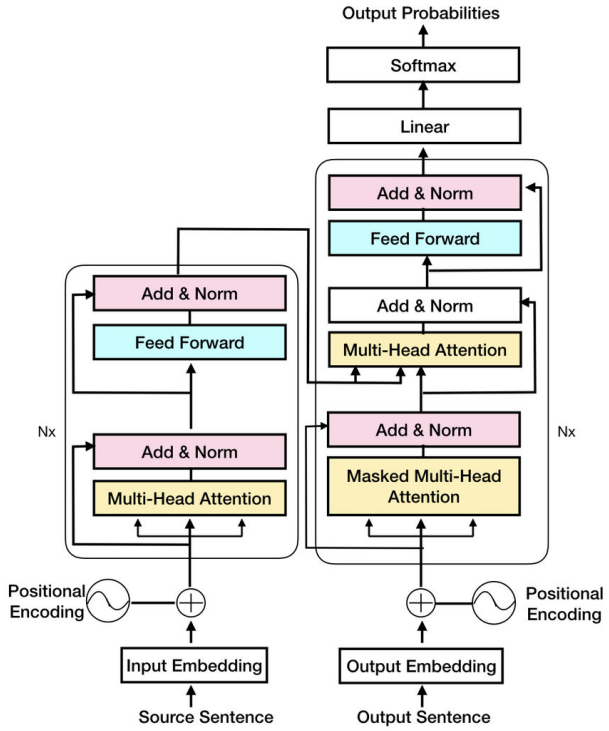


Abbildung 1: Transformer Encoder (links) und Transformer Decoder (rechts)

3.3 Transformer-Decoder

3.4 Bidirectional Encoder Representations from Transformers

BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) ist ein von Google (**DBLP:journals/corr/abs-1810-04805**) entwickeltes Sprachmodell, welches sich durch die bidirektionale kontextuelle Einbettung von Worten auszeichnet. Bei der Veröffentlichung von BERT konnten in vielen NLP Benchmarks Bestleistungen erzielt werden.

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 Transformerencoderblocks 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 ein- seitig, beziehungsweise kombinieren zwei einseitige Betrachtungen wie zum Beispiel ELMO 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.

3.5 Generative Pre-trained Transformer-2

GPT-2 (**G**enerative **P**re-trained **T**ransformer-2) ist ein von OpenAI (**radford2019language**) veröffentlichtes autoregressives Sprachmodell auf Basis von Transformerdecodern. Als generatives Modell erzielte es hervorragende Ergebnisse beim Generieren von kongruenten Textpassagen.

GPT-2 small besteht aus 12 sequentiellen Transformerdecodern, die um die autoregressiven Eigenschaften beizubehalten, Masked Self-Attention Layer verwenden. Durch die Masked Self-Attention Layer kann das GPT-2 Modell beim Training im Gegensatz zu BERT lediglich die bereits bekannten Tokens betrachten. So lassen sich nacheinander jeweils einzelne nachfolgende Tokens zu einer Eingabesequenz generieren. Im nächsten Schritt wird diese Eingabesequenz und das generierte Token als nächste Eingabe verwendet.

4 Variational Autoencoder

Variational Autoencoder (**kingma2014autoencoding**) gehören zu den probabilistischen generativen Modelarchitekturen. In den letzten Jahren haben generative Modelle, insbesondere auch Variational Autoencoder, beeindruckende Möglichkeiten aufgezeigt, um hochrealistische Daten wie zum Beispiel Bilder, Text 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.

Normale Autoencoder bestehen aus einem Encoder und einem Decoder, 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. Autoencoder erzeugen nur einen diskreten Latentraum, wodurch Interpolation durch den Latentraum um neue Ausgaben zu rekonstruieren kaum möglich ist, da im Latentraum viele leere Stellen existieren. 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 Latentspaces legt. Somit entsteht für die einzelnen Datenpunkte ohne Regularisierung oft Overfitting.

