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 **??** 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 **??** 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 **??** 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

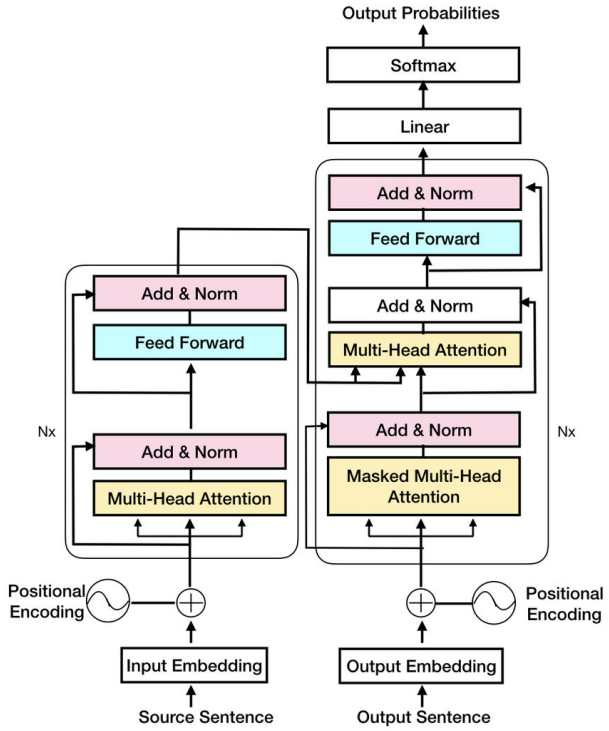


Abbildung 1: Transformer Encoder (links) und Transformer Decoder (rechts)

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 $c \in R^m$ für die Eingabedaten $x \in R^n$ im Latentspace zu finden. Die komprimierte Darstellung im Latentspace wird vom Decoder rekonstruiert mit dem Ziel eine möglichst identische Rekonstruktion zur Eingabe $x \in R^n$ 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.

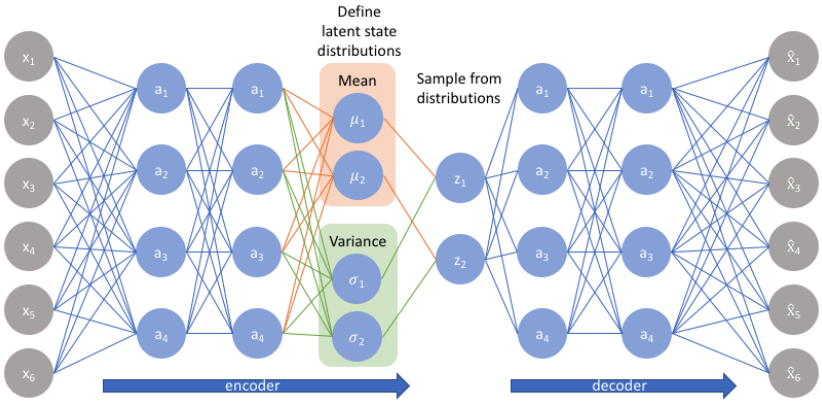


Abbildung 2: VAE Modellarchitektur

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 **??** 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:

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

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

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.

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

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 **??** 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

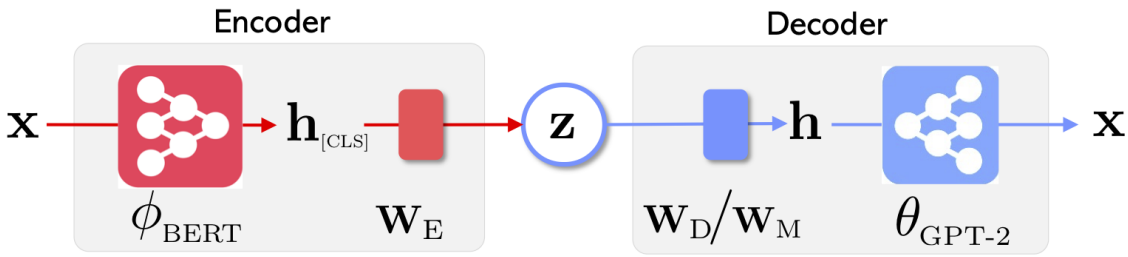
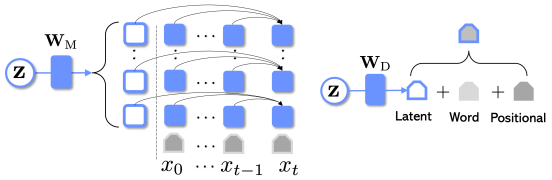


Abbildung 3: VAE Modellarchitektur von OPTIMUS mit BERT als Encoder und GPT-2 als Decoder

Segment Embedding wiedergibt, repräsentiert. Um beim Training den Loss des Rekonstruktionstask zu berechnen, werden die Sätze mit beiden Tokenisierungen tokenisiert.

Als Latentvektor $z \in R^P$ wird die gepoolte Ausgabe des letzten Hiddenlayers $h_{[CLS]} \in R^H$ von BERT multipliziert mit einer Gewichtsmatrix $W_E \in R^{P \times H}$ 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 $W_D \in R^{H \times P}$. Der Decoder kann hier die zusätzlichen Informationen des Embeddings beim Input und Output Layer verwenden.

Bei der Injezierung von z als Memoryvektor $h_{Mem} \in R^{L \times H}$ 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 $W_M \in R^{LH \times P}$ 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 **??** 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 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 Zusammenfassen von einzelnen Dokumenten enthalten mehrere Dokumente über ein Thema viele redundante Informationen, die so gefiltert werden müssen, dass diese Information nur einmal im Ausgabedokument dargestellt wird. Wiedersprüchliche Informationen sind ebenfalls eine Herausforderung bei der Zusammenfassung. Das Ziel von Multi-Document Summarization ist das Repräsentieren der wichtigen Aspekte aller Dokumente in einem einzelnen Dokument, um einen guten Überblick zu schaffen.

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.

MeanSum ist ein Autoencoder Modell mit LSTM Encoder und Decoder und errechnet zur Zusammenfassung von Bewertungen den Durchschnitt von den einzelnen Latentvektoren der Bewertungen.

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

7 Training von Optimus für Control Code Generation

8 Controllable Generation mittels Gradientenverschiebung

8.1 Gradientenverschiebung nach Bag of Words Klassifizierer

8.2 Kombiniert mit Beam Search

9 Evaluierung der Modelle

10 Zusammenfassung und Ausblick

Abbildungsverzeichnis

Tabellenverzeichnis