Im Gegensatz zu normalen Autoencodern verwenden Variational Autoencoder einen probabilistischen Ansatz um Datenpunkte x im Latentspace Z zu repräsentieren. Der Encoder des Variational Autoencoder encodiert die Eingabe in eine multivariate Verteilung von der die Latentvektoren gesamplet werden. Die Latentvektoren werden anschließend vom Decoder decodiert und der Rekonstruktionserror zurückgegeben.

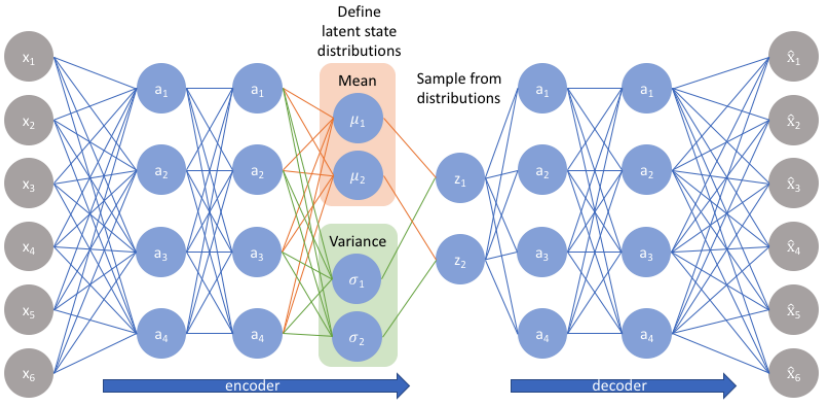


Abbildung 2: VAE Modellarchitektur

Die Beziehungen zwischen den Eingabedaten x und den Latentvektoren z sind wie folgt definiert, wobei ϕ die Parameter des Encoders und θ die Parameter des Decoders sind:

- Prior $p_\theta(z)$
- Likelihood $p_\theta(x|z)$
- Posterior $p_\theta(z|x) \approx q_\phi(z|x)$

Da die Posterior Verteilung $p_\theta(z|x)$ nicht bestimmt werden kann approximieren wir diese mit $q_\phi(z|x)$ unter Verwendung der Variationsinferenzmethode zur Minimierung der Kullback-Leibler-Divergenz, wie in Abschnitt 4.1 beschrieben.

Um mittels Variational Autoencoder neue Datenobjekte zu samplen wird zunächst mit der Prior-Verteilung $p_\theta(z)$ ein neuer Latentvektor $\hat{z}$ gezogen. Anschließend lässt sich ein neues Datenobjekt $\hat{x}$ mit der Verteilung $p_\theta(x|z = \hat{z})$ generieren.

4.1 Evidence Lower Bound

Bei Variational Autoencodern ist das Optimierungsziel die Minimierung des Rekonstruktionsfehlers zwischen Eingabe- und Ausgabedaten und die Approximation von $p_\theta(z|x)$ durch Minimierung der Kullback-Leibler-Divergenz $D_{KL}(q_\phi(z|x) \parallel p_\theta(z|x))$.

$$
\begin{aligned}
D_{KL}(q_\Phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x})) &= \int q_\Phi(\mathbf{z} \mid \mathbf{x}) \log \frac{q_\Phi(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z} \mid \mathbf{x})} \, d\mathbf{z} \\
&= \int q_\Phi(\mathbf{z} \mid \mathbf{x}) \log \frac{q_\Phi(\mathbf{z} \mid \mathbf{x}) p_\theta(\mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} \, d\mathbf{z} \\
&= \int q_\Phi(\mathbf{z} \mid \mathbf{x}) \left(\log(p_\theta(\mathbf{x})) + \log \frac{q_\Phi(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} \right) d\mathbf{z} \\
&= \log(p_\theta(\mathbf{x})) + \int q_\Phi(\mathbf{z} \mid \mathbf{x}) \log \frac{q_\Phi(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} \, d\mathbf{z} \\
&= \log(p_\theta(\mathbf{x})) + \int q_\Phi(\mathbf{z} \mid \mathbf{x}) \log \frac{q_\Phi(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z})} \, d\mathbf{z} \\
&= \log(p_\theta(\mathbf{x})) + E_{\mathbf{z} \sim q_\Phi(\mathbf{z}|\mathbf{x})}(\log \frac{q_\Phi(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z})} - \log(p_\theta(\mathbf{x} \mid \mathbf{z}))) \\
&= \log(p_\theta(\mathbf{x})) + D_{KL}(q_\Phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z})) - E_{\mathbf{z} \sim q_\Phi(\mathbf{z}|\mathbf{x})}(\log(p_\theta(\mathbf{x} \mid \mathbf{z})))
\end{aligned}
$$

der **E**vidence **L**ower **B**ound (ELBO).

ÜBERARBEITEN

4.1.1 Reparametisierung

Zum Training des Variational Autoencoder wird mittels Backpropagation der Fehler des Netzwerkes propagiert. Da das Samplen von $z/simq_\Phi(z|x)$ nicht deterministisch ist, kann hier keine Backpropagation durchgeführt werden. Um dennoch Backpropagation verwenden zu können wird durch Reparametisierung z durch eine deterministische Funktion $z = f_\Phi(x, \epsilon)$ dargestellt mit ϵ als externe unabhängig zufällige Hilfsvariable. Bei einer multivariaten Gaussverteilung für $q_\Phi(z|x)$ wäre die Reparametisierung wie folgt, wobei $\times$ die elementweise Multiplikation denotiert:

$$z \sim q_\Phi(z|x) = \mathcal{N}(z; \mu, \sigma\mathcal{I}) \tag{4}$$

$$z = \mu + \sigma \times \epsilon \text{ , mit } \epsilon \sim \mathcal{N}(0, \mathcal{I}) \tag{5}$$

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 Generation oft die globalen Eigenschaften vernachlässigt. 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.

Abhilfe schafft die Verwendung eines β-Variational Autoencoder 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.

$$L_\beta(\Phi, \theta, \beta) = -\mathbb{E}_{z\sim q_\Phi(\mathbf{z}|\mathbf{x})}[logp_\theta(\mathbf{x} \mid \mathbf{z})] - \beta D_{KL}(q_\Phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z})) \tag{6}$$

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 geplant 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 bereits 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) (**DBLP:journals/corr/abs-2004-04092**) ist ein großes vortrainiertes Deep Latent Variable Model für natürliche Sprache. Die Modelarchitektur von Optimus ist ein Variational Autoencoder, der als Encoder BERT und als Decoder GPT-2 verwendet wie in Abbildung 3 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 einen universellen Latentspace zu organisieren und somit übergreifende semantische Muster für die einzelnen Sätze zu finden. Somit kann durch gezieltes Verändern des Latentvektors z kontrolliert Text generiert werden. ÜBERARBEITEN

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 mit einer Vokabulargröße von 28.996. Die Ausgabe erfolge über die Byte Pair Encoding Tokenisierung von GPT-2 mit einer Vokabulargröße von 50.260. Innerhalb des Netzwerkes wird im Latentvektor ein Token durch eine Einbettung h_{Emb} die das Token, die Position und das

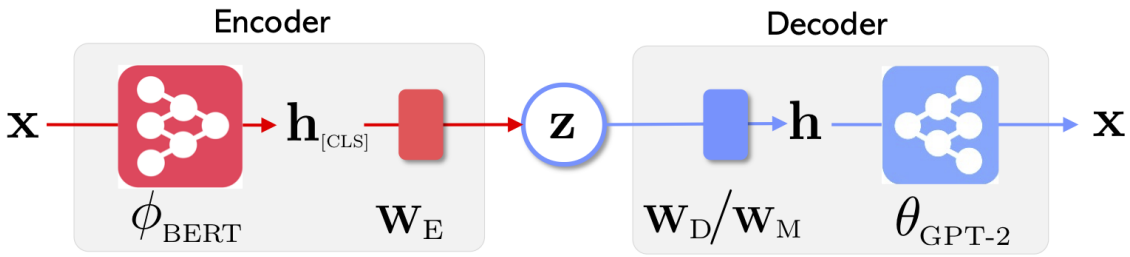
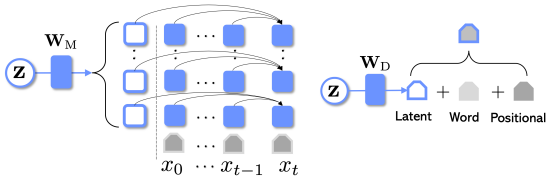


Abbildung 3: VAE Modellarchitektur von OPTIMUS mit BERT als Encoder und GPT-2 als Decoder

Segment Embedding wiedergibt, repräsentiert. Um beim Training den Loss des Rekonstruktionstask 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 multipliziert mit einer Gewichtsmatrix $W_E \in \mathbb{R}^{P \times H}$ gewählt. Somit kann ein Latentvektor wie folgt bestimmt werden $z = W_E h_{[CLS]}$.



(a) Memory (b) Embedding

Abbildung 4: Methoden um den Latentvector in GPT-2 zu injizieren

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 alten Embeddings addiert werden.

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 beim 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 Vector beschleunigt normalerweise den Decodiervorgang von GPT-2, da bei einem Decodierungsdurchlauf zu den vorherigen Tokens bereits entsprechende Key- und Valuevektoren in den Attentionlayern berechnet wurden. 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 (**DBLP:journals/corr/abs-1903-10145**) trainiert, um das KL Vanishing Problem beim Trainieren eines VAEs zu verhindern. Die β Variable, die wie in 4.2 erklärt den KL Regularisierer 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 für das nächste viertel der Trainingsmenge schrittweise β von 0 auf 1 erhöht und für das letzte viertel der Trainingsmenge auf 1 belassen um das VAE Model zu trainieren.

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

4.4 BiMeanVAE

BiMeanVAE ist ein von (**coop**) vorgestelltes Model 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 verwendet, um einen Latentvektor z zu erhalten. BiMeanVAE verwendet die Standard Variational Autoencoder Ziele, dass minimieren des Rekonstruktionsfehlers mit KL-Regularisierung verwendet. Die verwendete Prior Verteilung $p_\theta(z)$ ist die Gausssche Normalverteilung $\mathcal{N}(0, \mathcal{I})$. Beim Trainig wird ebenfalls das in Abschnitt 4.2 beschriebene Cyclical Annealing Verfahren verwendet, um graduell vom Autoencoder zum Variational Autoencoder Trainingsziel zu wechseln.

4.5 Latent Space Operations

5 Datensätze

Zum Training und zur Evaluation von neuronalen Netzen sind große Datensätze erforderlich. Für die Aufgabenstellung Durchschnittsrepräsentation 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 Restaurants zahlreiche Bewertungen.

Es existieren bereits bestehende Amazon und Yelp Datensätze mit menschlich erstellten Gold-Standard Zusammenfassungen. Das Variational Autoencoder Modell lässt sich somit mit den Review Daten auf Rekonstruktion trainieren.

5.1 Amazon Datensatz

Der Amazon Review Datensatz (**brazinskas2020-unsupervised**) umfasst nach dem Vorverarbeiten 4.807.338 Bewertungen zu 192.742 Produkten. Die Bewertungen wurden aus den Kategorien Kleidung, Schuhe, Schmuck, Elektronikartikel, Gesundheit- und Pflegeprodukte, Einrichtung und Küchenartikeln gewählt. Der Datensatz hat somit diverse Produkte, bei denen jeweils unterschiedliche Eigenschaften wichtig sind. Die entsprechenden Bewertungen sind bereits in Trainings- und Validierungsdaten gesplittet. Der Amazon Datensatz wurde gefiltert und es wurden nur Produkte mit mindestens 10 Bewertungen, die eine Länge zwischen 20 und 70 Wörter haben, ausgewählt. Des Weiteren enthält der Amazon Datensatz manuell erstellte Zusammenfassungen, die für den Dev / Test Split im Verhältnis von 28 / 32 vorliegen. Die meisten Bewertungen im Amazon Datensatz sind eher objektiv formuliert und beziehen sich auf die einzelnen Produktspezifischen Eigenschaften.

5.2 Yelp Datensatz

Der Yelp Review Datensatz (**pmlr-v97-chu19b**) besteht aus 1.126.653 Bewertungen zu 43.087 Restaurants, 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. Der Yelp Review Datensatz unterscheidet sich vom Amazon Datensatz dadurch, dass die Bewertungen wesentlich mehr persönliche Details und Erfahrungen der Nutzer enthalten. 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 meistens unnötige Zusatzinformationen, die nicht zur Bewertung des Restaurants an sich beitragen, wie zum Beispiel das Erwähnen von privat ausgetragenen Geburtstagsfeiern.

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 beim Multi-Document Zusammenfassen 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 weiteren Verlauf 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 Wörtern 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 representiert.

Insbesondere im Bereich der Multi-Review Summarization existieren viele unterschiedliche Ansätze. Zum einen existieren extraktive Ansätze, wie zum Beispiel LexRank, ein unüberwachter Algorithmus der repräsentative Sätze für eine Bewertung basierend auf Ihrer Zentralität in einem TF-IDF gewichteten Graphen selektiert. Zum anderen existieren abstraktive Ansätze wie zum Beispiel MeanSum, CopyCat oder COOP, 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 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, 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 r_i an sich zu betrachten. Durch den Zugriff auf die anderen Bewertungen r_i kann der Decoder spezifische Worte von diesen „kopieren“ und somit übernehmen.

6.1 Convex Aggregation for Opinion Summarization

Convex Aggregation for Opinion Summarization kurz COOP, 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.

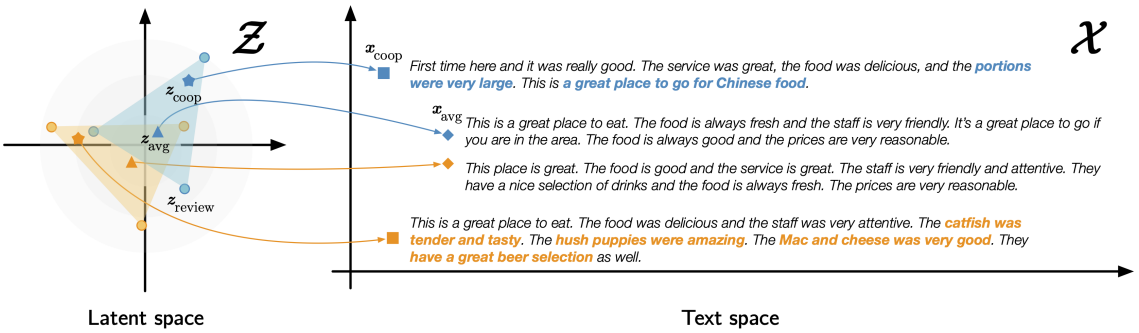


Abbildung 5: Latentraum Z mit den entsprechenden generierten Bewertungen X.

Wie in Abbildung 5 zu erkennen ist, ähneln sich Bewertungen die durch Bestimmung des Durchschnitts der Latentvektoren generiert worden sind sehr. Auffällig ist die Relation 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.

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

$$\max_{z} \quad Overlap(R_p, D_\theta(z)) \tag{7}$$

unter der Nebenbedingung:

$$z = \sum_{i=1}^{|R_p|} w_i z_i \tag{8}$$

$$\sum_{i=1}^{|R_p|} w_i = 1, \quad \forall w_i \in \mathbb{R}^+ \tag{9}$$

Optimiert wird das Maximieren des *input-ouput-word overlap* zwischen den Eingabebewertungen R_p und der generierten Bewertung $D_\theta(z)$. Diese *Overlap*-Metrik basiert auf dem Rouge-1 F1 Score und ermöglicht es Bewertungen zu generieren die konsistent zu den Eingabebewertungen sind. Die Suche der einzelnen Kombinationen wird auf die Potenzmenge der Eingabebewertungen R_p beschränkt. Der Zusammenfassunslatentvektor z_p wird anschließend aus dem Durchschnitt der ausgewählten Eingabebewertungslatent-

vektoren 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 von den einzelnen Latentvektoren, um ausdrucksstarke Bewertungen zu erhalten, erreicht State-of-the-Art Ergebnisse im Vergleich zu anderen Methoden, die den normalen Durchschnittslatentvektor verwenden. Die erzeugten Bewertungen sind gut 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.

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. COOP verwendet unter anderem das VAE Modell Optimus, welches zur Generation den kombinierten Latentvektor z mittels Decoder $p_\theta(x|z)$ zu einem Text $\hat{x}$ rekonstruiert. Als alternatives Modell wird BiMeanVAE vorgestellt, welches ein auf LSTM basierender Variational Autoencoder ist.

Es stellt sich die Frage ob diese Latentvektoren als Grundlage verwendet werden können, um durch weitere Optimierungen bessere Ergebnisse erzielen zu können.

Unkontrollierte Sprachmodelle modellieren Texte über die Wahrscheinlichkeit $p(X)$ für eine Sequenz $X = \{x_0, ..., x_n\}$. In Kapitel 3 wurde die Funktionsweise von Transformer-Sprachmodellen erklärt. Bei der Generation 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 Generation auf bereits vorher berechnete Key-, Value-Werte zurückgreifen zu können und somit effizienter Text generieren zu können.

Uber hat mit der Einführung von Plug and Play Language Models (**DBLP:journals/corr/abs-1912-02164**) es ermöglicht, die Textgeneration bei großen Sprachmodellen wie zum Beispiel GPT-2 kontrolliert zu beeinflussen. Kontrollierbare Generation 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 ist. Mit dem Satz von Bayes lässt sich das kontrollierbare Sprachmodell zu $p(x|a) \propto p(a|x)p(x)$ umformulieren. Das Attribut Modell $p(a|x)$ bewertet einen Satz x auf den Besitz eines Attributs a mit einer Wahrscheinlichkeit.

Zur kontrollierbaren Generation 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 $H_t = (H_t + \Delta H_t)$ wird die Wahrscheinlichkeit das nächste Token mit den gewünschten Attributen zu erhalten erhöht. Hierbei wird ΔH_t Schrittweise durch den Gradienten des Attribut-Modells errechnet und mit Null initialisiert. Um den Gradienten des Attribut-Modells bestimmen können wird dieses zu $p(a|H_t + \Delta H_t)$ umformuliert.

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

In der Gleichung gibt α die Schrittgröße und γ die Skalierung der Normalisierung an. Die Iteration kann merhmalig ausgeführt werden.

7.1 Verbessern der Textgeneration von Optimus

Den Variational Autoencoder Optimus mit einem Attributions-Modell zu kombinieren ist aufgrund der Injektion des Latentvektors schwierig. Optimus kann bereits unter Ein-

bezug des Latentvektors Texte kontrolliert generieren $p(x|z)$. Die Berücksichtigung eines Attribut-Models 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 folgende Iteration:

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

7.2 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 und den berechneten Hiddenstate h_t und Cellstate c_t. Um diesen LSTM Decoder mit einem Attributmodel zu optimieren bieten sich 3 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 Optionen einen Gradienten zur Veränderung der entsprechenden Variabel zu bestimmen. Nachfolgend wurden alle drei Optionen auf dem Amazon Datensatz evauliert, um die zu optimierende Variabel mit der bestmöglichen Performance zu finden.

	Amazon		
Method	**R1**	**R2**	**RL**
BIMEANVAE			
Baseline	39.50	8.62	22.79
optimze z			
optimze h_t			
optimze c_t	40.81	8.91	23.49

Tabelle 1: Ergebnisse für die Optimierung der unterschiedlichen Variablen z, h_t, c_t über ein Attributionsmodell

In Tabelle 1 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 hat sich ergeben, dass eine Optimierung der Variabel INSERT die größte Leistungssteigerung ergibt. Somit beschreibt folgende Gleichung den Optimierungsschritt Δc_t bei BIMEANVAE. CHANGE

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

7.3 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 ist die Log-Likelihood des Attribut-Modells:

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

Um den *Input-Output-Overlap* zwischen den Eingabereviews und den generierten Reviews zu maximieren wird das Bag of Words Modell erzeugt indem die k am häufigsten vorkommenden Wörter über alle Reviews eines Produktes gewählt werden. Vor dem auswählen der häufigsten vorkommenden Wörter werden von dieser Menge Stopwörter entfernt. Die besten Ergebnisse werden mit $k = 150$ erzielt.

Des Weiteren können die Bag of Words Tokens gewichtet werden, indem die k häufigsten Wörter nach ihrer Anzahl 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-Extraktion wie zum Beispiel durch YAKE (**CAMPOS2020257**) generiert werden.

Weiterhin kann die Bag of Words Menge bei der Generierung optimiert werden. So können bereits generierte Tokens nach einem Durchlauf aus der Bag of Words Menge entfernt werden, um zum Beispiel bei der Reviewgeneration einen Faktor nicht mehrmalig zu forcieren.

7.4 Latentvektoroptimisierung mit Beam Search

Beam Search ist ein Suchalgorithmus zur Auswahl des Ausgabesatzes mit der höchsten Wahrscheinlichkeit. Im Gegensatz zur normalen Greedy Search wird beim Beam Search Algorithmus die Gruppe mit den N bestmöglichen Tokens für eine Position ausgewählt. Über die Maximierung der bedingten Wahrscheinlichkeit wird final entschieden welche der zuvor ausgewählten Tokens

7.5 Moverscore Ranking

Der in Abschnitt 8.2 vorgestellte Moverscore ermöglicht einen semantischen Vergleich von Textsequenzen. So kann insbesondere Bewertungen mit anders formulierten Meinungen, die die gleiche Aussage treffen ein hoher Ähnlichkeitswert zugewiesen werden. Da dem Modell nur die Eingabebewertungen zur Verfügung stehen, ist eine Berechnung des *Input-Output-Overlap* lediglich über den ROUGE-1 Score nicht ausreichend. 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 für die generierten Ausgabebewertungen:

$$\text{SCORE}(\hat{x_i}) = Input\text{-}Output\text{-}Overlap(\hat{x_i}, \text{InputReviews}) + 1.5 \cdot Moverscore(\hat{x_i}, \text{InputReviews})$$

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

8 Evaluierung der Modelle

Die unterschiedlichen Optimierungen der Latentvektoren für ein Optimus Modell werden auf dem Amazon und dem Yelp Datensatz verglichen und mit aktuellen State-of-the-Art Modellen in Relation gesetzt. Es existieren im Dev-Datensatz jeweils 8 Eingabebewertungen und drei Gold-Summaries zum evaluieren der generierten Ausgabebewertungen. Die Eingabebewertungen werden durch ein Optimus VAE Modell in Latentvektoren umgewandelt, welche anschließend mittels in Abschnitt 6.1 erklärter COOP Herangehensweise kombiniert werden, um den *Input-Output-Overlap* zu maximieren. Weiterhin wird der Latentvektor mittels Attribut-Modell optimiert, um detailreichere Bewertungen zu erhalten.

8.1 Evaluationsmetriken

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

Zur Bewertung der generierten Bewertungen werden die vorhergesagten Ergebnisse mit den korrekten Ergebnissen verglichen. Die Konfusionsmatrix in Abbildung 6 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.

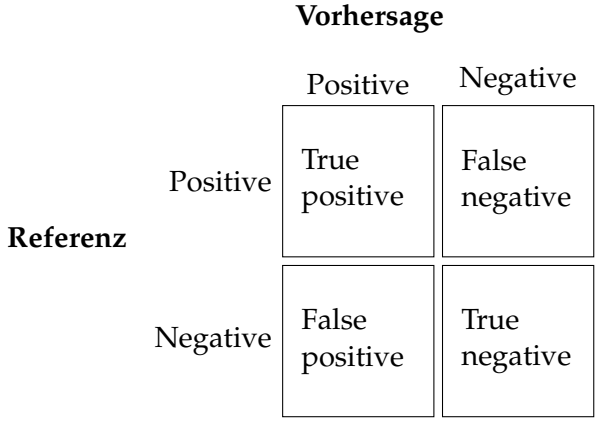


Abbildung 6: Konfusionsmatrix

Zur Berechnung des ROUGE-N Scores werden die einzelnen Textabschnitte in eine Menge aus N-Grams zerlegt. Mittels der Konfusionsmatrix in Abbildung 6 lassen sich Precision (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.

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

$\mathbf{F}_1$: 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 Unigrams, ROUGE-2 Bigramme und ROUGE-L misst die längste gleiche Subsequenz zwischen Vorhersage und Referenz.

Da ROUGE-Scores lediglich die einzelnen Wortsequenzen miteinander vergleicht, findet die semantische Bedeutung und Ähnlichkeit der Bewertungen mit der Referenz keinen Einfluss. Um trotzdem die semantische Ähnlichkeit zwischen Bewertungen und Referenz zu messen wird als weitere Metrik der Moverscore aus Abschnitt 8.2 verwendet. Moverscore basiert auf BERT und vergleicht Context Embeddings mittels Earth-Mover-Distance. Als Metrik konnte der Moverscore hohe Korrelationen mit menschlichem Urteilsvermögen aufweisen.

8.2 Moverscore

Der Moverscore (**moverscore_paper**) 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 nähsten 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, 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 Bewertungegenerierung ein wichtiger Score, 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.

8.3 Bewertung der Datensätze

8.3.1 Amazon-Datensatz

8.3.2 Yelp-Datensatz

8.4 Ergebnisse

	Amazon				Yelp			
Method	**R1**	**R2**	**RL**	**MV**	**R1**	**R2**	**RL**	**MV**
Our Method								
Optimus	**36.75**	**7.37**	<u>20.60</u>		**35.91**	**7.79**	<u>19.43</u>	
COOP								
BIMEANVAE⋆	<u>36.57</u>	<u>7.23</u>	**21.24**		<u>35.37</u>	<u>7.35</u>	**19.94**	
Optimus ⋆	35.32	6.22	19.84		33.60	7.00	18.95	
SimpleAvg								
BIMEANVAE⋆	33.60	6.64	20.87		32.87	6.93	19.89	
Optimus ⋆	33.54	6.18	19.34		31.23	6.48	18.27	
MeanSum ⋆	29.20	4.70	18.15		28.46	3.66	15.57	
CopyCat ⋆	31.97	5.81	20.16		29.47	5.26	18.09	
Extractive								
LexRank ⋆	28.74	5.47	16.75		25.01	3.62	14.67	

Tabelle 2: ROUGE Ergebnisse auf den Benchmarkdatensätze der unterschiedlichen Modelle. Die besten Ergebnisse sind fett markiert und die zweitbesten Ergebnisse unterstrichen. ⋆ denotiert, dass die Ergebnisse aus den Eregnissen von (**coop**) übernommen wurden.

9 Zusammenfassung und Ausblick

Abbildungsverzeichnis

Tabellenverzeichnis